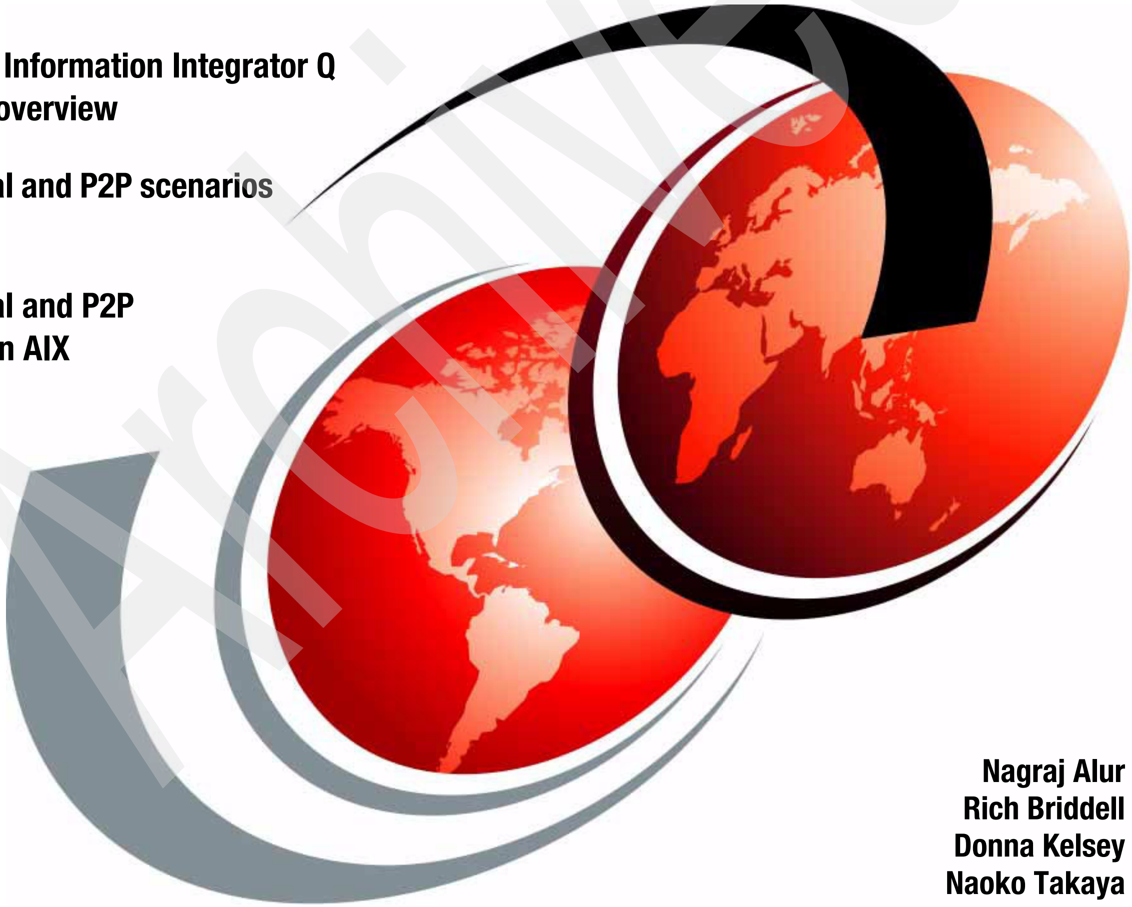
IBM

WebSphere Information Integrator Q Replication: Fast Track Implementation Scenarios

- WebSphere Information Integrator Q replication overview
- Bidirectional and P2P scenarios on z/OS
- Bidirectional and P2P scenarios on AIX



Nagraj Alur
Rich Briddell
Donna Kelsey
Naoko Takaya

Redbooks

ibm.com/redbooks

International Technical Support Organization

WebShphere Information Integrator Q Replication: Fast Track Implementation Scenarios

April 2005

SG24-6487-00

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

First Edition (April 2005)

This edition applies to Version 8, Release 2 of DB2 Information Integrator (product number 5724-C74).

Contents

Figures

Tables

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:

AIX®
DB2 Connect™
DB2®
DRDA®
@server®
@server®
Informix®
IBM®
ibm.com®
IMS™
iSeries™
Lotus Notes®
Lotus®
MVS™
Notes®
OS/390®
Redbooks™
Redbooks (logo) ™
RACF®
TXSeries®
WebSphere®
z/OS®

The following terms are trademarks of other companies:

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

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

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, and 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 using bidirectional and peer-to-peer replication solutions.

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

This book documents a step-by-step approach to implementing a bidirectional and two-way peer-to-peer replication high-availability architecture on z/OS and AIX platforms, simulating a financial industry environment.

> **Important:** WebSphere Information Integrator was formerly called DB2® Information Integrator. However, the product documentation still refers to the previous name of DB2 Information Integrator, and is referenced as such in this book. The DB2 Information Integrator name will be changed in the product documentation in the near future.

This book is organized as follows:

- Chapter 1, "Introduction to replication technologies" on page 1, provides an introduction to replication technologies in general, and describes the different flavors of replication, the product offerings available from IBM, and the pros and cons of each approach.
- Chapter 2, "WebSphere Information Integrator Q replication overview" on page 17, provides an overview of Q replication, and its architecture, processing flow, and key considerations in choosing a particular Q replication topology to address a business requirement.
- Chapter 3, "Bidirectional Q replication on z/OS platforms" on page 73, describes a step-by-step approach to implementing a high-availability bidirectional replication scenario involving two z/OS servers connected via a wide area network. Installation, configuration, and failover/switchback details are described.
- Chapter 4, "Peer-to-peer Q replication on z/OS platforms" on page 295, describes a step-by-step approach to implementing a high-availability two-way peer-to-peer replication scenario involving two z/OS servers connected via a wide area network. Installation, configuration, and failover/switchback details are described.

- Chapter 5, “Bidirectional Q replication on AIX platforms” on page 419, describes a step-by-step approach to implementing a high-availability bidirectional replication scenario involving two AIX servers connected via a local network. Installation, configuration, and failover/switchback details are described.
- Chapter 6, “Peer-to-peer Q replication on AIX platforms” on page 543, describes a step-by-step approach to implementing a high-availability two-way peer-to-peer replication scenario involving two AIX servers connected via a local network. Installation, configuration, and failover/switchback details are described.
- Appendix A, “WebSphere MQ overview” on page 721, provides a high-level overview of WebSphere MQ with particular emphasis on its relationship to Q replication.
- Appendix B, “Template for topology and configuration information” on page 739, describes the template to be used for collecting all topology and configuration information prior to commencing the setup of the bidirectional or two-way peer-to-peer Q replication environment.
- Appendix C, “Dead letter queues in a Q replication environment” on page 751, provides an overview of the dead letter queues in WebSphere MQ and their possible applicability in a Q replication environment.
- Appendix D, “Common troubleshooting setup problems” on page 763, describes the steps for diagnosing and resolving commonly occurring problems during the setup of a Q replication environment.
- Appendix E, “Startup and shutdown scripts for WebSphere MQ and Q replication” on page 777, contains sample scripts for starting and shutting down WebSphere MQ and Q replication.
- Appendix F, “Tables used in Q replication scenarios” on page 781, contains the DDL for the tables used in the Q replication bidirectional and peer-to-peer topology scenarios.
- Appendix G, “Communicating with Q replication programs in z/OS” on page 873, provides an overview of some of the methods available for communicating with Q replication programs in a z/OS environment.
- Appendix H, “Cataloging remote database servers” on page 881, describes the process for cataloging remote database servers in the Replication Center.

The team that wrote this redbook

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

Nagraj Alur is a Project Leader with the IBM International Technical Support Organization, San Jose Center. He holds a Masters Degree in Computer Science from the Indian Institute of Technology (IIT), Mumbai, India. He has more than 30 years of experience in DBMSs, and has been a programmer, systems analyst, project leader, independent consultant, and researcher. His areas of expertise include DBMSs, data warehousing, distributed systems management, and database performance, 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.

Rich Briddell is a Consulting IT Specialist with IBM Software Group, Lab Services, specializing in Information Integration, Migration and Distributed OLTP, working out of St. Louis, MO. He has 15 years of experience in the DBMS field and has been a development team lead and independent consultant. He holds a Master of Arts degree in Operations Management from the University of Arkansas and a Master of Science from Webster University in Computer Resources and Information Management. His areas of expertise include Multi-level Secure projects for the U.S. Government and extensive experience in the Telecommunications industry. He is an IBM Certified Solutions Expert for Informix® Dynamic Server V9 and V7, an IBM Certified Advanced Database Administrator for DB2 UDB database systems, an IBM Certified Application Developer for DB2 UDB Family, and an IBM Certified Specialist for WebSphere Application Server.

Donna Kelsey is an Advisory Software Engineer with IBM Software Group (DB2 and IMS™ Tools), Boulder, CO. She holds a Bachelor of Science Degree in Mathematics from the University of Nebraska. She has more than 20 years of experience in DBMSs, and has been a programmer, systems analyst, database administrator, database system programmer, customer support representative, development architect, and researcher. Recently she has spent seven years in support and development of data replication technologies. She has contributed to manuals and papers on data replication, and has taught customer classes on the subject.

Naoko Takaya is a Senior IT Engineer with IBM Systems Engineering Co., Ltd. in Japan. She holds a Masters Degree in Computer Science from Ritsumeikan University, Japan. She has two years of experience in DBMSs and has been in technical support for information management products. She specializes in DB2 UDB, and WebSphere Information Integrator and Replication on Windows® and AIX. She has experience in teaching WebSphere Information Integrator and SQL Replication in Japan.

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

Nicolas Adiba
Jaime Anaya
Doreen Fogle
Beth Hamel
Kevin Lau
Madhu Kochar
Somil Kulkarni
Jayanti Mahapatra
Patricia Shimer
Kris Tachibana
Sundari Voruganti
IBM Silicon Valley Laboratory

Rick Levin
Peter Miller
Cathy Wolfenbarger
IBM SEEL Silicon Valley LaboratorySan Jose

Pav Kumar-Chatterjee
IBM UK - Product Introduction Center

Emma Jacobs
International Technical Support Organization, San Jose Center

David Bennin
Richard Conway
Julie Czubik
International Technical Support Organization, Poughkeepsie Center

Become a published author

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

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

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

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

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

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

  ibm.com/redbooks

- Send your comments in an email to:

  redbook@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. QXXE Building 80-E2
  650 Harry Road
  San Jose, California 95120-6099

1

Introduction to replication technologies

In this chapter we introduce the replication concept, describe the various flavors of replication, and list the various replication-related product offerings available from IBM. We also briefly describe the key criteria in choosing one product offering over the other.

The topics covered are:

- Business of replication
- Replication options
- Asynchronous replication options
- IBM product offerings
- SQL replication, Q replication, and HADR comparison

1.1 Business of replication

Organizations copy data for a number of reasons, including the need for disaster recovery, information data warehousing, security, availability, and performance. The type of copy and the techniques that may be used to create them depend upon the requirements of the application accessing the copy. For example:

- Copies made to support disaster recovery are most likely mirror images of the source data and do not involve aggregations, summarizations, or data transformations. The copies are maintained at a remote site and are likely to be “near real time” (very low latency).
- An information warehouse supports strategic business decision making. Data in information warehouses tend to represent points in time and/or continuous history, be subsets of source data, or be merged from multiple potentially heterogeneous data sources. The data would most likely need to be transformed for performance reasons or to make the data more user friendly. Transformations may include code translations, aggregations, derived data, or removal of data of a sensitive nature not appropriate to strategic decision making.
- Copies made for improved availability, performance, machine capacity limitations, or isolation from critical production systems tend to have requirements that are permutations and combinations of disaster recovery and information warehouse needs.

Copies are distinct from the sources from which they are generated, and users generally[1] access copies as independent entities.

1.2 Replication options

Replication independence is a term used in distributed database technology that refers to a scenario where the distributed DBMS maintains one or more copies (primarily for read performance and availability reasons) of the same data. The user querying the data perceives the data as a single logical entity and is totally unaware of the existence of the multiple DBMS copies[2]. Replication refers to the process of maintaining these copies transparent to the user.

Two forms of replication exist, as follows:

- Asynchronous replication

[1] Exceptions to this are materialized query tables (MQTs), which are a form of copies, but are routed to automatically by relational optimizers to improve query performance when a user query accesses the underlying data sources. For more details on MQTs, please refer to the *DB2 UDB’s High Function Business Intelligence for e-business*, SG24-6526.
[2] MQTs provide replication independence.

It describes a process where, at a given instant in time, the copy and the source data are out of sync with each other. The delay (or latency) can range from a few seconds or a few minutes to many hours.

Asynchronous replication has the potential advantages of:

– Being able to cope with the unavailability of the target system or communication system

– Minimizing the overhead on the updating transaction in the source system

– Enabling subsetting and merging of source data, and data transformations

– Using optimization algorithms such as grouping multiple updates to the same row on the source system to a single update on the copy

The potential disadvantage is that the copies are out of sync with the sources from which they are generated. This may not necessarily be a disadvantage in the case of information warehouse applications that require point-in-time copies rather than "real time" or "near real-time" data consistency. For other types of applications, the issue may well be the degree of latency provided by a particular vendor solution.

- Synchronous replication

  It describes a process where the copy and the data in the sources from which it is generated are in lock-step at a unit-of-work level. A two-phase commit process is essential to implementing synchronous replication.

  The potential advantage of synchronous replication is the up-to-the-second consistency of the copy, which might be critical in a financial application.

  The disadvantages of synchronous replication are:

  – It imposes an overhead on the updating transaction on the source system, which increases exponentially with the number of copies being maintained.

  – It is dependent on the availability of the target system and the communication system between the source and the target systems.

Given the wider applicability of asynchronous replication technology and the potential disadvantages of synchronous replication, almost all replication product offerings in the market only support asynchronous replication.

> **Important:** Almost a decade ago, vendors, consultants, and the trade press blurred the earlier precise definition of replication. Today, replication is the term used to describe the process of generating a distinct copy (in the user's perception) from one or more sources, and delivering them to one or more sites. The implicit assumption of replication is asynchronous replication.

1.3 Asynchronous replication options

Vendors support a variety of options in their asynchronous replication offerings. They include the following:

- Unidirectional replication

  In unidirectional replication, changes that occur at a single data source may be replicated to one or more targets, with or without any intervening data transformations occurring, as shown in Figure 1-1. Data transformations may involve subsets of rows and columns, aggregations, and summarizations. The targets are invariably read only.

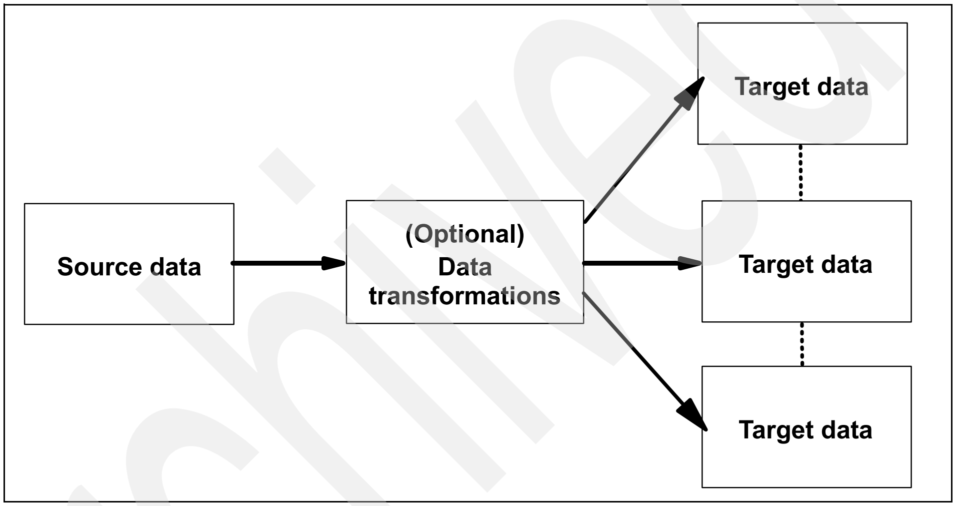
  

  Figure 1-1 Unidirectional replication

  Unidirectional replication may be used in information warehouse applications, for addressing machine capacity limitations, or isolating lower priority applications from critical production systems.

- Bidirectional replication

  With bidirectional replication, two data sources are involved with changes occurring at either source being replicated to the other, as shown in Figure 1-2.

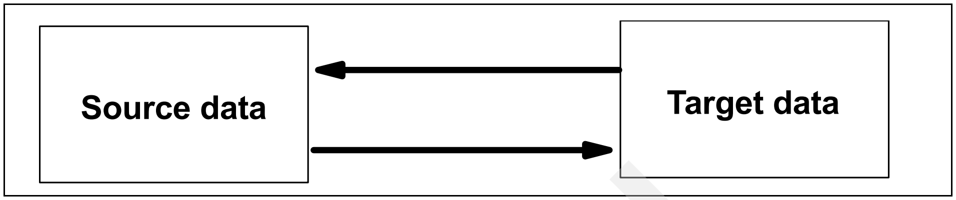


Figure 1-2 Bidirectional replication

A bidirectional replication implementation can result in conflicts when both sources update the same data element, and appropriate conflict detection and resolution algorithms need to be available to choose the winner and loser. Vendors provide varying degrees of complexity with their conflict detection and resolution algorithms in their products. A master/slave relationship tends to exist between the two data sources. The data sources involved have identical structures (columns and data types) but may have a subset of rows at one site.

Bidirectional replication may be appropriate in high-availability scenarios where a secondary site is ready to take over on the failure of a primary site. It is also appropriate in toggle replication scenarios such as implemented in Lotus® Notes® replication.

- Peer-to-peer replication

  With peer-to-peer replication, multiple data sources are involved with simultaneous updates occurring at each source, as shown in Figure 1-3 and Figure 1-4.

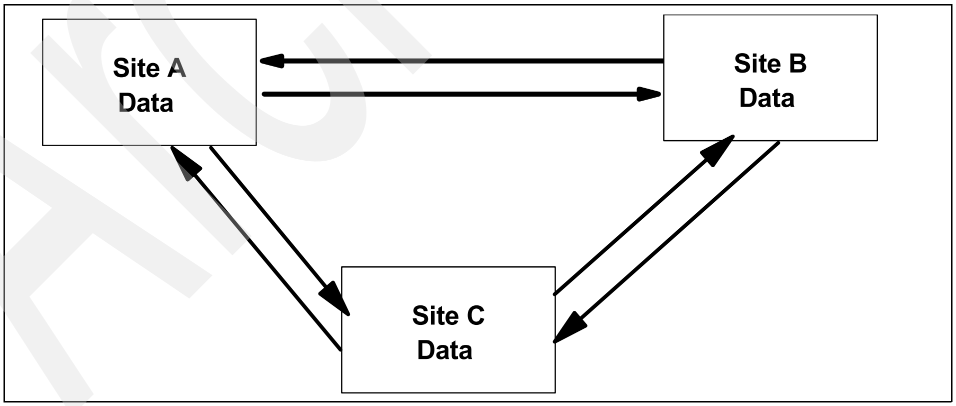


Figure 1-3 Peer-to-peer replication

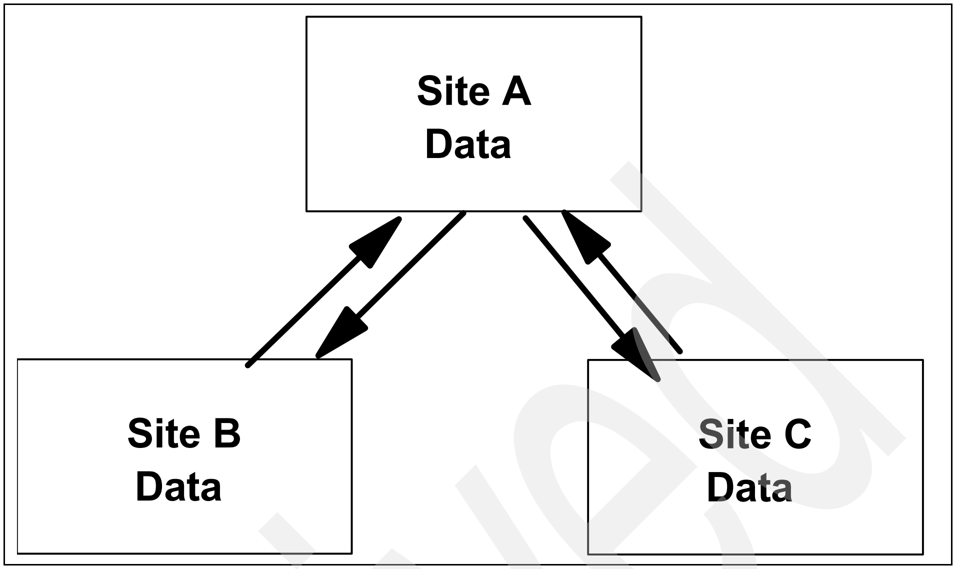


Figure 1-4 Hub and spoke model of peer-to-peer replication

> **Note:** In the hub and spoke model shown in Figure 1-4, any of the replicas can be updated simultaneously. Any resulting conflicts are resolved by the node higher in the hierarchy.

Peer-to-peer replication may be considered to be an extension of the bidirectional replication model. As in the case of bidirectional replication, conflicts may occur and will need to be detected and resolved. Here again, vendors provide varying degrees of complexity with their conflict detection and resolution algorithms in their products. Again, as in the case of bidirectional replication, the data sources involved have identical structures (columns and data types), but unlike bidirectional replication, do not support subsetting of the rows.

Peer-to-peer replication is appropriate in scalability scenarios where an application may be directed to any of the data sources based on workload management. Peer-to-peer replication may also be used in the same scalability scenario as a high-availability solution, where the loss of one of the sites is not catastrophic, since the remaining sites shoulder the added burden until the failing site is restored and resynchronized with the other sites in the set.

1.4 IBM product offerings

IBM provides three replication-related offerings that address different application requirements but have some overlapping capabilities, as follows:

- SQL replication (Data Propagator)

  IBM's first replication product that is available on the z/OS, Windows, UNIX®, and Linux® platforms.

  Figure 1-5 shows a simple configuration in SQL replication.

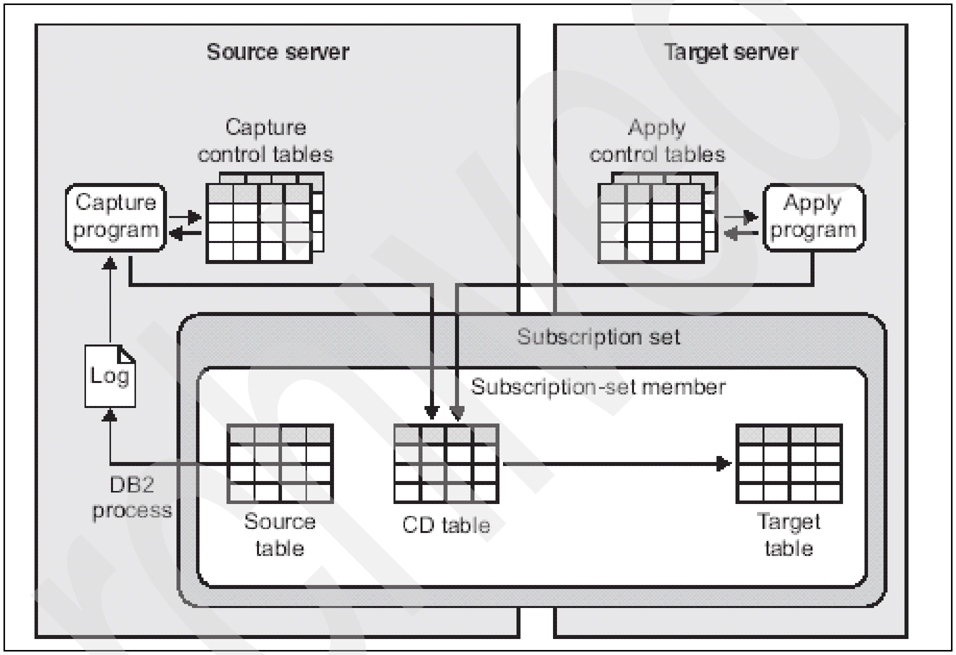


Figure 1-5 Simple configuration in SQL replication

  SQL replication allows one to replicate data from DB2® UDB sources to targets by using two programs, Capture and Apply.

  - The Capture program runs on the source system. The Capture program reads DB2 recovery logs for changed source data and saves the committed changed data to staging tables.
  - The Apply program typically runs on the target system. The Apply program retrieves captured data from staging tables and delivers the data to targets.

Both programs use a set of DB2 control tables to track the information that they require to perform their tasks, including monitoring the SQL replication environment.

The Capture program uses a set of DB2 UDB Capture control tables that contain information about replication sources and the current position of the Capture program in the DB2 recovery log. In most cases, the control tables for a Capture program need to be on the same DB2 UDB server as the sources associated with the program. If multiple Capture programs are used on the same DB2 UDB server, they each use their own set of Capture control tables. The schema associated with a set of Capture control tables identifies the Capture program that uses those control tables. This schema is called a Capture schema.

Note: If the data sources involved are non-DB2 relational databases, then triggers are used to capture changes.

The Apply program uses a set of DB2 UDB Apply control tables that contain information about the targets and where their corresponding sources are located. The control tables for the Apply program usually reside on the system where the Apply program runs. Unlike the case with Capture programs, multiple Apply programs may use the same set of control tables. Each Apply program is identified in these control tables by a name called an Apply qualifier.

Some of the main features of SQL replication are as follows:

– z/OS, UNIX, Windows, and Linux platforms support.

– Replication of LOBs and DB2 Spatial Extender data.

– Unidirectional (one source to one or more targets), bidirectional[3] (two or more servers involving the DB2 family only).

  Note: SQL replication only supports peer-to-peer replication through manual modifications of the control tables.

  In the bidirectional implementation (called *update-anywhere*), one of the servers is designated as the MASTER while the rest are designated as REPLICA or slaves. Changes made at the replica servers are copied to the master server. Changes made at the master server are copied to the replica server, including changes that came to the master server from the other replicas. This is a hierarchic form of conflict detection and resolution.

[3] In SQL replication, bidirectional can involve more than two servers.

Multi-tier replication in SQL replication is a special replication configuration in which changes are replicated from a replication source in one database to a replication target in another database, and changes from this replication target are replicated again to a replication target in another database.

- Data transformation including row and column subsets, as well as aggregations and summarizations with unidirectional support.
- Changes to multiple tables originating within a single transaction at the source are applied to their corresponding target tables outside the originating transaction scope.
- Supports continuous, interval-timing, or event-timing scheduling.
- Replication Alert Monitor to monitor replication.
- Heterogeneous database support including Oracle and SQL Server.

For complete details of SQL replication, refer to the *A Practical Guide to DB2 UDB Data Replication V8*, SG24-6828, and *IBM DB2 Information Integrator SQL Replication Guide and Reference Version 8.2*, SC27-1121-02.

- Q replication

  IBM's most recent offering is also available on z/OS, Windows, UNIX, and Linux platforms.

  Figure 1-6 shows a simple configuration in Q replication.

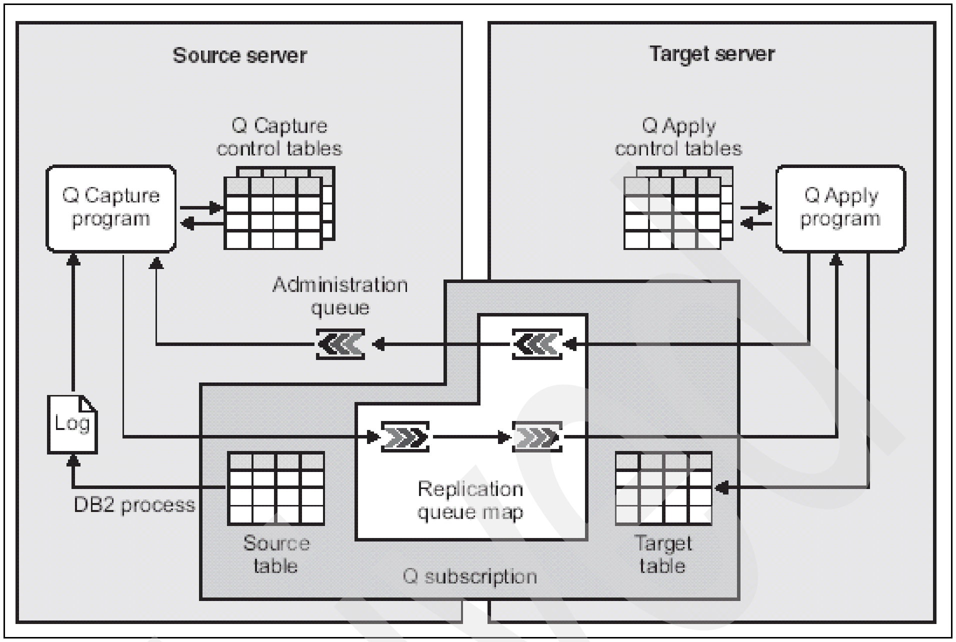


Figure 1-6 Simple configuration in Q replication

Just as in the case of SQL replication, Q replication allows one to replicate committed transactional data from DB2® UDB sources to targets by using two programs: Q Capture and Q Apply. However, there are a number of subtle differences in how replication is performed.

– The Q Capture program runs on the source system, reads DB2 recovery logs for changed source data, and writes the changes to WebSphere® MQ queues.

– The Q Apply program runs on the target system, retrieves captured changes from queues, and writes the changes to targets.

Both the Capture and Apply programs use a set of DB2 UDB control tables to track the information required to perform their tasks, including monitoring the Q replication environment.

The Q Capture program uses a set of control tables called Q Capture control tables that contain information about replication sources, the targets that correspond to them, and the WebSphere MQ queue manager and queues used by the Q Capture program. If multiple Q Capture programs are used on the same DB2 UDB server, they each use their own set of control tables. The schema associated with a set of Q Capture control tables identifies the Q Capture program that uses those control tables. This schema is called a Q Capture schema.

The Q Apply program uses a set of control tables called Q Apply control tables that contain information about targets, where their corresponding sources are located, and information about the WebSphere MQ queue manager and queues used by the Q Apply program. As with Q Capture programs, multiple Q Apply programs may be executed on the same DB2 UDB server. In this case, each Q Apply program uses its own set of control tables. The schema associated with a set of Q Apply control tables identifies the Q Apply program that uses those control tables. This schema is called a Q Apply schema.

Important: The control tables for the Q Capture and Q Apply programs that are on each individual server must have the same schema name. For example, if one has a server named SERVER_RED and a server named SERVER_GREEN, then the Q Capture and Q Apply programs that are on SERVER_RED must both have the same schema, and the Q Capture and Q Apply programs that are on SERVER_GREEN must both have the same schema.

Note: Q replication V8.2 only supports the DB2 family. However, fixpak 9, which is due to be out in April 2005, provides support for non-DB2 targets as well.

Some of the main features of Q replication are as follows:

- Unidirectional (one source to one or more targets), bidirectional (two servers only), and peer-to-peer (two or more servers) support.

  In the bidirectional implementation, one of the servers is designated as the primary while the other is designated as the secondary. Changes made at either server are copied to the other server. Any conflicts are resolved by designating either the primary or secondary (during configuration) as the consistent winner.

  In the peer-to-peer implementation, there is no concept of a primary or secondary, as each server is treated equally. Conflict detection and resolution in this case is based on timestamps.
- Data transformation support using a stored procedure as the target in unidirectional replication only.
- Subsetting of rows and columns is supported in the unidirectional scenario.
- Changes to multiple tables originating within a single transaction at the source are applied to their corresponding target tables within the same originating transaction scope.

  - Parallel apply of transactions at the target.
  - Supports referential integrity constraints on the source and target tables.
  - Online load feature that allows adding a new table to be replicated without impacting other tables being replicated.
  - Supports continuous replication only. However, the DBA can manually stop and start the browser to support “scheduled” replication.
  - No restrictions in peer-to-peer regarding key updates and deletes.
  - Replication Alert Monitor to monitor replication.
  - Replication of LOBs.
  - z/OS, UNIX, Windows, and Linux platform support.

  For complete details about Q replication, refer to *IBM DB2 Information Integrator Replication and Event Publishing Guide and Reference Version 8.2*, SC18-7568.

- High Availability Disaster Recovery (HADR)

  DB2 high availability disaster recovery (HADR) is a data replication feature of DB2 UDB 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. When a failure occurs on the primary, one can fail over to the standby. The standby then becomes the new primary. Since the standby database server is already online, failover can be accomplished very quickly, resulting in minimal down time. HADR became available in DB2 UDB V8.2.

  Figure 1-7 shows a simple configuration in HADR.

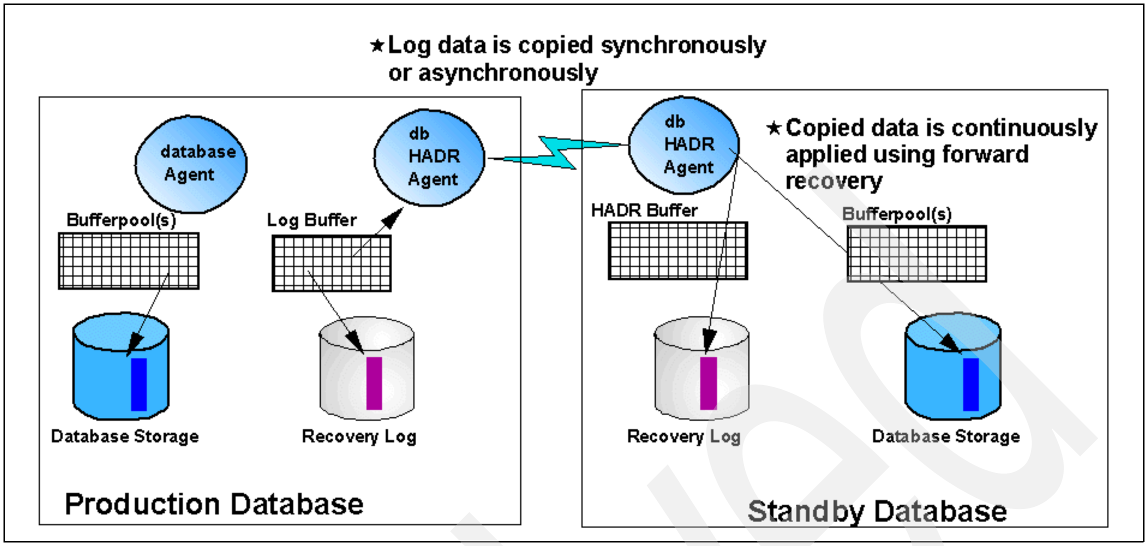


Figure 1-7 HADR overview

A database that does not use HADR is referred to as a standard database. HADR might be the best option if most or all of the database requires protection, or if DDL operations are performed that must be automatically replicated on the standby database. Applications can only access the current primary database. Updates to the standby database occur by rolling forward log data that is generated on the primary database and shipped to the standby 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. Further, you can redirect the clients that were using the original primary database to the standby database (new primary database) by using automatic client reroute 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, the primary database might be located at the head office in one city, while the standby database is located at a 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, one can bring the original primary database back up and return it to its primary database status; this is known as fallback.

With HADR, one can choose the level of protection one wants from potential loss of data by specifying one of three synchronization modes: Synchronous, near synchronous, or asynchronous. After the failed original primary server is repaired, it can rejoin the HADR pair as a standby database if the two copies of the database can be made consistent. After the original primary database is reintegrated into the HADR pair as the standby database, one can switch the roles of the databases to enable the original primary database to once again be the primary database

For complete details about HADR, refer to *IBM DB2 UDB Data Recovery and High Availability Guide and Reference Version 8.2*, SC09-4831-01.

1.5 SQL replication, Q replication, and HADR comparison

Table 1-1 provides a very high-level overview of some of the key criteria that should be considered in choosing between SQL replication, Q replication, and HADR to address a particular business requirement.

> **Important:** Table 1-1 is only meant to provide readers with a cursory perception of the differences between the three offerings, and readers are strongly urged to refer to more detailed sources of documentation, building prototypes and soliciting third-party experiences before making a product choice for their business requirement.

Table 1-1 SQL replication, Q replication, and HADR comparison

Comparison criteria	SQL replication	Q replication	HADR
Applicability	Multiple uses including: ► Populating data warehouses and data marts ► Offloading processing (capacity relief) ► Auditing change history	Multiple uses including: ► Data availability for planned, rolling upgrades, or outages—failover capabilities ► Offloading processing (capacity relief) ► Support for geographically distributed applications	Failover from partial or complete site failure

Comparison criteria	SQL replication	Q replication	HADR
Synchronization modes used	Asynchronous	Asynchronous	► Synchronous ► Near synchronous ► Asynchronous
Replication models supported	Unidirectional, bidirectional, and peer-to-peer	Unidirectional, bidirectional, and peer-to-peer	Unidirectional
Operating systems supported	Linux, UNIX®, Windows®, z/OS™, iSeries™	Linux, UNIX®, Windows®, z/OS™	Linux, UNIX®, Windows®
Source and target combinations	Heterogeneous	Homogeneous for source; heterogeneous for targets[a]	Homogeneous
DBMSs supported	DB2 UDB, Oracle, Sybase, Teradata, Microsoft® SQL Server, Informix	DB2 UDB with WebSphere Information Integrator	DB2 UDB ESE only
Sources supported	Tables or views	DB2 tables	DB2 database
Targets supported	Tables or views	DB2 tables	DB2 database
Target accessible during replication	Yes	Yes	No
Source/target mapping	► Column and row filtering: – Filtering at source, target,or both ► Data cleansing, data aggregation, and calculated columns in target tables. ► Manipulate data once and replicate to many targets, as well as manipulate data and replicate to selected targets.	► Column and row filtering (unidirectional only) – Filtering at source only ► Data transformation using stored procedure (unidirectional only)	No
SQL DDL replicated	No	No	Yes
How data is replicated	Committed transaction data is captured and stored in staging tables by a Capture program. An Apply program reads the information from the staging tables and applies the committed data to the target tables.	Committed transaction data is captured and put on WebSphere® MQ message queues by a Capture program. An Apply program reads the information from the message queues and applies the transactionally consistent committed data to the target tables.	Log operations from the log buffer (not logs on disk) are shipped to the standby database and replayed continuously through forward recovery.

Comparison criteria	SQL replication	Q replication	HADR
Reference documentation	► *A Practical Guide to DB2 UDB Data Replication V8*, SG24-6828 ► *IBM DB2 Information Integrator SQL Replication Guide and Reference Version 8.2*, SC27-1121-02 ► http://www.ibm.com/software/data/integration/db2ii/support.html ► http://www-306.ibm.com/software/data/db2/udb/support/manualsv8.html ► http://publib.boulder.ibm.com/infocenter/db2help/index.jsp	► *IBM DB2 Information Integrator Replication and Event Publishing Guide and Reference Version 8.2*, SC18-7568 ► http://www.ibm.com/software/data/integration/db2ii/support.html ► http://www-306.ibm.com/software/data/db2/udb/support/manualsv8.html ► http://publib.boulder.ibm.com/infocenter/db2help/index.jsp	► *IBM DB2 UDB Data Recovery and High Availability Guide and Reference Version 8.2*, SC09-4831-01 ► http://www-306.ibm.com/software/data/db2/udb/support/manualsv8.html ► ftp://ftp.software.ibm.com/ps/products/db2/info/vr82/pdf/en_US/db2hae81.pdf

a. Oracle and Sybase in fixpak9; due April 2005

2

WebSphere Information Integrator Q replication overview

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

The topics covered are:

- Q replication overview
- Q replication processing flow
- Choosing a particular Q replication topology
- Best practices 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 Information Integrator was formerly called DB2 Information Integrator. However, the product documentation still refers to the previous name of DB2 Information Integrator, and is referenced as such in this book. The DB2 Information Integrator name will be changed in the product documentation in the near future.

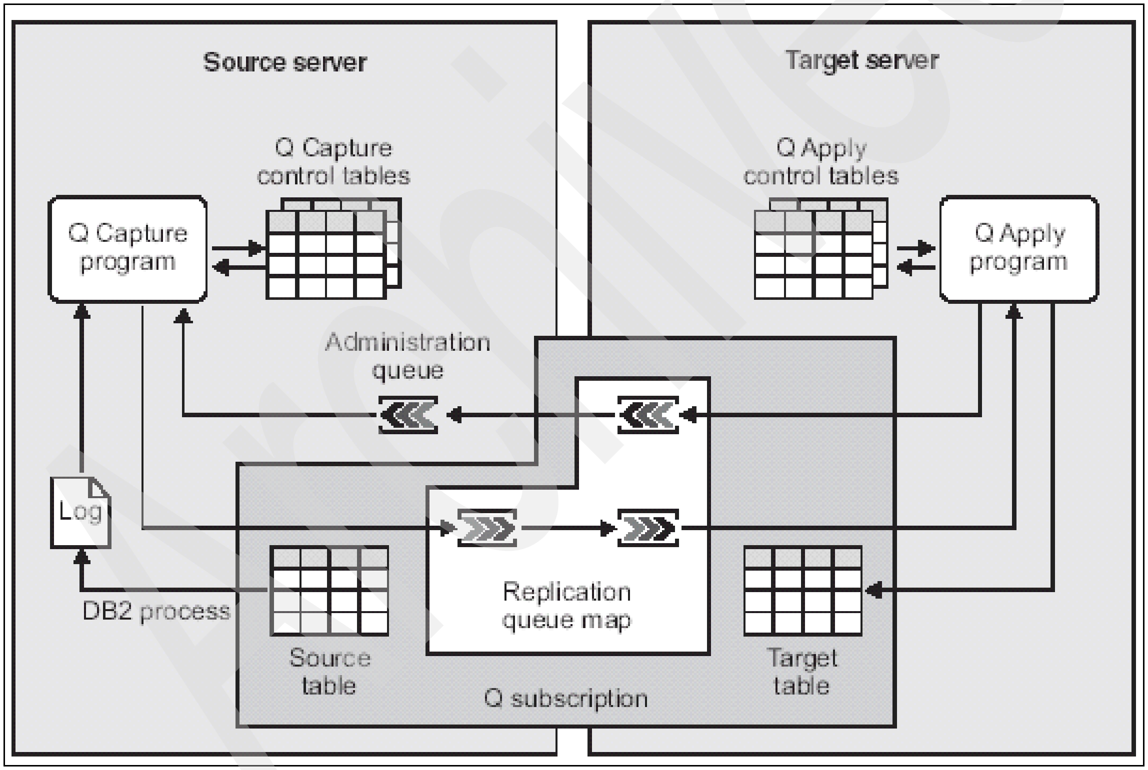


Figure 2-1 A simple Q replication configuration

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 targets 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 either of the replication programs is stopped, messages remain on queues to be processed whenever the program is ready. Because the messages are persistent, the source and target remain synchronized even in the event of 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:

- Replication queue maps, which identify the WebSphere MQ queues to be used for sending and receiving data
- Q subscriptions, which identify options such as the rows and columns to be replicated or published, and the options for loading target tables
- WebSphere MQ sender and receiver channels, transmit queues, and model/local/remote queues

For a complete description of these Q replication objects, refer to the *IBM DB2 Information Integrator Replication and Event Publishing Guide and Reference Version 8.2*, SC18-7568.

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 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.
- 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.
- 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.

2.1.4 Replication Alert Monitor

The Replication Alert Monitor is a program that checks the status of the replication environment. When the Replication Alert Monitor is running, it automatically checks the status of replication and notifies designated persons about certain conditions that occur in the replication environment. For example, the Replication Alert Monitor can send a notification when any Q Capture program deactivates a Q subscription, or when a threshold has been exceeded in your replication environment. For example, when the QCAPTURE_MEMORY alert condition or QAPPLY_QDEPTH alert condition is specified, the Replication Alert Monitor will send a notification any time the Q Capture program uses more memory than its threshold allows, or the depth of the receive queue used by Q Apply exceeds the threshold specified.

One can check the status of the replication environment by using the following methods:

- View Replication Center windows that report statistics about the Q Capture and Q Apply programs.
- Run select statements on the control tables to view statistics about the operation of these programs.

The Replication Alert Monitor automatically monitors the replication environment across all of one's operating systems. It checks replication on one's servers and automatically alerts one to conditions that require attention.

The Replication Alert Monitor may be configured in one of two ways—run one monitor or multiple monitors. Consider a single monitor when a few replication programs need to be monitored. Multiple monitors should be considered when many replication programs need to be monitored, or to prioritize the monitoring of certain programs, or split up the monitoring workload. Setting up multiple monitors is like cloning the Replication Alert Monitor. When multiple monitors are set up, the control information for each monitor is stored on the server that it is assigned to monitor. When a single monitor is set up, all the control information is stored on one server. Each monitor can monitor multiple replication programs, but the monitor checks for alerts on each server one at a time. It must check all of the other servers that it monitors before it returns to any one server.

The server that contains the control information is called a monitor control server. The following terms describe components of the Replication Alert Monitor:

- Monitor

  A monitor is one instance or occurrence of the Replication Alert Monitor. One can set up a monitor to check the status of the replication programs that are running on a single server or multiple servers. Each monitor checks the replication activity on the server, or servers, that it is assigned to.

- Monitor qualifier

  A monitor qualifier is a name specified for a monitor. Every monitor has a unique monitor qualifier.

- Monitor control server

  A monitor control server is any server containing control information for the Replication Alert Monitor.

- Alerts

  Alerts are notices that notify designated persons about events and conditions in the replication environment. The Replication Alert Monitor sends alerts via e-mail or pager.

- Alert conditions

  Alert conditions are conditions of the replication environment that cause the Replication Alert Monitor to send alerts. There are three kinds of alert conditions, as follows:

  – Alert conditions that are triggered by status

    Status alert conditions inform of the state of the replication programs. For example, by specifying the APPLY_STATUS alert condition, the Replication Alert Monitor will send an alert if an Apply program is not running.

  – Alert conditions that are triggered by events

    Event alert conditions notify when specific events in replication happen. For example, by specifying the QAPPLY_ERRORS alert condition, the

Replication Alert Monitor will send an alert any time the Q Apply program records an error in the IBMQREP_APPLYTRACE table.

– Alert conditions that are triggered by thresholds

Threshold alert conditions notify when a threshold has been exceed in the replication environment. For example, by specifying the QCAPTURE_MEMORY alert condition, the Replication Alert Monitor will notify any time the Q Capture program uses more memory than its threshold allows.

- Contacts

  A contact is an e-mail address or a pager address for a person to receive alerts from the Replication Alert Monitor.

- Contact groups

  A contact group is a collection of contacts that receive the same alerts.

The Replication Alert Monitor can send an e-mail when an alert condition occurs. The content of the e-mail notification depends on whether the e-mail address provided is for a pager.

- The e-mail that is sent to non-pager devices shows the time when each alert condition occurred at the specific server. It also shows the number of times that each alert condition occurred and the associated message.
- The e-mail that the Replication Alert Monitor sends to pagers contains a summary of the parameters that triggered the alert instead of a complete message. If an alert condition occurred many times, the timestamp reflects the last time that the alert condition occurred.

For further details on Replication Alert Monitor, refer to the *IBM DB2 Information Integrator Replication and Event Publishing Guide and Reference Version 8.2*, SC18-7568.

2.2 Q replication processing flow

Figure 2-2 describes the main steps in setting up a Q replication environment. Chapter 3, “Bidirectional Q replication on z/OS platforms” on page 73; Chapter 4, “Peer-to-peer Q replication on z/OS platforms” on page 295; Chapter 5, “Bidirectional Q replication on AIX platforms” on page 419; and Chapter 6, “Peer-to-peer Q replication on AIX platforms” on page 543, describe the implementation of bidirectional and peer-to-peer replication in z/OS and AIX environments using the step-by-step approach described in Figure 2-2.

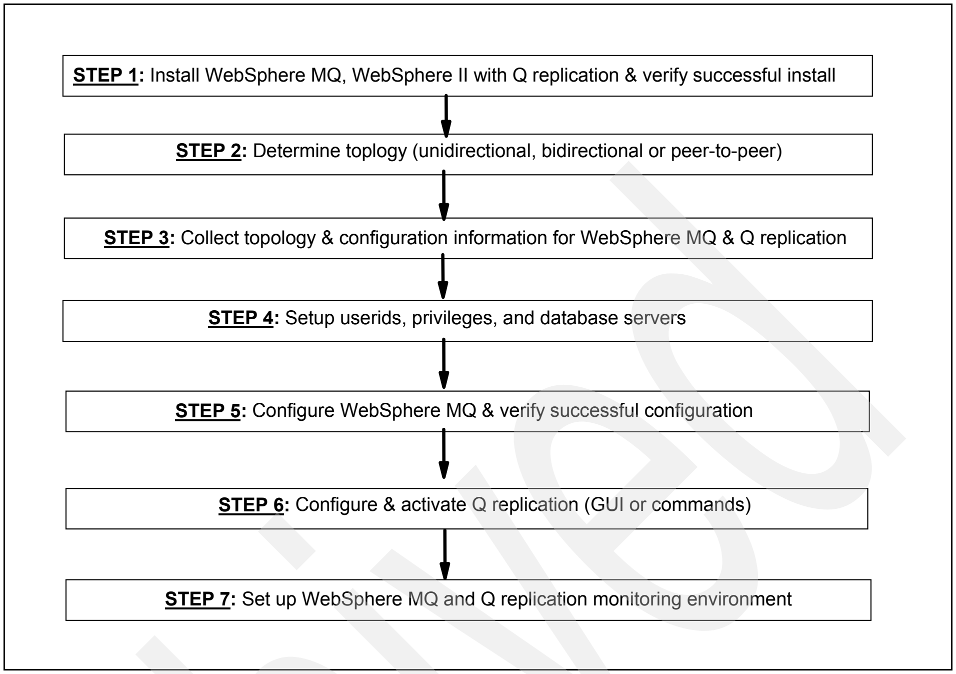


Figure 2-2 Overview of setup steps

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 2-3.

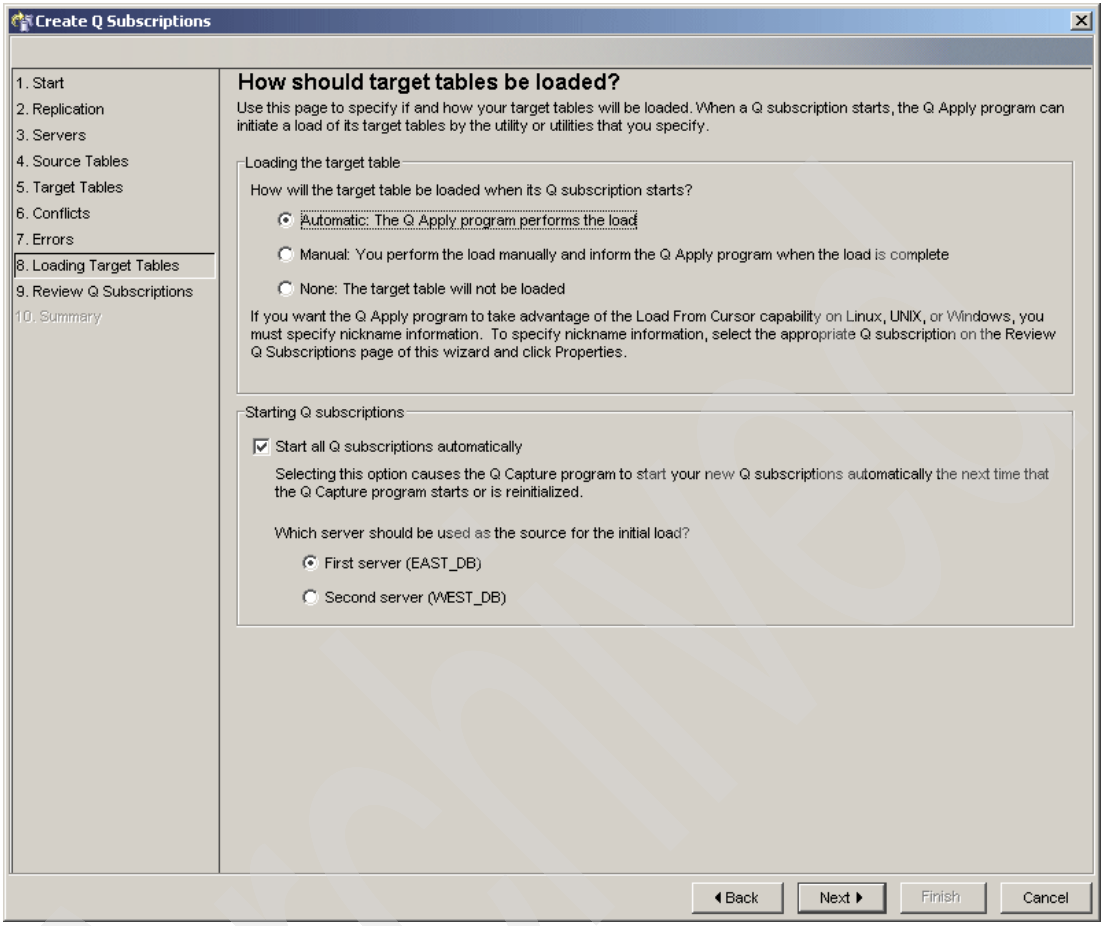


Figure 2-3 Specifying how target table is to be loaded for given Q subscription

The Replication Center and ASNCLP commands generate SQL that records the new state "N" in the STATUS column of the IBMQREP_SUBS table. When Q Capture sees the "N" state during startup or after being re-initialized, it automatically inserts a signal into the IBMQREP_SIGNAL table, as shown in Example 2-1. 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 2-4, 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
```

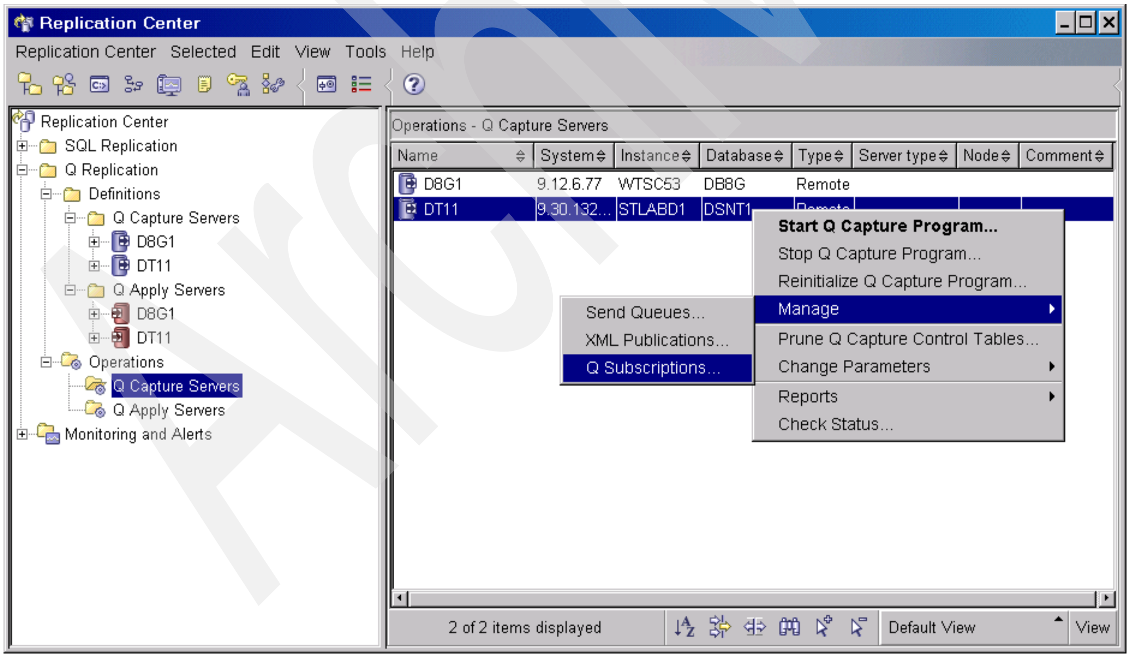


Figure 2-4 Activating Q subscription via the Replication Center

Figure 2-5 describes the four main steps involved when manual loading of the target table is configured for a subscription in a unidirectional Q replication configuration.

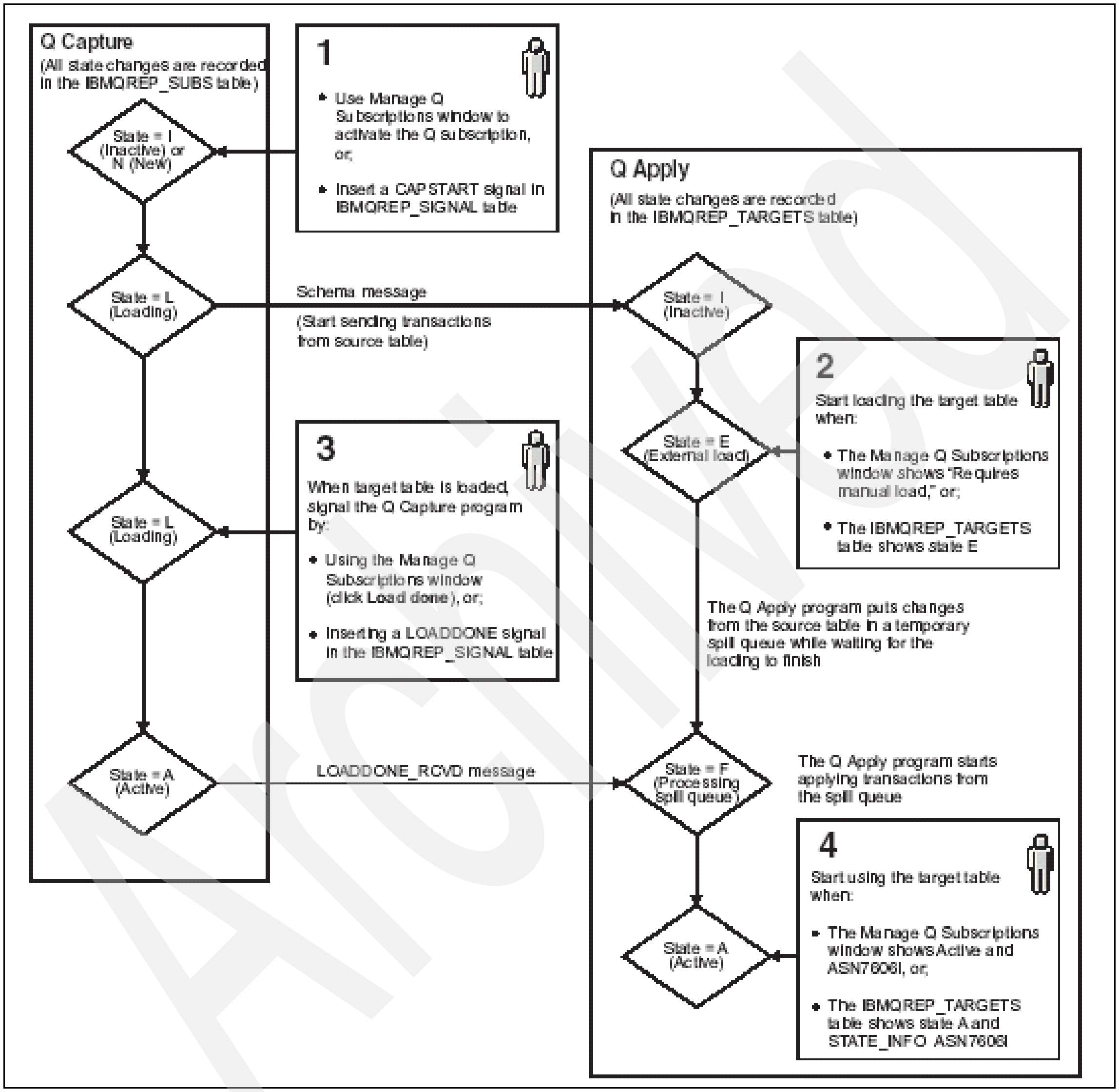


Figure 2-5 Manual load processing flow in unidirectional Q replication

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 2-5, 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. For details on the bidirectional and peer-to-peer flows, please refer to the *IBM DB2 Information Integrator Replication and Event Publishing Guide and Reference Version 8.2*, SC18-7568.

2.2.2 Ongoing replication after the initial synchronization

Figure 2-1 on page 18 describes a simple unidirectional replication configuration, highlighting the main objects involved in the process.

Figure 2-6 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.

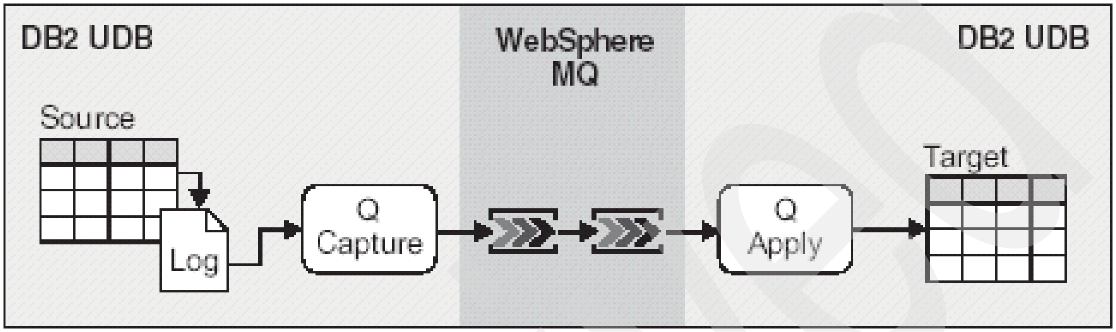


Figure 2-6 Ongoing replication flow

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 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 single thread reads the log and 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.

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. Some of the key Q Capture parameters are discussed in 2.4, “Best practices considerations” on page 40.

WebSphere MQ

WebSphere MQ ensures that every message[1] 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 messages from a Q Capture program to a Q Apply program or user application. In remote configurations, this is defined as a remote queue on the source system corresponding to the receive queue on the target system. Each send queue should be used by only one Q Capture program.
- *Receive queue* is a queue that receives data and informational messages from a Q Capture program to a Q Apply program. This is a local queue on the target system.
- *Administration queue* is a queue that receives control messages from a Q Apply program or a user application to the Q Capture program. This is a local

[1] Q replication messages are persistent.

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. The spill queue must have a specific name, “IBMQREP.SPILL.MODELQ”.

> **Note:** In fixpak 9, which is due out in April 2005, the spill queue name may be chosen by the user.

The WebSphere MQ objects must be defined to 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.

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, one 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, one must provide the name of a queue manager on the system where the Q Apply program runs.
- For the replication queue maps, one must provide the name of a send queue on the system where the Q Capture program runs, and a receive queue and administration queue on the system where the Q Apply program runs.
- For the publishing queue maps, one must provide the name of a send queue on the system where the Q Capture program runs.

Attention: In fixpak8, support is provided for a WebSphere MQ client 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 as described in “MAXMSGL” on page 58) of the messages on the queue, while the queues allow one to specify the maximum number of messages (MAXDEPTH as described in “MAXDEPTH” on page 56) 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. Some of the key settings are discussed in 2.4, “Best practices considerations” on page 40.

Q Apply

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

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

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

To allow the Q Apply program to track what changes it has already applied to the target, each target table must have some mechanism to enforce that rows are unique. This uniqueness can come from a primary key, unique constraint, or unique index at the target. One can specify whether the program applies transactions in parallel, change its operating behavior in several ways, and set options for how the Q Apply program detects and handles conflicts[2].

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 order[3] 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.

> **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

[2] 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.

[3] 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.

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. Some of the key Q Apply parameters are discussed in 2.4, “Best practices considerations” on page 40.

2.3 Choosing a particular Q replication topology

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

Table 2-1 Choosing a particular Q replication topology

Evaluation criteria	**Unidirectional**	**Bidirectional**	**Peer-to-peer**
One-way replication	Yes		
Subset rows and columns in target	Yes		
Data transformation required	Yes		
“Toggle”[a] replication between two servers		Yes	
More than two servers involved in replication			Yes
Conflict detection and resolution requirements ► “Master” wins 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 ► 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.

2.4 Best practices considerations

Q Capture, Replication queue map, Q Apply, and WebSphere MQ provide a number of parameters to optimize the operation and performance of the Q replication environment. Each of the main parameters associated with these components is discussed in the following sections. A brief description of each parameter function, and its key considerations and best practices recommendations is provided. Since latency is a critical criterion in the selection of Q replication as the technology to address business requirements, it is specifically addressed here.

2.4.1 Q Capture considerations

The Q Capture program's operating parameters govern the way that it starts, the amount of memory that it uses, which queue manager it connects to, and how frequently it commits messages to queues, amongst other things. The parameters are read from the IBMQREP_CAPPARMS table when Q Capture is started or re-initialized. One can specify temporary runtime values for some parameters when the program is started, and also while the program is running.

The main parameters of interest discussed here are COMMIT_INTERVAL, MEMORY_LIMIT, MONITOR_INTERVAL, and SLEEP_INTERVAL.

COMMIT_INTERVAL

This parameter specifies how often, in milliseconds, a Q Capture program commits transactions to WebSphere MQ. By default, a Q Capture program waits 500 milliseconds (a half second) between commits. At each interval, the Q Capture program issues an MQCMIT call. This signals the WebSphere MQ queue manager to make messages that were placed on send queues available to the Q Apply program or other user applications. All of the DB2 transactions that are grouped within an MQCMIT call are considered to be a WebSphere MQ unit of work, or transaction. Typically, each WebSphere MQ transaction contains several DB2 transactions.

The default COMMIT_INTERVAL value is 500 milliseconds. Committing messages more frequently requires more CPU usage and I/O overhead. Lowering the commit interval value is not likely to improve throughput, but will potentially improve latency. If you cannot increase the amount of memory (MEMORY_LIMIT) allocated for the Q Capture, then you can reduce the commit interval. Reducing the commit interval flushes messages from memory more often and prevents the Q Capture program from spilling transactions to disk. Raising the commit interval reduces CPU usage, which may be appropriate in a low volume environment, but can increase latency.

Note: If the DB2 transaction is large, the Q Capture program will not issue an MQCMIT call even if the commit interval is reached. The Q Capture program will commit only after the entire large DB2 transaction is put on the send queue. When the number of committed DB2 transactions that are read by a Q Capture program reaches 128, the program issues an MQCMIT call regardless of your setting for COMMIT_INTERVAL.

Performance considerations

Finding the best commit interval is a compromise between latency and CPU overhead associated with the commit process.

- To reduce latency, shorten the commit interval.

  Transactions will be pushed through with less delay. This is especially important if changes to the source database are used to trigger events. To judge whether one can reduce latency by shortening the commit interval, determine if a Q Capture program's idle time (as identified by the CAPTURE_IDLE[4] value in the IBMQREP_CAPMON control table) after processing all transactions is high. Also, if the number of transactions published per commit interval (TRANS_PUBLISHED[5] value in the IBMQREP_CAPQMON table) is high, one might want to have the Q Capture program commit fewer transactions at a time to WebSphere MQ.

- To reduce CPU overhead caused by logging of messages in WebSphere MQ, lengthen the commit interval.

  A longer commit interval lets one send a larger number of DB2 transactions for each WebSphere MQ transaction. A longer commit interval also reduces I/O that is caused by logging of messages. If one lengthens the commit interval, one might be limited by the memory (MEMORY_LIMIT) allocated for a Q Capture program, the maximum depth (number of messages MAXDEPTH) for send queues, and the queue manager's maximum uncommitted messages (MAXUMSGS) attribute. If a Q Capture program waits longer between commits, the publication of some transactions might be delayed, which could increase latency.

Best practices

We recommend starting with the default value of 500 milliseconds, and then tuning the value based on the results of monitoring and business requirements for latency.

[4] This is the time in seconds that a Q Capture program slept while waiting for new transactions to send after reaching the end of the log.
[5] The number of transactions that the Q Capture program sent via a particular send queue.

For details on tuning the COMMIT_INTERVAL parameter, please refer to the *IBM DB2 Information Integrator Tuning for Replication and Event Publishing Performance Version 8.2*, SC18-9289.

MEMORY_LIMIT

This parameter specifies the amount of memory that a Q Capture program can use to build DB2 transactions in memory. By default, a Q Capture program uses a maximum of 32 MB.

When the memory amount allocated by this parameter is used, a Q Capture program spills in-memory transactions to files (one transaction per file) that are located in the CAPTURE_PATH directory. On z/OS, the Q Capture program spills to VIO or to the file that is specified in the CAPSPILL DD card.

One can adjust the memory limit based on one's needs.

> **Note:** For z/OS, the REGION parameter of the JCL also limits memory usage by a Q Capture program. Because a Q Capture program does not know the region size, the program could reach this operating system limit before reaching the memory limit. To avoid this situation, set MEMORY_LIMIT to about half of the region size.

Performance considerations

Amongst other considerations, the throughput of the Q Capture program depends on the amount of memory available to build the transactions in memory before writing them out to the WebSphere queues. The larger the amount of memory available, the better the throughput.

- To improve the performance of a Q Capture program, increase the memory limit.

  If higher throughput is desired, maximize the memory limit whenever possible.

- To conserve system resources, lower the memory limit.

  A lower memory limit reduces competition with other system operations. However, setting the memory limit too low will use more space on your system for the spill file and prompt more I/O that can slow the system.

Best practices

The default value of 32 MB is sufficient for most workloads. This limit may need to be increased if the Q Capture program spills transactions to disk or virtual I/O.

For details on tuning the MEMORY_LIMIT parameter, please refer to the *IBM DB2 Information Integrator Tuning for Replication and Event Publishing Performance Version 8.2*, SC18-9289.

MONITOR_INTERVAL

This parameter tells a Q Capture program how often to insert performance statistics into the two following control tables:

- The IBMQREP_CAPMON table shows statistics for overall Q Capture program performance.
- The IBMQREP_CAPQMON table shows Q Capture program statistics for each send queue.

The default is 300 seconds.

> **Note:** MONITOR_INTERVAL is technically not a tuning parameter. Changing the interval does not affect performance. However, to obtain more current statistics, consider lowering the value for this parameter. The more often the monitor data is recorded, the more accurate it will be when queried.

Typically, a Q Capture program commits WebSphere MQ transactions at a much shorter interval (default COMMIT_INTERVAL is 500 milliseconds). Therefore, with the shipped defaults, each insert into the monitor tables contains totals for 600 commits[6].

Performance considerations

If the MONITOR_INTERVAL value is set too low, it will cause many rows to be inserted into the IBMQREP_CAPMON and IBMQREP_CAPQMON control tables every time the MONITOR_INTERVAL is reached, thereby potentially impacting performance. If the value is set too high, there will be less of a performance impact; however, the granularity of the information obtained may be insufficient for adequate analysis.

Best practices

We recommend adopting the default value of 300 seconds. To monitor a Q Capture program's activity at a more granular level, use a monitor interval that is closer to the commit interval. We recommend that the MONITOR_INTERVAL value be adjusted to have a lower value during monitoring and a higher value otherwise.

[6] 300 seconds divided by 500 milliseconds.

For details on tuning the MONITOR_INTERVAL parameter, please refer to the *IBM DB2 Information Integrator Tuning for Replication and Event Publishing Performance Version 8.2*, SC18-9289.

SLEEP_INTERVAL

This parameter specifies the number of milliseconds that a Q Capture program waits after reaching the end of the active log and assembling any transactions that remain in memory. By default, a Q Capture program sleeps for 5000 milliseconds (5 seconds). After this interval, the program starts reading the log again.

Performance considerations

The value chosen for this parameter can affect the latency as follows:

- Lower the sleep interval to reduce latency.

  A smaller sleep interval can improve performance by lowering latency, reducing idle time, and increasing throughput in a high-volume transaction environment.

- Increase the sleep interval to save resources.

  A larger sleep interval gives you potential CPU savings in an environment where the source database has low traffic, or where targets do not need frequent updates.

Best practices

We recommend adopting the default value of 5 seconds, and then tuning the value based on the results of monitoring and business requirements for latency.

For details on tuning the SLEEP_INTERVAL parameter, please refer to the *IBM DB2 Information Integrator Tuning for Replication and Event Publishing Performance Version 8.2*, SC18-9289.

2.4.2 Replication queue map considerations

In Q replication, a replication queue map identifies the WebSphere® MQ queues that a Q Capture program and a Q Apply program use to transport data and communicate. Each replication queue map identifies one of each of the following WebSphere MQ queues:

- Send queue

  The WebSphere MQ queue where the Q Capture program sends source data and informational messages. Messages that are sent using a send queue that is identified by a replication queue map must be in compact format, which is the message format that the Q Apply program reads.

- Receive queue

  The WebSphere MQ queue where the Q Apply program receives source transactions before applying them to the target table or passing them to a stored procedure.

- Administration queue

  The Q Apply program uses this administration queue to send control messages to the Q Capture program.

A single replication queue map may be used to transport data for one or more Q subscriptions. Also, a single Q Capture program can write data to many send queues, and a single Q Apply program can read and apply data from many receive queues.

In addition to identifying WebSphere MQ queues, a replication queue map also contains replication attributes that involve the send queue and the receive queue, as shown in Figure 2-7.

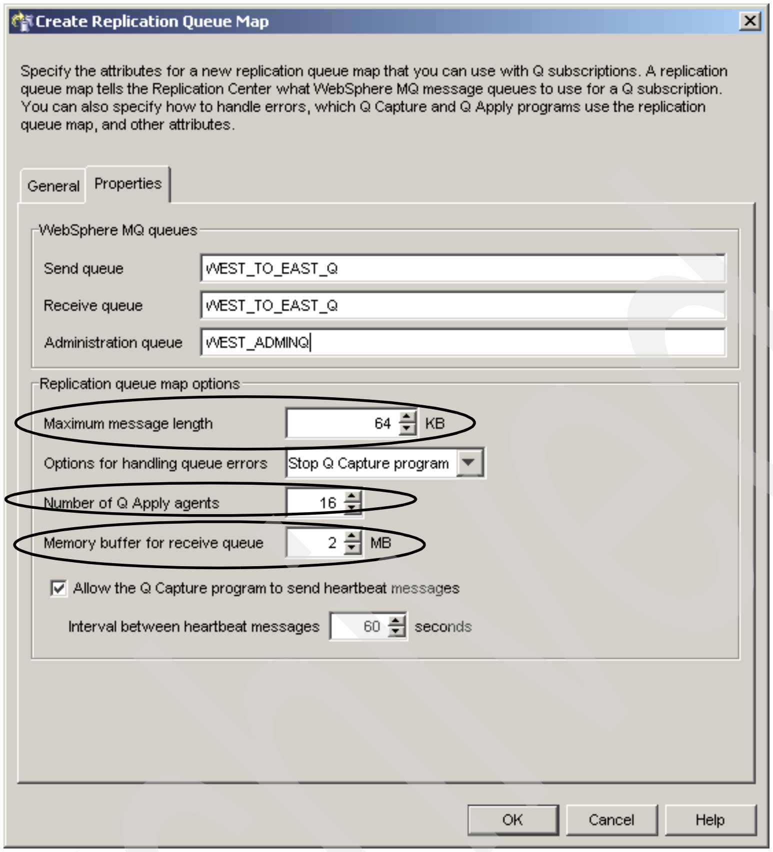


Figure 2-7 Number of Q Apply agents in replication map definition

Note: In fixpak9, which is due out in April 2005, Figure 2-7 has been changed as shown in Figure 2-8.

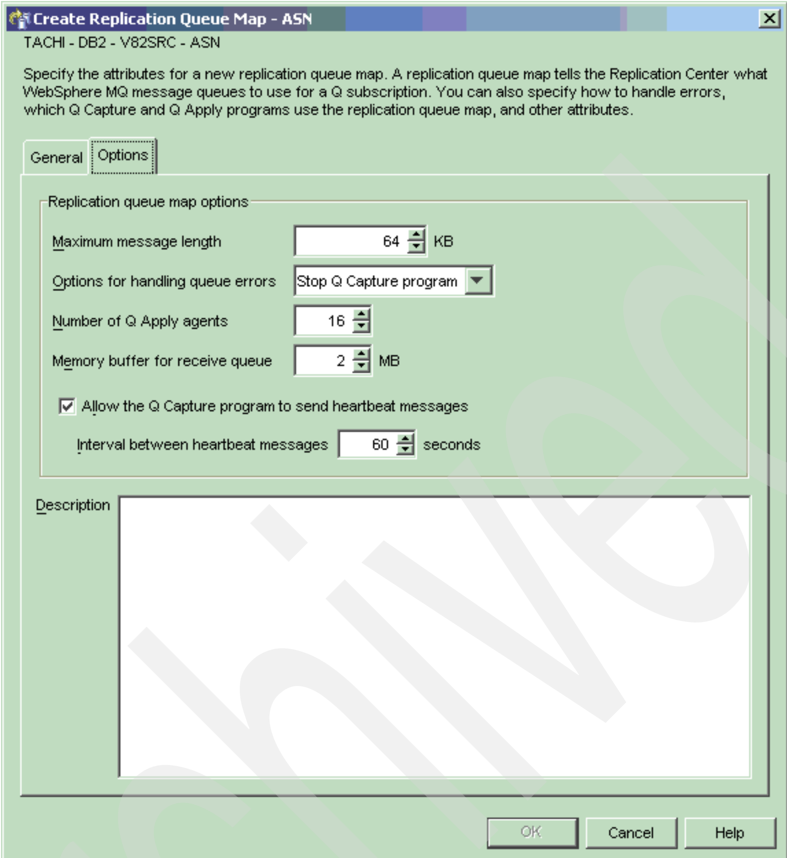


Figure 2-8 Fixpak9 screen corresponding to Figure 2-7

For each replication queue map, one specifies the following attributes that involve the send queue:

- The WebSphere MQ name of the send queue.
- The maximum size of a message (MAX_MESSAGE_SIZE) that the Q Capture program can put on this send queue. This MAX_MESSAGE_SIZE limit is independent of the WebSphere MQ Series maximum message length (MAXMSGL), but MAX_MESSAGE_SIZE must be equal to or less than the WebSphere MQ Series maximum message length (MAXMSGL).
- How the Q Capture program responds if an error occurs at the WebSphere MQ queue.
- How often the Q Capture program sends messages on this queue to tell the Q Apply program that the Q Capture program is still running when there are no changes to replicate.

The following attributes that involve the receive queue must be specified:

- The WebSphere MQ name of the receive queue.
- The number of threads (NUM_APPLY_AGENTS) for each Q Apply browser to be created to apply transactions to target tables or pass transactions to stored procedures to manipulate the data.
- The amount of memory (MEMORY_LIMIT) that the Q Apply program can use to process messages from the receive queue.
- The schema that identifies the Q Capture program that is sending messages that contain transactions.
- The name of the WebSphere MQ message queue to use for the administration queue, which is the queue where the Q Apply program sends messages when it needs to communicate with the Q Capture program.

The main parameters discussed here are MAX_MESSAGE_SIZE, MEMORY_LIMIT, and NUM_APPLY_AGENTS.

MAX_MESSAGE_SIZE

This parameter specifies the maximum size (in kilobytes) of the buffer that is used for sending messages over this send queue. The size of the buffer must not be larger than the WebSphere MQ maximum message length (MAXMSGL) attribute that is defined for any queues[7] that will contain the message, or all Q subscriptions and XML publications that use this send queue will be invalidated. The default size is 64 kilobytes.

A single message must accommodate the changes corresponding to a single row (excluding LOBs).

Performance considerations

The Q Capture program automatically divides transactions that exceed the MAX_MESSAGE_SIZE into multiple messages by breaking up the transaction at a row boundary. When replicating or publishing data from large object (LOB) columns in a source table, the Q Capture program automatically divides the LOB data into multiple messages. This ensures that the messages do not exceed the MAX_MESSAGE_SIZE that is defined for each send queue.

[7] Defined for the transmission, administration, receive, restart, and spill queues.

Attention: On some operating systems, WebSphere MQ allows one to define message segmentation so that messages that are too large for queues or channels are automatically divided. Q replication and event publishing do *not* use this message segmentation feature. If one sets up WebSphere MQ objects to use message segmentation, one must still ensure that the MAXMSGL for queues is equal to or larger than the MAX_MESSAGE_SIZE for the Q Capture program.

The relationship between MAXMSGL and MAX_MESSAGE_SIZE from a message splitting perspective is as follows. Assuming that the WebSphere MQ maximum message length (MAXMSGL) is 100 MB or higher, and the MAX_MESSAGE_SIZE is set to 20 MB, each 100 MB message will be divided into five messages, each of which is 20 MB long.

LOB values from a source table are likely to exceed the maximum amount of memory that a Q Capture program allocates as a message buffer for each send queue. In DB2® UDB, LOB values can be up to 2 gigabytes. Therefore, LOB values will frequently be divided into multiple messages. A large LOB value that is split based on a relatively small message buffer will create a very large number of LOB messages that can exceed the maximum amount of messages (MAXDEPTH as described in “MAXDEPTH” on page 56) setting for a transmission or receive queue. This will prompt a queue error. When a remote receive queue is in error, the message channel agent on the target system sends a WebSphere® MQ exception report for each message that it is unable to deliver. These exception reports can fill the Q Capture program’s administration queue.

If LOB data is to be replicated, ensure that the MAXDEPTH value for the transmission queue and administration queue on the source system, and the receive queue on the target system, is large enough to account for divided LOB messages.

Splitting a message into multiple messages can negatively impact performance, and may not be avoidable when LOBs are replicated. One can reduce the number of messages that are required to send LOB and non-LOB data by increasing the MAX_MESSAGE_SIZE for the send queue when creating a replication queue map or publishing queue map.

Best practices

Since a single message must accommodate the changes corresponding to a single row (excluding LOBs), a reasonable starting point is to determine the largest row size (excluding LOBs) of all tables of all subscriptions, and double that to arrive at a starting MAX_MESSAGE_SIZE value, and then tune it as required.

MAX_MESSAGE_SIZE should be smaller than or equal to the value of MAXMSGL. It should also be at least three times smaller than MEMORY_LIMIT (the maximum amount of memory in megabytes that the Q Apply program can use as a buffer for messages that it gets from this receive queue, as described in "MEMORY_LIMIT" on page 50); in practice, it should be orders of magnitude smaller than MEMORY_LIMIT to ensure parallel application of transactions.

The objective should be to minimize the number of messages written to the send queue, which can be addressed by choosing a sufficiently large MAX_MESSAGE_SIZE. Monitoring the MAX_TRAN_SIZE value (which excludes the LOB values) in the IBMQREP_CAPMON table can be used to optimize the MAX_MESSAGE_SIZE.

> **Important:** If one allows a larger message buffer for the Q Capture program than the queues are set up to handle, replication or publishing *cannot* occur. In most cases, the safest strategy is to use the same value for the Q Capture MAX_MESSAGE_SIZE and the MAXMSGL that one defines for the queues that will hold the messages.

For details on tuning the MAX_MESSAGE_SIZE parameter, please refer to the *IBM DB2 Information Integrator Tuning for Replication and Event Publishing Performance Version 8.2*, SC18-9289.

MEMORY_LIMIT

This parameter specifies the maximum amount of memory in megabytes that the Q Apply program can use as a buffer for messages that it gets from this receive queue until it applies them to the target. The default is 32 MB.

Most of the memory that is used by the Q Apply program is used to apply transactions to targets, and the amount of memory that is needed for that activity depends on the size of the transaction.

Performance considerations

The amount of memory that is required by the Q Apply program is proportional to the scope of the transactions and the size of the records that are updated. If transactions are very large and the Q Apply program runs out of memory, the Q Apply program will stop caching the transactions in memory but continue processing one message at a time[8]. Parallel processing will be limited, and depending on the size of the transaction that is being processed, the Q Apply program might need to process transactions serially, one after the other. Long-running transactions with infrequent commits can impact the memory and

[8] If a single transaction is large enough to exceed the memory limit, the agent threads will apply partial rows of the transaction before processing more rows of the same transaction.

storage requirements of Q replication. Consider adding more frequent commits to such transactions.

> **Important:** LOBs are *not* cached in memory. The agent will directly read the LOB value from the receive queue just prior to applying it to the target table. This approach ensures that LOBs do not stress memory requirements or cause performance degradation due to memory constraints.

Best practices

We recommend starting out with a high value for MEMORY_LIMIT, such as 100 MB, and then tuning value based on the results of monitoring and business requirements for latency.

The MEMORY_LIMIT value should be at least three times larger than MAX_MESSAGE_SIZE (preferably orders of magnitude larger) in order to ensure that parallel apply by agents is not inhibited.

For details on tuning the MEMORY_LIMIT parameter, please refer to the *IBM DB2 Information Integrator Tuning for Replication and Event Publishing Performance Version 8.2*, SC18-9289.

NUM_APPLY_AGENTS

This parameter specifies the number of agent threads that the Q Apply program uses to concurrently apply transactions from this receive queue. A value of 1 requests that transactions be executed in the order they were received from the source table; the default is 16. This parameter is associated with a specific receive queue and is part of the definition of the replication queue map, as shown in Figure 2-7 on page 46.

> **Note:** Each agent has a connection to the database, and the connections are primed when Q Apply starts.

Performance considerations

Since this parameter affects the maximum number of parallel threads that may be used by the Q Apply program in order to apply changes to the target tables from transactions originating from a particular receive queue, it can affect the Q Apply latency described in “Q Apply latency” on page 68. The larger the number of such threads, the smaller the potential Q Apply latency. However, since each agent represents a connection to the database that is primed when Q Apply is started, having a larger than necessary NUM_APPLY_AGENTS value can result in unnecessary connections to the database, and may also result in deadlocks.

In an AIX environment, the EXTSHM environment variable needs to be set as described in “Database servers” on page 434.

The tuning goal of this parameter is to use the smallest number of agents that one needs to handle the workload. If too many agents are defined, then each agent will sleep when it has no transactions to process, which could waste CPU.

> **Note:** The specification of a large number of agent threads does not necessarily cause Q Apply to apply changes in parallel, since Q Apply will only use parallel threads for transactions that do not have dependencies between them.

Best practices

We recommend starting with the default value of 16 and then monitoring and tuning for optimal performance.

- For Linux, UNIX, and Windows environments, the default value of 16 agents should provide good performance for most medium-volume to high-volume replication workloads.
- For z/OS environments, the optimal number of agents might be slightly lower than the default of 16. If increased contention is seen at the target tables that increases latency, consider one or more of the following:
  - Reduce the number of agents.
  - Use row-level locking instead of the typical default page-level locking at the target tables.
  - Reduce the deadlock timeout interval.

For details on tuning the NUM_APPLY_AGENTS parameter, please refer to the *IBM DB2 Information Integrator Tuning for Replication and Event Publishing Performance Version 8.2*, SC18-9289.

2.4.3 Q Apply considerations

The Q Apply program’s operating parameters specify which queue manager it connects to, how frequently it prunes its control tables, and whether it stops when it has no work to do, among other things. The parameters are read from the IBMQREP_APPLYPARMS table when Q Apply is started. One can specify temporary runtime values for some parameters when the program is started, and also while the program is running.

Also of interest is the NUM_APPLY_AGENTS parameter in the IBMQREP_RECVQUEUES table, which specifies the number of agent threads that the Q Apply program uses to concurrently apply transactions from the particular receive queue.

The main parameter of interest discussed here is MONITOR_INTERVAL.

MONITOR_INTERVAL

This parameter tells the Q Apply program how often to insert performance statistics into the IBMQREP_APPLYMON table. One can view these statistics by using the Q Apply Throughput and Latency windows, as shown in Figure 2-9 and Figure 2-10.

The path to Figure 2-9 is **Q replication** → **Operations** → **Q Apply Servers** in the Replication Center. Then right-click the relevant server in the content pane, and select **Reports** and **Q Apply Throughput**.

The path to Figure 2-10 is **Q replication** → **Operations** → **Q Apply Servers** in the Replication Center. Then right-click the relevant server in the content pane, and select **Reports** and **Latency**.

By default, rows are inserted into this table every 300 seconds.

> **Note:** MONITOR_INTERVAL is technically not a tuning parameter. Changing the interval does not affect performance. However, to obtain more current statistics, consider lowering the value for this parameter. The more often the monitor data is recorded, the more accurate it will be when queried.

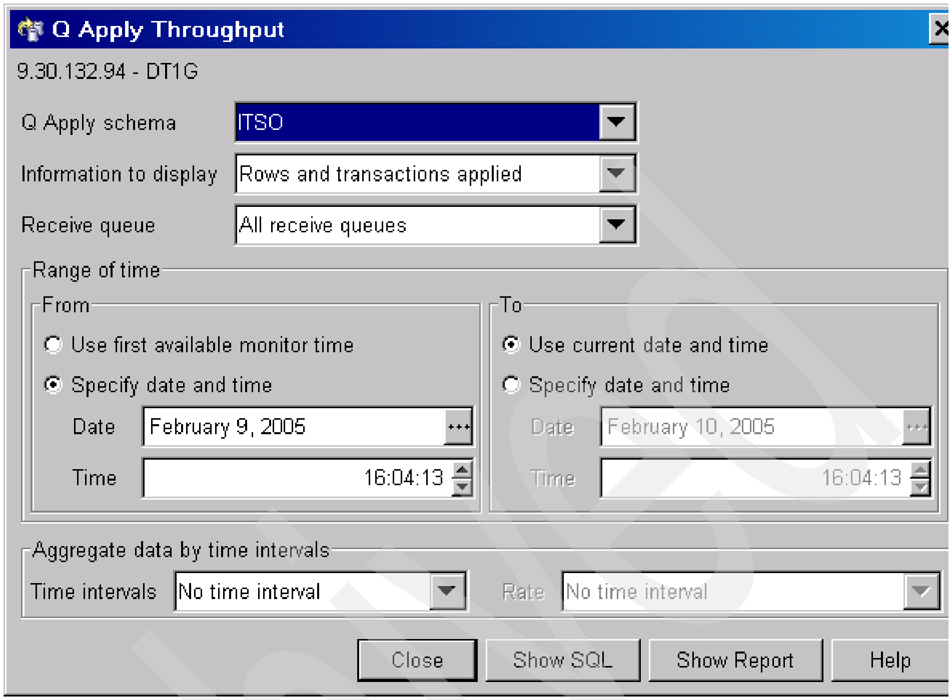


Figure 2-9 Q Apply Throughput screen

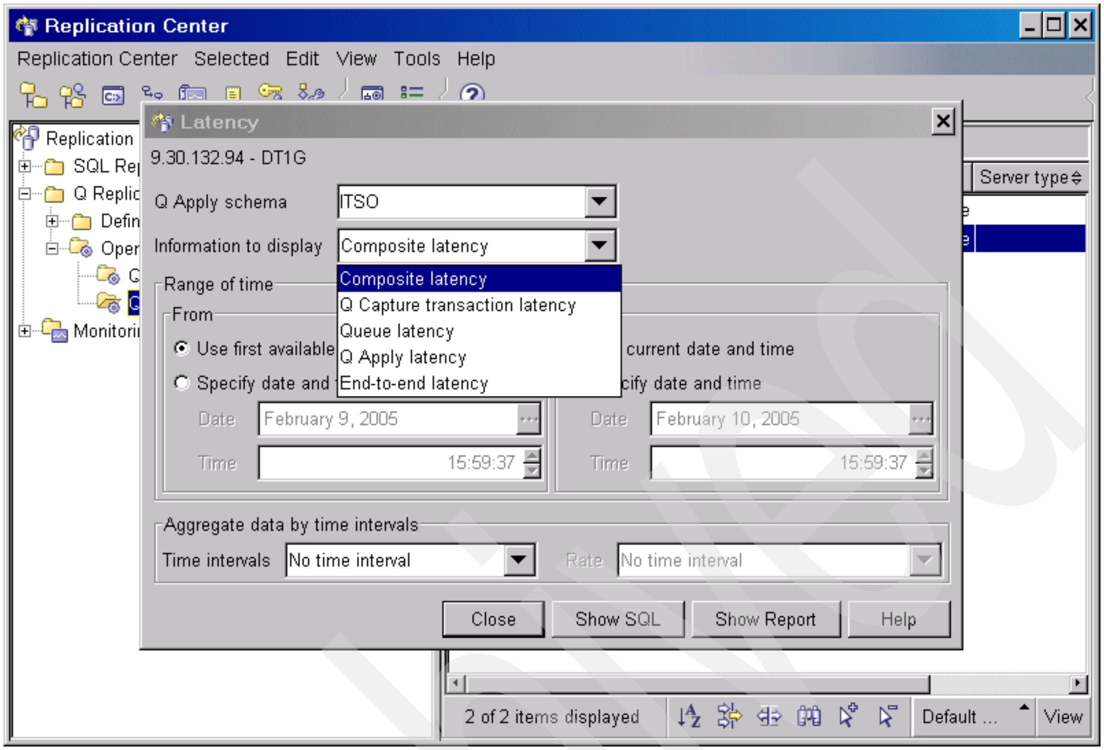


Figure 2-10 Q Apply Latency screen

Performance considerations

If the MONITOR_INTERVAL value is set too low, it will cause many rows to be inserted into the IBMQREP_APPLYMON table every time the MONITOR_INTERVAL is reached, thereby potentially impacting performance. If the value is set too high, there will be less of a performance impact; however, the granularity of the information obtained may be insufficient for adequate analysis.

This value may be set to a low value to get a very accurate current assessment of current latency at the target. IBM internal measurements have shown minimal performance impact even at intervals less than 5 seconds. Values between one and five minutes should be adequate to obtain a running history of statistics and performance.

Best practices

We recommend starting out with the default value of 300 seconds, and then adjusting the interval to suit your needs.

- To monitor a Q Apply program's activity at a more granular level, use a monitor interval that is closer to the commit interval. For example, one might want to see the statistics for the number of messages on queues broken down by each minute rather than five-minute intervals.
- Lengthen the monitor interval to view Q Apply performance statistics over longer periods. For example, if you view latency statistics for a large number of five-minute periods, you might want to average the results to get a broader view of performance. Seeing the results averaged for each half hour or hour might be more useful in one's replication environment.

For details on tuning the MONITOR_INTERVAL parameter, please refer to the *IBM DB2 Information Integrator Tuning for Replication and Event Publishing Performance Version 8.2*, SC18-9289.

2.4.4 WebSphere MQ considerations

As mentioned earlier, Q replication and event publishing use WebSphere® MQ to transmit transactional data and exchange other messages. A number of WebSphere MQ objects (as described in "WebSphere MQ" on page 34) are required by the Q Capture and Q Apply programs to operate, and they need to be configured with specific settings for a successful implementation.

In this section we focus on the parameters that may be tuned for specific environments. The key parameters are MAXDEPTH, MAXUMSGS, and MAXMSGL.

> **Note:** In z/OS environments, additional considerations such as queue indexing, queue buffer pools, WebSphere MQ logging, and WebSphere MQ channel parameters apply. These considerations are discussed briefly in this section.

MAXDEPTH

This parameter specifies the maximum number of messages in a queue, and is set for the WebSphere MQ transmission, send, receive, administration, and spill queues. WebSphere MQ stores messages on the assigned file system. Increasing MAX_DEPTH increases the storage capacity of WebSphere MQ, but this is limited to the free file system space that is available to accommodate those messages. The default value of MAX_DEPTH is 5000 in AIX and 999,999,999 in z/OS.

If one plans to use a single transmission queue for multiple remote send queues, ensure that the maximum number of messages allowed on the transmission queue is at least as large as the combined maximums for all receive queues corresponding to the remote send queues.

Performance considerations

When this limit is exceeded, a queue error is returned and the message is diverted to a dead letter queue in the case of a receive queue. If the transmission queue is full when a PUT is done to the remote SENDQ, then an MQ error of reason code 'MQRC_Q_FULL', 2053, is returned to the Q Capture program.

Best practices

If possible, choose the maximum value for this parameter to avoid queue full messages.

If not, the default value of MAXDEPTH for one's operating system might be too small for a high-rate workload, particularly for the transmission queue at the source, and the receive queue at the target.

Increase the MAXDEPTH value for the transmission queue if one plans to replicate or publish a large number of transactions, or if one expects many large transactions that will be divided into multiple messages. To determine the size of the transmission queue, consider the following factors:

- The number of source tables that one is replicating or publishing data from.
- The number and size of rows in the tables.
- The value of the Q Capture MAX_MESSAGE_SIZE parameter, which determines the way that large transactions are divided into multiple messages.
- How long one wants messages to collect on the transmission queue in the event of a network outage.
- Whether the Q Apply program will run continuously.
- Whether other applications are sharing the transmission queue.

Also, consider increasing the MAXDEPTH value for the transmission queue if the value of the COMMIT_INTERVAL parameter for the Q Capture program is longer than the default of 500 milliseconds. A longer commit interval increases the number of messages that are put on the transmission queue at the same time. This effect is heightened if messages from multiple send queues are funneled into one transmission queue. Set the MAXDEPTH value for the destination receive queue to be at least as large as the MAXDEPTH value for the transmission queue, so that the receive queue can accommodate a large volume of transactions. If the Q Apply program loads the target tables, set the MAXDEPTH value for spill queues based on the number of changes that one expects at the source while target tables are being loaded.

For details on tuning the MAXDEPTH parameter, please refer to the *IBM DB2 Information Integrator Tuning for Replication and Event Publishing Performance Version 8.2*, SC18-9289.

MAXUMSGS

This parameter specifies the maximum number of uncommitted messages for a queue manager. It limits the number of messages that can be put on queues or retrieved from queues within a synchpoint. The Q Capture program puts messages on send queues and then commits them at an interval specified in the COMMIT_INTERVAL parameter. The Q Apply program commits the messages that it deletes from the receive queue within a WebSphere MQ unit of work. The default is 10000.

Performance considerations

An insufficient value for this parameter can result in the Q Capture program stopping or Q subscriptions becoming deactivated.

When this limit is exceeded, a MQRC_SYNCPOINT_LIMIT_REACHED, 2024, message is returned to the Q Capture program, and action is taken as directed in the parameter ERROR_ACTION in the IBMQREP_SENDQUEUES table.

The COMMIT_INTERVAL parameter (described in "COMMIT_INTERVAL" on page 40) in conjunction with the volume of messages generated by Q Capture in the commit interval will affect the number of uncommitted messages written to the send queue, and may cause the MAXUMSGS limit to be exceeded.

Best practices

Ensure that the MAXUMSGS value is greater than the sum of all uncommitted messages generated by all Q Capture programs to avoid the Q Capture program from being stopped or the Q subscriptions becoming deactivated.

MAXMSGL

This WebSphere MQ parameter defines the maximum size of messages allowed on queues for this queue manager. The default is 4194304 bytes.

MAXMSGL must be at least as large as the MAX_MESSAGE_SIZE defined in the replication queue map or publishing queue map. It should also be at least as large as the MAXMSGL that is defined for each send queue, transmission queue, and the administration queue.

> **Note:** MAXMSGL is defined only at the queue definition level in z/OS™, and not at the MQ manager level (as in AIX).

Performance considerations

Ensure that the maximum size of messages for the queue is not less than the MAXMSGL defined for the receive queue on the target system, and the MAX_MESSAGE_SIZE and MEMORY_LIMIT that you set when you create a replication queue map.

If you plan to replicate LOB data, ensure that the MAXDEPTH value for the transmission queue and administration queue on the source system, and the receive queue on the target system, is large enough to account for divided LOB messages. You can reduce the number of messages that are required to send LOB data by increasing the MAX_MESSAGE_SIZE for the send queue when you create a replication queue map or publishing queue map.

Best practices

Ensure that MAXMSGL is greater than MAX_MESSAGE_SIZE, as specified in “MAX_MESSAGE_SIZE” on page 48.

z/OS considerations

This section briefly discusses WebSphere MQ tuning parameters specifically related to the z/OS environment, such as queue indexing, queue buffer pools, WebSphere MQ logging, and WebSphere MQ channel parameters.

For details on these considerations, please refer to the *IBM DB2 Information Integrator Tuning for Replication and Event Publishing Performance Version 8.2*, SC18-9289.

Queue indexing in z/OS

Queue indexing greatly improves the performance of retrieving messages that are accessed out of FIFO order, which is the case with Q replication.

The Q Apply program gets messages from the receive queue twice:

- Once in first-in, first-out order for applying to the target table
- A second time to delete the message after it has been applied

Because messages can be applied in a different order than they arrived on the queue, the Q Apply program must look for the message using its message identifier (MSGID) for the second MQGET operation at the receive queue. Creating the receive queue with INDXTYPE(MSGID) allows the queue manager to maintain an index of message identifiers that dramatically improves the speed at which the Q Apply program can prune receive queues. Otherwise, the program must search the queue sequentially. We therefore have the INDXTYPE parameter set to MSGID in the template tables (Table 3-2 on page 86 and Table 4-2 on page 302).

Queue buffer pools in z/OS

Q replication performance can be improved by creating a larger buffer pool for queues that are expected to carry a high message load. The transmission queue at the source and the receive queue at the target are likely to have the heaviest and most consistent replication or publishing message load. Creating a larger queue buffer may be particularly useful at the target system because of the number of WebSphere MQ operations[9] that are performed on the receive queue. These operations are much more efficient if the message remains in the buffer pool.

One can associate one or more local queues with a dedicated buffer pool to increase the amount of memory used to cache messages on the queue using the following procedure:

1. DEFINE BUFFPOOL with at least 50,000 4-KB pages, not the default of 1000.
2. Define a specific page set with DEFINE PSID, and set the BUFFPOOL parameter for the pageset to this buffer pool so that it will be used to cache this particular PSID.
3. Define a storage class with DEFINE STGCLASS that will map a queue to the page set by specifying a page set identifier on the STGCLASS.
4. Define the transmission queue, or the target receive queue to the above specific storage class.

One can define up to 16 buffer pools for each queue manager. The maximum number of buffers is determined by the amount of storage available in the queue manager address space. Usually, the more buffers you have, the more efficient the buffering and the better the performance of WebSphere MQ.

[9] The message is put from the channel to the queue, the browser thread gets the message from the queue, and the pruning thread gets the message a second time, deleting it from the queue and committing.

Attention: For Linux, UNIX, and Windows environments, one can use the DefaultQBufferSize parameter to increase the amount of memory that the queue manager uses to cache messages on queues up to a limit of 1 MB. One can specify this parameter in the mqs.ini file on Linux and UNIX operating systems, or in the TuningParameter stanza in the Windows registry.

If one wants a particular high-volume queue such as the source transmission queue or receive queue at the target to have a large buffer and the rest of the queues to take defaults, one can do the following:

1. Start the appropriate queue manager with a large value for DefaultQBufferSize.
2. Create the queue that you want to have a larger buffer.
3. Stop the queue manager.
4. Start the queue manager without the DefaultQBufferSize parameter (Linux and UNIX) in the mqs.ini file or the TuningParameter stanza (Windows).

The queue previously defined keeps its large value because its definition is stored on disk. Other local queues that you create will have the default buffer size. If the value of DefaultQBufferSize is too large, paging to disk can occur in some environments with smaller amounts of memory.

WebSphere MQ Logging on z/OS

Replication requires all WebSphere MQ messages to be persistent so that in the event of a forced shutdown or system failure, messages can be recovered, and data at the source and target remains in synch. The queue manager logs all persistent messages in active and archive logs in a manner similar to DB2.

The extra overhead required for message logging is an important performance consideration. Ideally, one should store the log files required by WebSphere MQ on a separate physical disk from the data files that are used for message queues, in much the same way that one should ideally separate DB2 data files and log files. This can dramatically reduce I/O contention and improve performance. If possible, use disk striping for the queue manager logs to allow for parallel I/O and to speed recovery time.

Preformatting of logs and using only primary allocations for log datasets can improve performance. Dual logging improves recovery speed following failures, and dual logs are written in parallel operations, which does not result in a performance impact.

Channel parameters in z/OS

When defining a channel, one can tune the number of messages that are committed by the queue manager between two synchpoints. This grouping is called a batch. Messages are always sent individually over the channel, but they are not committed and removed from the transmission queue until the synchpoint is reached. By increasing the size of a batch (set by the BATCHSZ parameter) from its default of 50, one can boost performance in a high-workload environment. Throughput is improved because the queue manager requires fewer commits.

In general, the heavier the workload, the higher the setting of the batch size should be. However, if the batch size is too large, there is a performance penalty of higher I/O and CPU usage caused by memory paging of uncommitted messages.

For a high-volume replication environment, set BATCHSZ between 50 and 640. Values above this are not likely to improve throughput and can consume excessive resources.

One can also set a time interval for committing batches when defining a channel. If the time interval specified by the BATCHINT parameter expires, the queue manager commits a batch of messages even if the BATCHSZ has not been reached. A batch interval can conserve CPU in a low-volume environment, but may impact response times. For replication, start by accepting the default of 0 for BATCHINT, and tune this in concert with BATCHSZ.

2.4.5 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.

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)
- Send queue and 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.

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

Q Capture latency

Q Capture latency measures the difference between a given point in time and the timestamp of the last committed transaction. 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.

Figure 2-11 shows an example of a Q Capture latency report. The path to this report is **Q replication** → **Operations** → **Q Capture Servers** in the Replication Center. Then right-click the relevant server in the content pane, and select **Reports** and **Q Capture Latency**. On the Q Capture Latency screen, set the time range and the click **Show Report**.

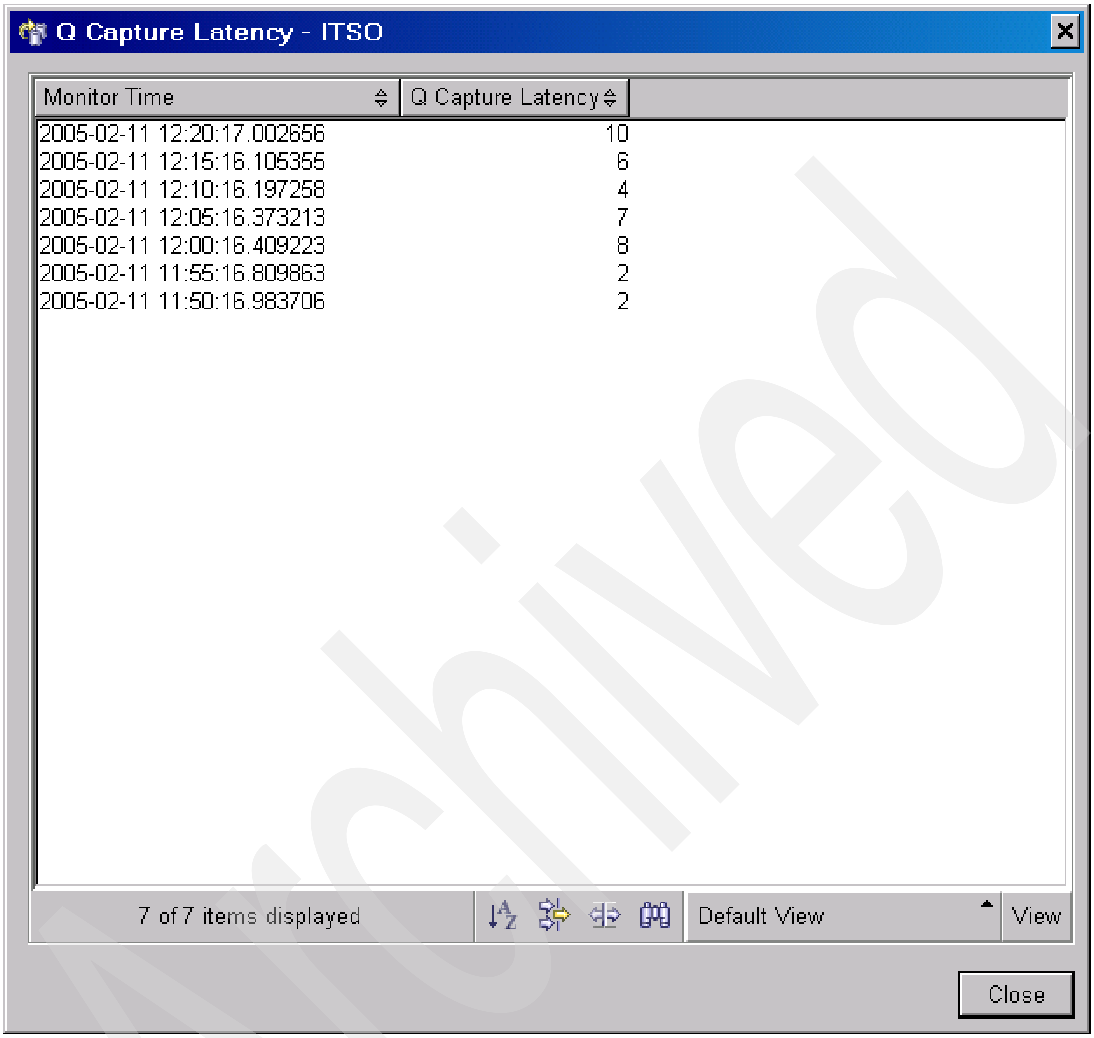


Figure 2-11 Q Capture Latency report

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 adds another worker thread to read the log.

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.

Figure 2-12 shows an example of a Q Capture transaction latency report. The path to this report is **Q replication** → **Operations** → **Q Apply Servers** in the Replication Center. Then right-click the relevant server in the content pane, and select **Reports** and **Latency**. On the Latency screen, open the drop-down on **Information to display** and select **Q Capture Latency** and right-click it. Set the time range and then click **Show Report**.

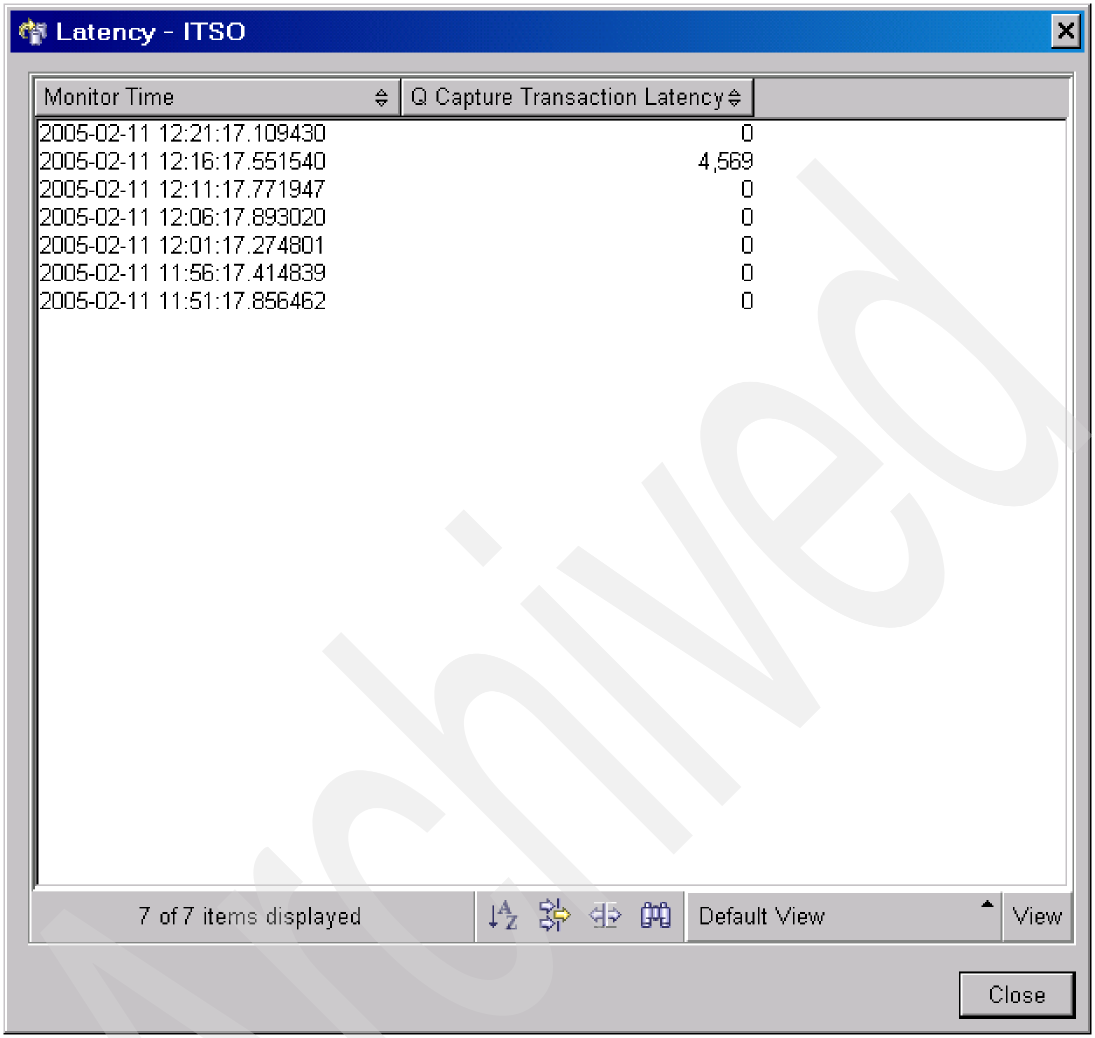


Figure 2-12 Q Capture transaction latency report

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, as described in "MEMORY_LIMIT" on page 42.
- Raising the MAX_MESSAGE_SIZE parameter, which is defined when you create a replication queue map or publication queue map, as described in 2.4.2, "Replication queue map considerations" on page 44. 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, as described in “COMMIT_INTERVAL” on page 40.

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.

Figure 2-13 shows an example of a Q latency report. The path to this report is **Q replication** → **Operations** → **Q Apply Servers** in the Replication Center. Then right-click the relevant server in the content pane, and select **Reports** and **Latency**. On the Latency screen, open the drop-down menu on the Information to display and select **Queue Latency** and right-click it. Set the time range and then click **Show Report**.

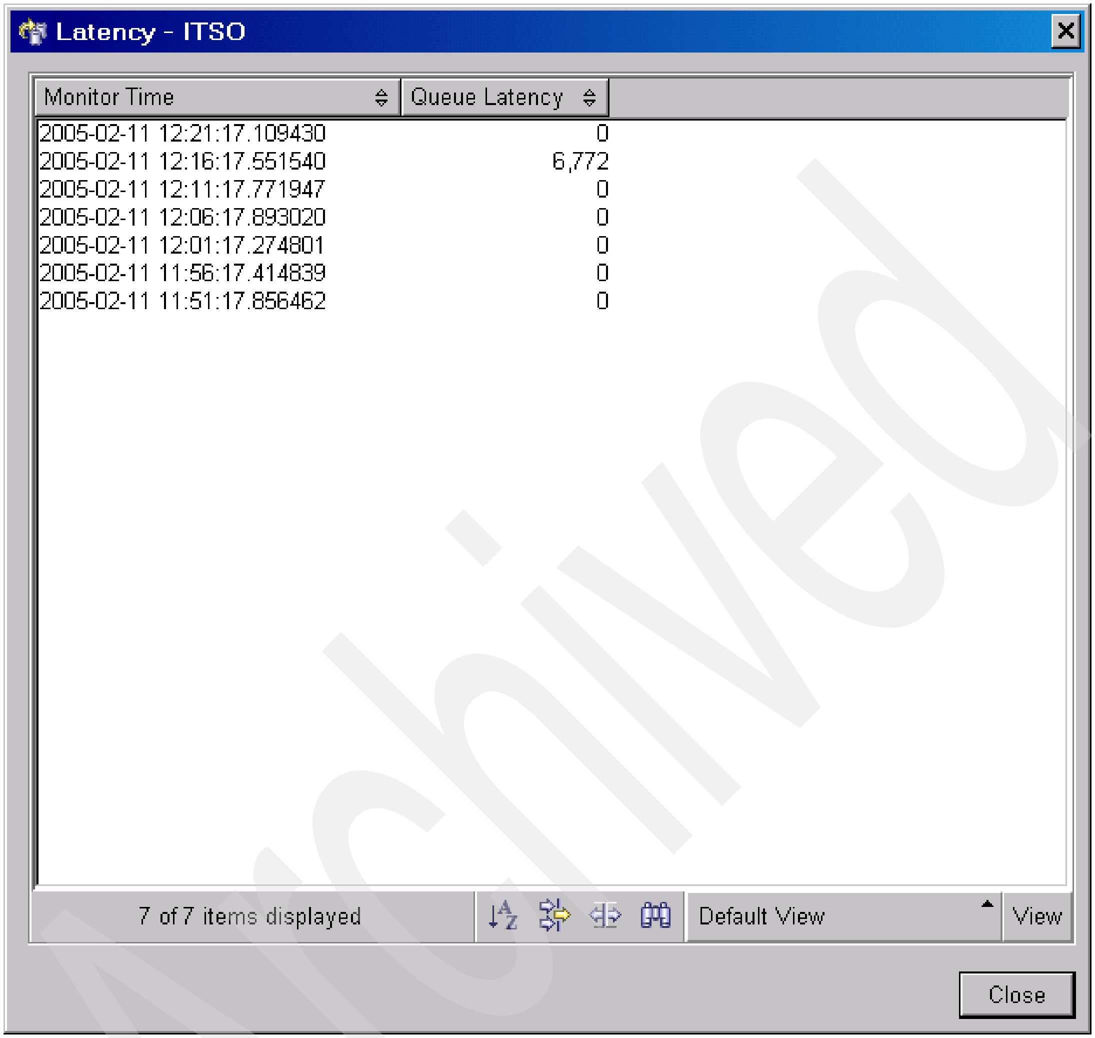


Figure 2-13 Q latency report

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

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.

Figure 2-14 shows an example of a Q Apply latency report. The path to this report is **Q replication** → **Operations** → **Q Apply Servers** in the Replication Center. Then right-click the relevant server in the content pane, and select **Reports** and **Latency**. On the Latency screen, open the drop-down menu on the Information to display and select **Q Apply Latency** and right-click it. Set the time range and then click **Show Report**.

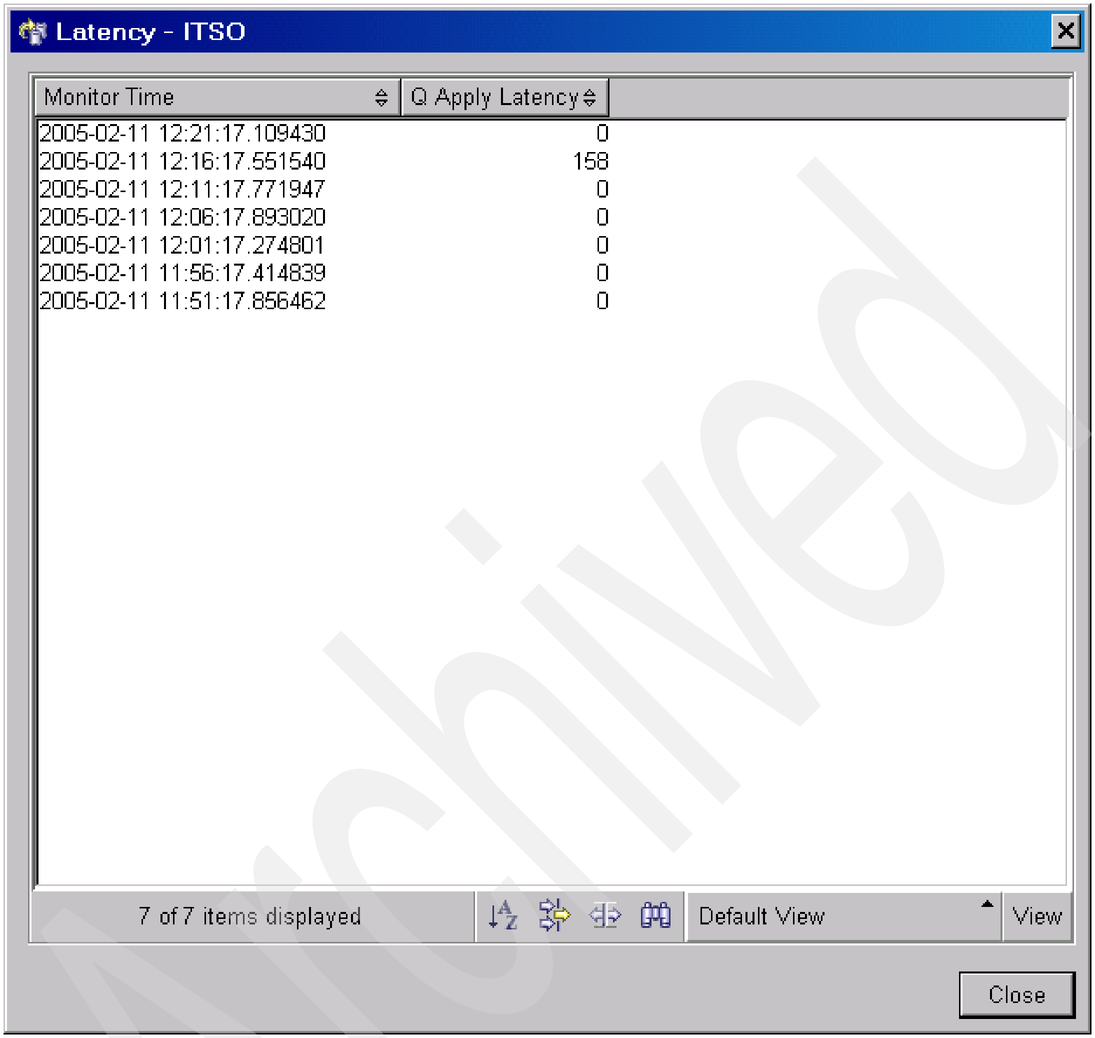


Figure 2-14 Q Apply latency report

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; refer to “NUM_APPLY_AGENTS” on page 51 for further details.
- Increasing the MEMORY_LIMIT if it is determined to be due to insufficient memory to perform parallel operations, as discussed in “MEMORY_LIMIT” on page 50.

- Tuning the database to reduce deadlocks that can contribute to increased apply latency.

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.

Figure 2-15 shows an example of a End-to-end latency report. The path to this report is **Q replication** → **Operations** → **Q Apply Servers** in the Replication Center. Then right-click the relevant server in the content pane, and select **Reports** and **Latency**. On the Latency screen, open the drop-down menu on Information to display and select **End to End Latency** and right-click it. Set the time range and then click **Show Report**.

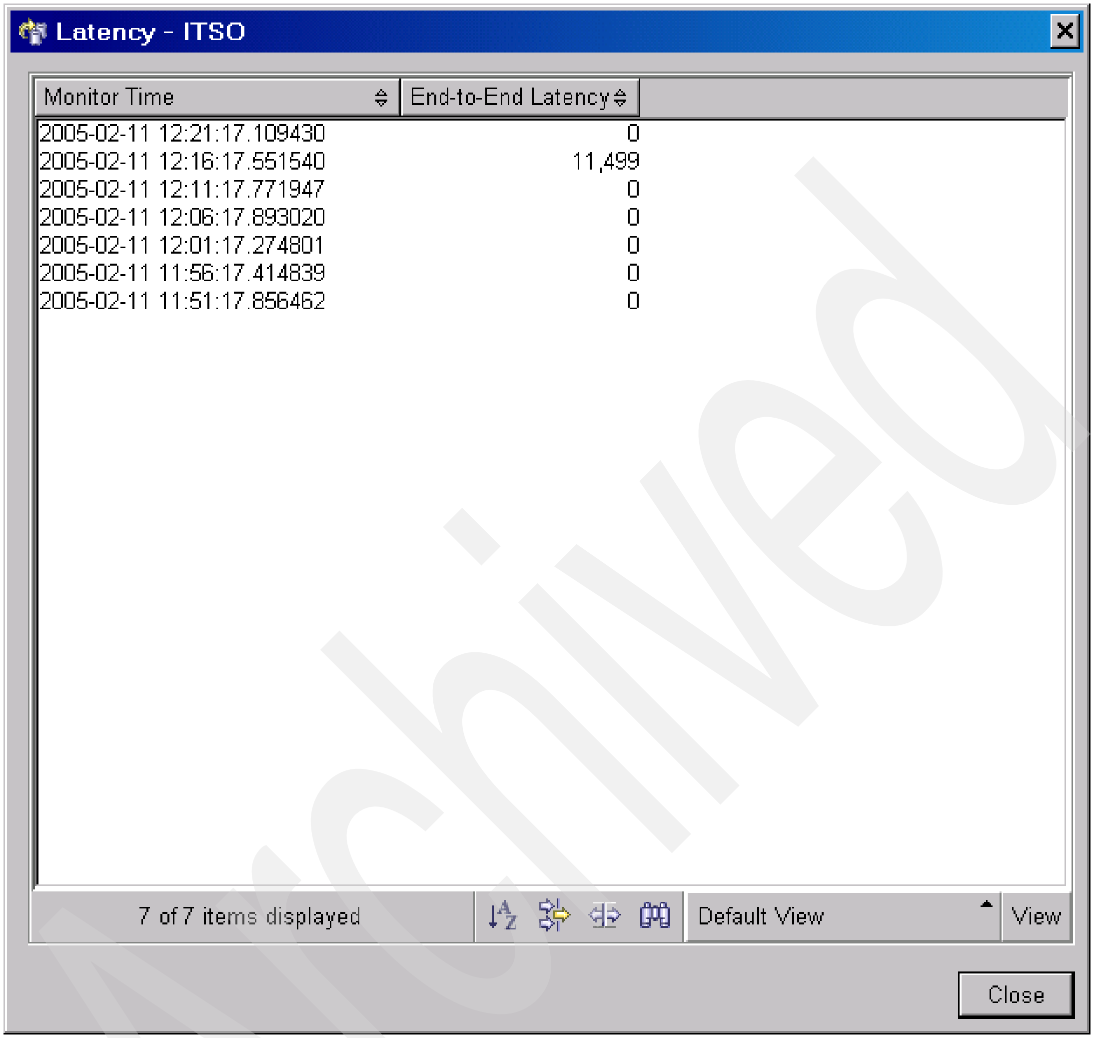


Figure 2-15 End-to-end latency report

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

3

Bidirectional Q replication on z/OS platforms

In this chapter we describe a step-by-step approach to implementing a bidirectional queue replication financial industry solution on z/OS platforms.

The topics covered are:

- Business requirements
- Rationale for the bidirectional solution
- Environment configuration
- Step-by-step setup
- Failover considerations
- Switchback considerations

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 backup 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 cycle 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 are documented as follows:

- Business requirement
- Rationale for choosing the bidirectional solution
- Environment configuration
- Step-by-step setup
- Failover considerations
- Switchback considerations

3.2 Business requirement

Our fictitious company Ithaca 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, Ithaca 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:

- 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.
- Replicated tables on the second server must be available on failover within 30 minutes.
- Replicated tables on the second server must be available for read only access while the primary server is available.
- The primary server is to be resynchronized with the secondary after it is restored to service.
- Limited resources allowed for replication-only purposes.

3.3 Rationale for the bidirectional solution

Table 2-1 on page 39 lists the evaluation criteria for choosing between unidirectional, bidirectional, and peer-to-peer replication topologies.

Since Ithaca's business requirement is support for a single backup site, the choice is between a bidirectional and a peer-to-peer replication topology.

The bidirectional replication topology is appropriate for Ithaca for the following reasons:

- Less stringent requirements for failover time (within 30 minutes instead of an instantaneous requirement, which would tilt in favor of peer-to-peer).
- Non-dedicated backup server requirement.
- Absence of conflicts[1] due to the lack of simultaneous updates—only one server is updated at a given time.
- Limited computing resources available.
- As discussed in 2.3, "Choosing a particular Q replication topology" on page 39, 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

[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 conditions are resolved partially by the conflict resolution mechanism during switchback and may require manual intervention.

alternative and does not require additional columns and triggers to be defined on existing data.

- Ithaca'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 Ithaca bidirectional replication topology.

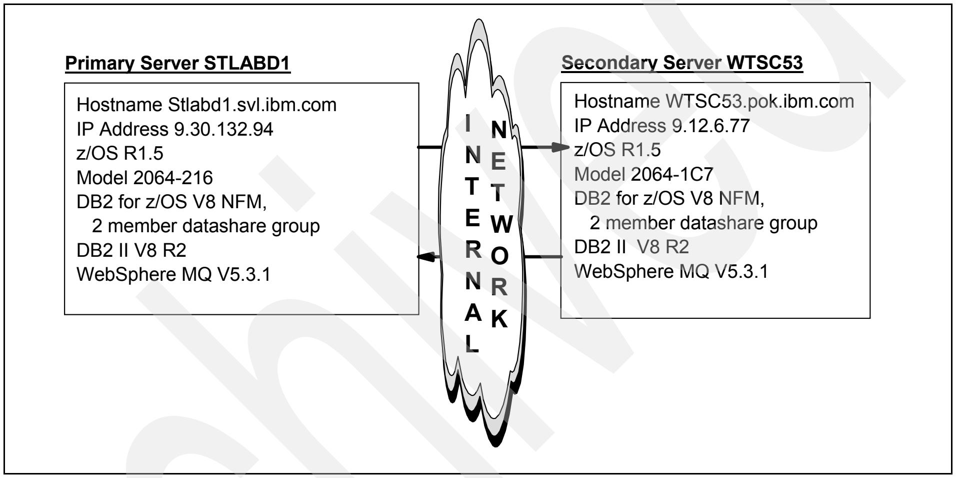


Figure 3-1 Bidirectional replication topology configuration for Ithaca

We installed a set of 23 tables with referential integrity constraints defined between some of them. Full details of these tables are documented in Appendix F, "Tables used in Q replication scenarios" on page 781.

3.5 Step-by-step setup

In this section we document the step-by-step setup of the bidirectional replication topology in our fictitious company. Figure 3-2 lists the main steps involved in setting up the environment. Each of these steps is described in detail in the following subsections.

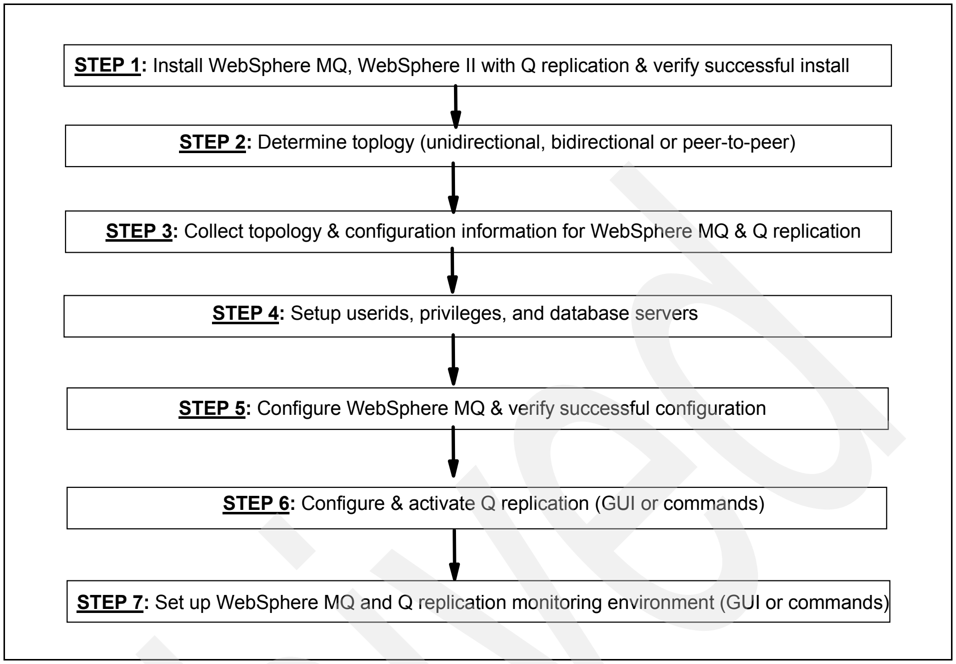


Figure 3-2 Overview of setup steps

3.5.1 Step 1: Install WebSphere MQ, WebSphere II with Q replication

The following documentation provides details on installing WebSphere MQ and WebSphere II Replication.

- *IBM Program Directory for WebSphere MQ for z/OS Version 5 Release 3, Modification Level 1,* GI10-2548-03, for details on prerequisites and installation; and *IBM WebSphere MQ for z/OS System Setup Guide V5.3.1,* SC34-6052-02, to customize WebSphere MQ after it is installed.
- *IBM DB2 Information Integrator Replication Installation and Customization Guide for z/OS Version 8.2,* SC18-9127-01, for details on prerequisites and steps for installing WebSphere II with Q replication.

We implemented the following configuration for our scenario:

1. Authorize WebSphere II programs to run as UNIX System Services (USS) as follows:
   a. Define RACF® group to contain any user IDs that will run WebSphere II for z/OS, as shown in Example 3-1.

Example 3-1 RACF group

```
ADDUSER DB2PROD DFLTGRP(SYS1) OMVS(UID(3214) HOME(/u/db2prod) PROGRAM(/bin/sh))
+  NAME('QREP STARTED TASKS')  NOPASSWORD NOOIDCARD SETROPTS CLASSACT(STARTED)
SETROPTS RACLIST(STARTED) GENERIC(STARTED)
RDEFINE STARTED QREP*.* STDATA(USER(DB2PROD) GROUP(SYS1) TRACE(YES))
RDEFINE STARTED QCAP*.* STDATA(USER(DB2PROD) GROUP(SYS1) TRACE(YES))
SETROPTS RACLIST(STARTED) GENERIC(STARTED) REFRESH
```

b. Grant any user ID that runs Q replication programs read-access to the HFS install directory and its subdirectories.

c. Grant any user ID that runs Q replication programs write-access to the /tmp directory or the directory that the TMPDIR environment variable specifies:

```
chmod -R a+r .....................
```

2. Define the WebSphere Information Integration environment to Unix System Services (USS). Although we do not plan to execute the capture or apply programs from USS or the Replication Center, this definition is required to run the command programs to modify parameters via the BPXBATCH job.

   a. A USS profile must be defined for any user ID running BPXBATCH.

   b. The time zone specified for an East Coast environment is TZ=EST5EDT.

   c. The language for messages is LANG=EN_US.

   d. The temporary directory for output information is TMPDIR=/tmp.

   e. The PATH environment variable points to a list of directories that the system searches to find executable programs:

```
PATH=$PATH:$HOME:
```

   f. The STEPLIB environment variable points to the data set that contains executable load modules:

```
STEPLIB=ASN.V8R2M0.SASNLOAD:DB8G8.SDSNLOAD
```

   g. The TMPDIR variable must be the same for user IDs that start a program and issue commands to that program:

```
TMPDIR=/tmp
```

   Since we intend to start the capture and apply programs via MVS Started Task procedures, the HFS message catalog path must be specified in the JCL, including the language to be used for messages and the USS path to locate the messages.

   If SMPE is used to install the product, the MSGS HFS file will be placed into the following path. This will become the MSGS DD statement for capture and apply, as follows:

```
//MSGS     DD PATH='/usr/lpp//db2repl_08_02/msg/En_US/db2asn.cat'
```

Example 3-2 shows an example of a profiles member.

Example 3-2 Profiles member example

```
# This is a sample .profile defining login environment parameters for
# an individual user. This should be copied to the user's $HOME/.profile
# and modified to their individual taste. More information may be
# found in the MVS/ESA OpenEdition User's Guide, in the chapter on
# customizing the shell.
#
# One may add or remove a # to disable or enable the settings below.

# The following variable may be set to point to a login script.
# In this case it is set to a file called .set up in the home
# directory. Refer to the User's Guide for more information on how to
# use this variable.

# ENV=$HOME/.setup
# export ENV

# This line appends your home directory to the current path.
PATH=$PATH:$HOME:

# This line sets the default editor to ed.
EDITOR=ed

# This line sets the prompt to display your login name, and current
# directory.
# added nice PS1 to know where you are, who you are, and what your
# current directory is
export HOSTNAME=$(uname -n)
export PS1='$LOGNAME @ $HOSTNAME:$PWD>'

# This line exports the variable settings so that they are known to the
# system.
export PATH EDITOR

# Set Timezone
TZ=EST5EDT

#Run all SPAWN'd processes in the same address space
#export _BPX_SHAREAS="YES"
#
# QREPL
#export LANG=EN_US
#export NLSPATH=$NLSPATH:/usr/lpp/db2repl_08_02/msg/%L/%N
#export TMPDIR=/tmp
#export PATH=$PATH:/usr/lpp/db2repl_08_02/bin
#export DBRMLIB=ASN.V8R2M0.SASNDBRM
```

```
#export SYSPROC=DB8G8.SDSNCLST
#export STEPLIB=ASN.V8R2M0.SASNLOAD:DB8G8.SDSNLOAD

# QREPL temporary
export LANG=EN_US
export TMPDIR=/tmp
#export PATH=$PATH:/u/qrepadm
export NLSPATH=$NLSPATH:/u/nalur/db2asn.cat
#export DBRMLIB=ASN.V8R2M0.SASNDBRM
#export SYSPROC=DB8G8.SDSNCLST
export STEPLIB=ASN.V8R2M0.SASNLOAD:DB8G8.SDSNLOAD
echo PATH is $PATH
echo STEPLIB is $STEPLIB
#
```

3. The Data Administration Server (DAS) is part of the Management Clients Package for DB2 UDB on z/OS. This product is only required if the Replication Center on the workstation is used to operate the capture, apply, or monitor programs. Since most z/OS environments are likely to operate these programs as an MVS started task, we did *not* install DAS.

   > **Note:** Refer to the *IBM Program Directory for DB2 UDB for z/OS DB2 Management Clients Package*, GI10-8567-02, for installation details.

4. z/OS Enablement (Version 8) or 390 Enablement (Version 7) is also part of the Management Clients Package for DB2 UDB on z/OS. It allows z/OS DB2 objects to be viewed from the Replication Center, and Configuration SQL scripts to be generated and executed from the workstation-based Replication Center. We installed z/OS Enablement V8 (FMID JDB881D).

   > **Note:** The workstation that uses the Replication Center to view and generate Q replication configuration scripts must have DB2 Connect™ Personal Edition (or some product that contains it) and WebSphere II Version 8.2 installed.

5. Each DB2 subsystem that must be accessed from the Replication Center must be configured as follows:

   a. Enter a DB2 command prompt:

      `DB2CMD`

   b. `db2 catalog tcpip node <node_name> remote <tcpip_addr> server <db2_tcpport> OSTYPE MVS`

   c. `db2 catalog database <db2_alias_name> at node <node_name> authentication dcs`

d. `db2 catalog dcs db <db2_alias_name> as <db2_location_name>`

Example 3-3 shows a script for configuring the two DB2 subsystems used in this scenario. Information for the variables used in Example 3-3 can be found in Table 3-1 on page 85.

Example 3-3 Script for configuring four DB2 subsystem in the Replication Center

```
db2 catalog tcpip node WTSC53 remote 9.12.6.77 server 38060
db2 catalog db D8G1 at node WTSC53 authentication dcs
db2 catalog dcs db D8G1 as DB8G
db2 catalog tcpip node STLABD1 remote 9.30.132.94 server 8010
db2 catalog db DT11 at node STLABD1 authentication dcs
db2 catalog dcs db DT11 as DSNT1
```

Verify the connection to the database by issuing:

```
db2 connect to <db2_alias_name user <userid> using <password>
```

6. An alternative to using the Replication Center to generate and submit configuration SQL scripts to the z/OS platform is to use the replication command line processor ASNCLP. While the ASNCLP *cannot* run natively in the z/OS environment, it can generate replication definitions for any environment, which can then be ported to a z/OS system for execution.

   The following steps set up the workstation to run ASNCLP to generate the SQL scripts. A similar process can be used to execute ASNCLP from a UNIX system.

   a. Set up the Java™ runtime environment in the machine's environment path as follows:

      Right-click **My Computer**, select **Properties**, and click the **Advanced** tab. Open Environment Variables, and highlight PATH, as shown in Figure 3-3.

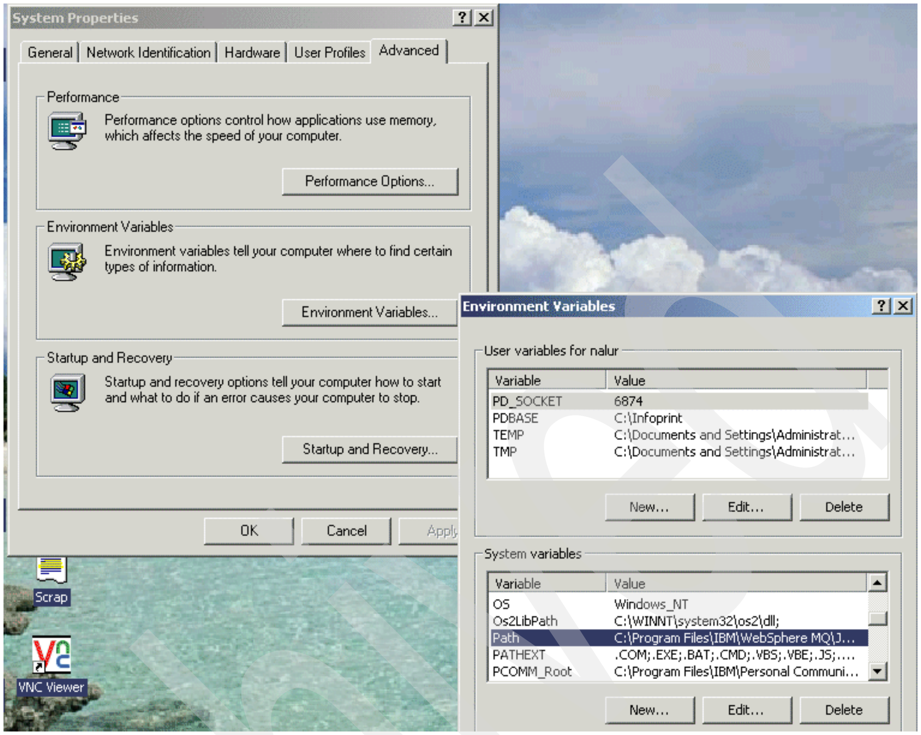


Figure 3-3 Edit PATH environment variable

Edit it and add the following to anything that is already there:

```
;C:\PROGRA~1\ibm\sqllib\java\jdk\bin
```

Or identify the location of any existing java.exe program.

b. Execute the script shown in Example 3-4 to add the following jar files to the CLASSPATH environment variable.

Example 3-4 Configure CLASSPATH environment variable

```
set CP=%CP%;c:\progra~1\ibm\sqllib\java\Common.jar;
set CP=%CP%;c:\progra~1\ibm\sqllib\tools\db2cmn.jar;
set CP=%CP%;c:\progra~1\ibm\sqllib\tools\db2replapis.jar;
set CP=%CP%;c:\progra~1\ibm\sqllib\tools\db2qreplapis.jar;
set CP=%CP%;c:\progra~1\ibm\sqllib\tools\jt400.jar;
set CLASSPATH=%CLASSPATH%;%CP%;
```

The CLASSPATH may also be found in the same manner as the Java runtime PATH variable.

c. ASNCLP can run against any host database that is cataloged on the workstation. The following bind jobs must be executed on the host after connecting to it:

   i. Change the directory to the location of the SQLLIB\BND directory created during DB2 ESE installation.

   ii. Type `DB2CMD` to get to the DB2 command line processor prompt.

   iii. Connect to the host db2 by issuing the following command:

```
db2 connect to <db2_alias_name> user <userid> using <password>
```

   iv. Issue the following bind commands:

```
db2 bind @ddcsmvs.lst isolation ur blocking all
db2 bind @db2cli.lst isolation ur blocking all
```

d. After these objects have been bound, ASNCLP may be accessed by typing `ASNCLP` from a command prompt.

   Details on using ASNCLP to generate configuration scripts and modifications can be found in *ASNCLP Program Reference for Replication and Event Publishing,* SC18-9410-00.

3.5.2 Step 2: Determine topology

We chose the bidirectional replication topology to address the Ithaca business requirement as described in 3.3, “Rationale for the bidirectional solution” on page 75.

3.5.3 Step 3: Collect topology and configuration information

Implementing bidirectional replication is a complex task involving effective coordination of the configuration settings of the operating system, database management system, WebSphere MQ and WebSphere Information Integrator Q replication offerings.

> **Important:** To ensure successful and error-free implementation, we strongly encourage systematic and careful planning that involves identifying and gathering all required information prior to commencing implementation.

Towards this end, we have developed a template that identifies all the information required to implement a bidirectional replication topology, and the cross-relationships between the information elements to ensure a smooth implementation.

Figure 3-4 provides a high-level overview of the various objects involved in implementing a bidirectional replication topology, and serves as a reference for the host and DB2 system information template (Table 3-1 on page 85), WebSphere MQ configuration information template (Table 3-2 on page 86), Q replication configuration information template (Table 3-3 on page 88), and Replication Alert Monitor configuration information template (Table 3-4 on page 89). Each of the information elements for each server is associated with a reference identification such as “A.7a” or “B.7b”, where “A” represents one of the servers and “B” the other server. These reference IDs are then cross-referenced in the template itself as well as the WebSphere II Q replication configuration screens and scripts. For example, in Table 3-2 on page 86 on WebSphere MQ information, for the XMITQ parameter (reference ID A.18a for Server A) in the SendQ, we require reference ID “A.13” (which is the name of the TransmitQ), and that value is “MQZ1XMITQ”.

Attention: In Figure 3-4, there appear to be two sets of transmission queues, and sender and receiver channels on each server. However, there is only set set on each server, as can be deduced from the identical names. Figure 3-4 has the appearance of two sets so that the flow of data and messages between the two servers is easily understood.

Once all the information identified in the template has been collected, we can proceed with the actual implementation.

Note: The template shows certain parameter values that are required for Q replication (such as parameter DEFPSIST *must* be set to YES for the TransmitQ [reference ID A.13] and does *not* have a reference ID), while others can be customized for a particular environment (such as reference ID A.13d parameter MAXDEPTH *may* be set to any value).

We have collected all the identified information for the Ithaca bidirectional replication implemented and recorded it in the templates Table 3-1 on page 85, Table 3-2 on page 86, Table 3-3 on page 88, and Table 3-4 on page 89. We are now ready to proceed with configuring the various resources.

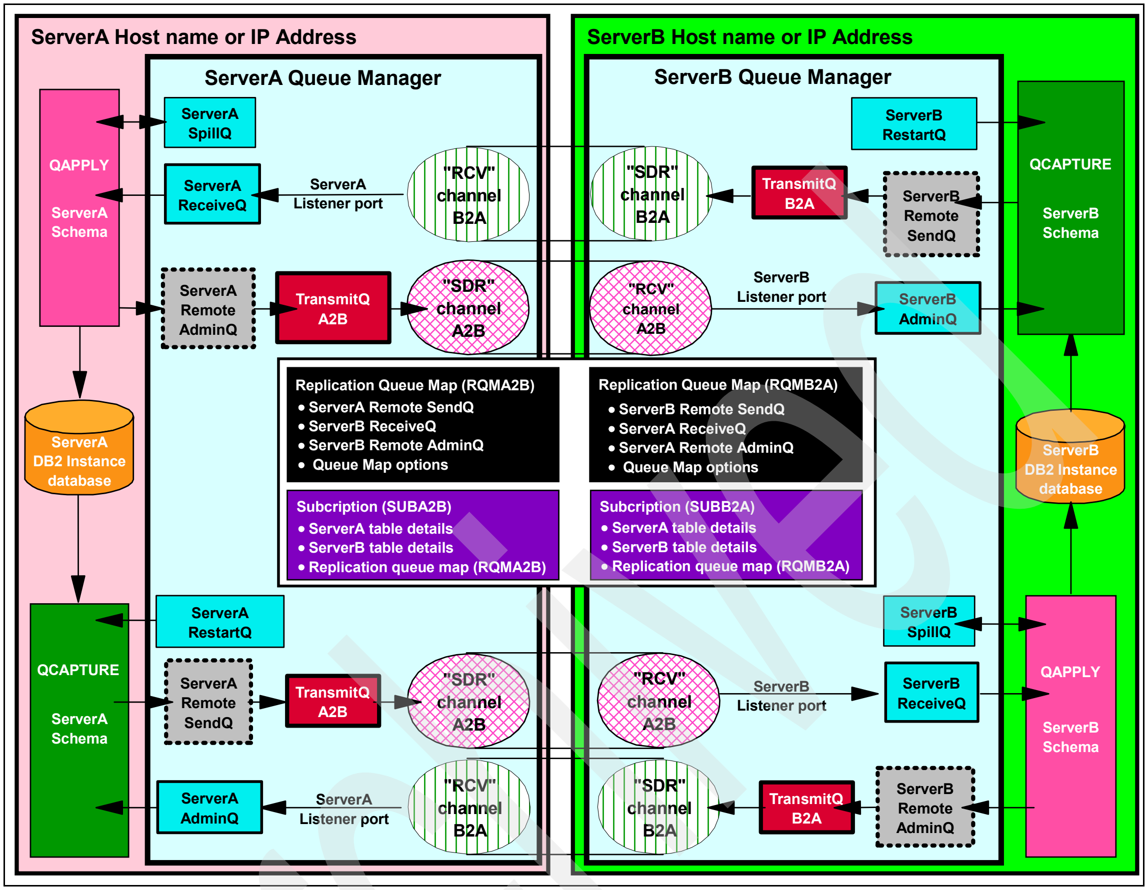


Figure 3-4 Bidirectional replication topology objects overview

Table 3-1 Host and DB2 system information

Description	Server A	Server B
Host name	(A.1) STLABD1	(B.1) WTSC53
Host IP address	(A.2) 9.30.132.94	(B.2) 9.12.6.77
DB2 subsystem name or instance owner	(A.3) DT11	(B.3) D8G1
DB2 group name (z/OS only)	(A.4) DT1G	(B.4) D8GG
DB2 location name (z/OS only)	(A.5) DSNT1	(B.5) DB8G
DB2 tcp port	(A.6) 8010	(B.6) 38060

Description	Server A	Server B
Database server/alias information ► Database server/alias ► Q Capture user ID and password ► Q Apply user ID and password	 ► (A.7a) DT11 ► (A.7b) DB2PROD/xxx ► (A.7c) DB2PROD/xxx	 ► (B.7a) D8G1 ► (B.7b) DB2PROD/xxx ► (B.7c) DB2PROD/xxx
User ID group (Unix only)	(A.8)	(B.8)
Other configuration user IDs requiring access	(A.9) QREPADM/xxx	(B.9) QREPADM/xxx
Logical database for control tables (z/OS only—must preexist) ► STOGROUP ► Volumes ► VCAT	(A.10) QREPITSO ► (A.10a) ITSOQREP ► (A.10b) TOTDDM ► (A.10b) DB8GU	(B.10) QREPITSO ► (B.10a) ITSOQREP ► (B.10b) TOTDDM ► (B.10b) DB8GU

Table 3-2 WebSphere MQ information

Description	Server A	Server B
Queue Manager ► DEADQ (optional) ► MAXUMSGS (or use default) ► MAXMSGL (or use default)	(A.11) MQS1 ► (A.11a) ► (A.11b) 10000 ► (A.11c) 4194304	(B.11) MQZ1 ► (B.11a) ► (B.11b) 10000 ► (B.11c) 4194304
Listener port	(A.12) 1414	(B.12) 1540
TransmitQ ► USAGE ► PUT ► GET ► DEFPSIST ► TRIGGER ► TRIGTYPE ► TRIGDATA ► INITQ ► MAXMSGL (or use default) ► MAXDEPTH (or use default)	(A.13) MQZ1XMITQ ► XMITQ ► ENABLED ► ENABLED ► YES ► YES ► FIRST ► (A.13a)=(A14) MQS1.TO.MQZ1 ► (A.13b) SYSTEM.CHANNEL.INITQ ► (A.13c) 4194304 ► (A.13d) 999,999,999	(B.13) MQS1XMITQ ► XMITQ ► ENABLED ► ENABLED ► YES ► YES ► FIRST ► (B.13a)=(B14) MQZ1.TO.MQS1 ► (B.13b) SYSTEM.CHANNEL.INITQ ► (B.13c) 4194304 ► (B.13d) 999,999,999
SDR channel ► CHLTYPE ► TRPTYPE ► XMITQ ► CONNAME ► HBINT (or use default)	(A.14) MQS1.TO.MQZ1 ► SDR ► (A.14a) TCP ► (A.14b)=(A.13) MQZ1XMITQ ► (A.14c) 9.12.6.77 (1540) ► (A.14d) 300	(B.14) MQZ1.TO.MQS1 ► SDR ► (B.14a) TCP ► (B.14b)=(B.13) MQS1XMITQ ► (B.14c) 9.30.132.94 (1414) ► (B.14d) 300

Description	Server A	Server B
RCV channel ▸ CHLTYPE ▸ TRPTYPE ▸ HBINT (or use default)	(A.15)=(B.14) MQZ1.TO.MQS1 ▸ RCVR ▸ (A.15a) TCP ▸ (A.15b) 300	(B.15)=(A.14) MQS1.TO.MQZ1 ▸ RCVR ▸ (B.15a) TCP ▸ (B.15b) 300
RestartQ ▸ PUT ▸ GET ▸ SHARE ▸ DEFSOPT ▸ DEFPSIST ▸ MAXMSGL (or use default) ▸ MAXDEPTH (or use default)	(A.16) QREP.STLA.RESTARTQ ▸ ENABLED ▸ ENABLED ▸ YES ▸ SHARED ▸ YES ▸ (A.16a) 4194304 ▸ (A.16b) 999,999,999	(B.16) QREP.POKA.RESTARTQ ▸ ENABLED ▸ ENABLED ▸ YES ▸ SHARED ▸ YES ▸ (B.16a) 4194304 ▸ (B.16b) 999,999,999
AdminQ ▸ PUT ▸ GET ▸ SHARE ▸ DEFSOPT ▸ DEFPSIST ▸ MAXMSGL (or use default) ▸ MAXDEPTH (or use default)	(A.17) QREP.STLA.ADMINQ ▸ ENABLED ▸ ENABLED ▸ YES ▸ SHARED ▸ YES ▸ (A.17a) 4194304 ▸ (A.17b) 999,999,999	(B.17) QREP.POKA.ADMINQ ▸ ENABLED ▸ ENABLED ▸ YES ▸ SHARED ▸ YES ▸ (B.17a) 4194304 ▸ (B.17b) 999,999,999
SendQ (remote) ▸ PUT ▸ DEFPSIST ▸ XMITQ ▸ RNAME ▸ RQMNAME	(A.18) QREP.STLA.TO.POKA.SENDQ ▸ ENABLED ▸ YES ▸ (A.18a)=(A.13) MQZ1XMITQ ▸ (A.18b)=(B.19) QREP.STLA.TO.POKA.SENDQ ▸ (A.18c)=(B.11) MQZ1	(B.18) QREP.POKA.TO.STLA.SENDQ ▸ ENABLED ▸ YES ▸ (B.18a)=(B.13) MQS1XMITQ ▸ (B.18b)=(A.19) QREP.POKA.TO.STLA.SENDQ ▸ (B.18c)=(A.11) MQS1
ReceiveQ (local) ▸ PUT ▸ GET ▸ SHARE ▸ DEFSOPT ▸ DEFPSIST ▸ INDXTYPE ▸ MSGDLVSQ ▸ MAXMSGL (or use default) ▸ MAXDEPTH (or use default)	(A.19) = (B.18b) QREP.POKA.TO.STLA.RECVQ ▸ ENABLED ▸ ENABLED ▸ YES ▸ SHARED ▸ YES ▸ MSGID ▸ PRIORITY ▸ (A.19a) 4194304 ▸ (A.19b) 999,999,999	(B.19) = (A.18b) QREP.STLA.TO.POKA.RECVQ ▸ ENABLED ▸ ENABLED ▸ YES ▸ SHARED ▸ YES ▸ MSGID ▸ PRIORITY ▸ (B.19a) 4194304 ▸ (B.19b) 999,999,999

Description	Server A	Server B
SpillQ ► DEFTYPE ► DEFSOPT ► MSGDLVSQ ► MAXMSGL (or use default) ► MAXDEPTH (or use default)	IBMQREP.SPILL.MODELQ ► PERMDYN ► SHARED ► FIFO ► (A.20a) 4194304 ► (A.20b) 999,999,999	IBMQREP.SPILL.MODELQ ► PERMDYN ► SHARED ► FIFO ► (B.20a) 4194304 ► (B.20b) 999,999,999
AdminQ (remote) ► PUT ► DEFPSIST ► XMITQ ► RNAME ► RQMNAME	(A.21) = (B.17) QREP.POKA.ADMINQ ► ENABLED ► YES ► (A.21a)=(A.13) MQZ1XMITQ ► (A.21b)=(B.17) QREP.POKA.ADMINQ ► (A.21c)=(B.11) MQZ1	(B.21) = (A.17) QREP.STLA.ADMINQ ► ENABLED ► YES ► (B.21a)=(B.13) MQS1XMITQ ► (B.21b)=(A.17) QREP.STLA.ADMINQ ► (B.21c)=(A.11) MQS1

Table 3-3 Q replication configuration information

Description	Server A	Server B
QCAPTURE Schema	(A.22) ITSO	(B.22) ITSO
QAPPLY Schema	(A.23) ITSO	(B.23) ITSO
Q Capture path (log files)	(A.24) //DB2PROD	(B.24) //DB2PROD
Q Apply path (log files)	(A.25) //DB2PROD	(B.25) //DB2PROD
Q replication type (U/B/P)	(A.26) B	(B.26) B
Replication Queue Map name ► Max message length (KB) ► Error handling ► Num apply agents ► Memory buffer for Recvq (MB) ► Allow QCapture to send heartbeat ► Heartbeat interval (sec)	(A.27) QMAP_STL_TO_POK ► (A.27a) 64 ► (A.27b) Stop Q Capture ► (A.27c) 16 ► (A.27d) 2 ► (A.27e) yes ► (A.27f) 60	(B.27) QMAP_POK_TO_STL ► (B.27a) 64 ► (B.27b) Stop Q Capture ► (B.27c) 16 ► (B.27d) 2 ► (B.27e) yes ► (B.27f) 60
Q Subscriptions parameters ► Target table schema/creator ► Target table index schema/name ► Target tablespaces ► Check for conflicts setting ► Conflict resolution action ► Error response action ► Initial load option ► Source server for initial load ► Qsub start automatic	► (A.28a) same ► (A.28b) same ► (A.28c) each gets own ► (A.28d) all columns ► (A.28e) D8G1 wins ► (A.28f) Stop Q Apply program ► (A.28g) Automatic ► (A.28h) DT11 ► (A.28i) yes	► (B.28a) same ► (B.28b) same ► (B.28c) each gets own ► (B.28d) all columns ► (B.28e) D8G1 wins ► (B.28f) Stop Q Apply program ► (B.28g) Automatic ► (B.28h) DT11 ► (B.28i) yes

Description	Server A	Server B
Q Subscriptions list	(A.29) 23 ITSO-xxxx tables	(B.29) 23 ITSO-xxxx tables

Table 3-4 Replication Alert Monitor configuration information

Description	Designated server
Host name	(M.1) WTSC53
Host IP address	(M.2) 9.12.6.77
DB2 subsystem name or instance owner	(M.3) D8G1
DB2 tcp port	(M.4) 38060
Monitor database server/alias information ► Database server/alias ► db2instance ► User ID/password ► Contact name ► Contact email address ► Monitor qualifier ► Schema name	 ► (M.5a) DB8G ► (M.5b) D8G1 ► (M.5c) QREPADM/xxx ► (M.5d) QMONITOR ► (M.5e) QREPADM@us.ibm.com® ► (M.5f) ITSO_QMON ► ASN
Monitor log path (not for z/OS)	(M.6)
DB2 group name (z/OS only)	(M.7) D8GG
DB2 location name (z/OS only)	(M.8) DB8G

Note: These tables and our configuration use a single subsystem member of the DB2 share group as the database alias for connecting replication programs to the database. If one is running in a share group, for greater flexibility consider using the DB2 group name as the alias, and placing the MQ objects in a WebSphere MQ cluster. That will allow replication programs to run on any member of the sysplex.

3.5.4 Step 4: Set up user IDs, privileges, and database servers

The user IDs that run the Q replication and event publishing programs need authority to connect to servers, access or update tables, put and get from WebSphere MQ queues, and perform other operations such as write to directories. In addition, appropriate databases to be used in replication need to be created and cataloged for access by the Replication Center. The steps involved are described here.

User IDs and privileges

User IDs need to be defined for executing Q Capture (reference A.7b in Table 3-1 on page 85) and Q Apply (reference A.7c in Table 3-1 on page 85).

- Q Capture

  User IDs that run a Q Capture program must:

  – Be registered with access to UNIX System Services (USS).

  – The Q Capture program must be APF authorized, which means that the program must reside in an APF authorized library, and the user must have execution access to it.

  – Be defined to use z/OS UNIX or OS/390® UNIX (must have an OMVS segment).

    An OMVS segment is the portion of a RACF® profile that contains OMVS logon information.

  – Have read and write access permission to either the /tmp directory or the directory that is specified by the TMPDIR environment variable.

  – Have SYSADM authority, or the following specific privileges:

    - SELECT, UPDATE, INSERT, and DELETE for all control tables on the Q Capture server
    - SELECT for the DB2 catalog (SYSIBM.SYSTABLES, SYSIBM.SYSCOLUMNS, SYSIBM.SYSTABCONST, SYSIBM.SYSKEYCOLUSE, SYSIBM.SYSINDEXES, and SYSIBM.SYSKEYS)
    - TRACE
    - MONITOR1 and MONITOR2
    - EXECUTE for the Q Capture program packages

- Q Apply

  User IDs that run a Q Apply program must:

  – Be registered with access to UNIX System Services (USS).

  – The Q Apply program must be APF authorized, which means that the program must reside in an APF authorized library, and the user must have execution access to it. The program must be APF-authorized in order for the Q Apply program to use the Automatic Restart Manager (ARM) or produce SVCDUMPs.

  – Be defined to use z/OS UNIX or OS/390® UNIX (must have an OMVS segment).

    An OMVS segment is the portion of a RACF® profile that contains OMVS logon information.

– Have read and write access permission to either the /tmp directory or the directory that is specified by the TMPDIR environment variable.
– Have SYSADM authority, or the following specific privileges:
  - SELECT, UPDATE, INSERT, and DELETE for all control tables on the Q Apply server
  - SELECT on the source tables if the Q Apply program will be used to load target tables
  - SELECT for the DB2 catalog (SYSIBM.SYSRELS, SYSIBM.SYSTABLES, and SYSIBM.SYSDUMMY1)
  - EXECUTE for the Q Apply program packages
  - Have WRITE access to the directory that is specified by the APPLY_PATH parameter (USS) or high-level qualifier (z/OS)

The following steps created the user IDs and assigned them the required system and database privileges:

1. We created the user IDs DB2PROD (references A.7b, A.7c, B.7b, and B.7c) and QREPADM (reference A.9 and B.9) on each of the servers STLABD1 and WTSC53.
2. We created a logical database QREPITSO (references A.10 and B.10) on each of the DB2 subsystems DT11 (reference A.3) on server STLABD1 and D8G1 (reference B.3) on server WTSC53 using the user ID QREPADM, as shown in Example 3-5.

> **Note:** The QREPADM user ID must either have SYSADM or SYSCTRL authority to create this database. If such authority cannot be provided, then a database administrator who has such privileges should grant QREPADM with the CREATESG and CREATEDBA privileges, as follows:
>
> ```
> GRANT CREATESG to QREPADM;
> GRANT CREATEDBA to QREPADM;
> ```

Example 3-5 Create logical database QREPITSO on both servers

```
-- This member will create a database for the qreplication control tables.
-- When using the Replication Center, the database must already be defined.
############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 3-1 on page 85
############################################################
##QREPADM from reference A.9 and B.9 in Table 3-1 on page 85
##ITSOQREP from reference A.10a and B.10a in Table 3-1 on page 85
##TOTDDM from reference A.10b and B.10b in Table 3-1 on page 85
```

```
##DB8GU from reference A.10c and B.10c in Table 3-1 on page 85
##QREPITSO from reference A.10 and B.10 in Table 3-1 on page 85
#############################################################

SET CURRENT SQLID = 'QREPADM';
CREATE STOGROUP ITSOQREP  VOLUMES ("TOTDDM") VCAT DB8GU ;
COMMIT WORK;
CREATE DATABASE QREPITSO  STOGROUP ITSOQREP   BUFFERPOOL BP0;
COMMIT WORK;
```

3. User ID DB2PROD will be used to write log files to MVS datasets as discussed in “Step 6f: Start the Q Capture on both servers” on page 173 and “Step 6g: Start the Q Apply on both servers” on page 176. This user ID “DB2PROD” must have RACF authority to create sequential datasets with the High Level Qualifier DB2PROD.

   On each of the servers STLABD1 and WTSC53, we created directories //DB2PROD (references A.24, B.24, A.25 and B.25) where the Q Capture and Q Apply programs will operate and need to have write permissions.

Example 3-6 RACF authority to create sequential datasets

```
#############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 3-3 on page 88
############################################################
##//DB2PROD from reference A.24, B.24, A.25 and B.25 in Table 3-3 on page 88
#############################################################
ADDSD     'DB2PROD.**'    UACC(NONE)   OWNER(DB2PROD)
           PE 'DB2PROD.**' ACCESS(ALTER) ID(DB2PROD)
```

4. The user IDs (DB2PROD) that execute Q Capture and Q Apply may be granted the SYSADM privilege in order to manage the various objects, as shown in Example 3-7.

Example 3-7 GRANT SYSADM authority

```
GRANT SYSADM TO DB2PROD;
```

   However, granting of SYSADM is generally not acceptable in many customer environments, and therefore individual privileges may have to be granted, as shown in Example 3-8. We therefore granted user ID “DB2PROD” on each server the individual authorities on the various objects required by Q Capture and Q Apply, as shown in Example 3-8.

Important: The following assumes that the Q replication control tables have been created either via the Replication Center (as described in Figure 3-18 on page 125 and Figure 3-30 on page 137) or using ASNCLP commands (as described in Example 3-40 on page 194 and Example 3-42 on page 209).

Example 3-8 Grant individual privileges on various objects to DB2PROD

```
#############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 3-1 on page 85
############################################################
##QREPADM from reference A.9 in Table 3-1 on page 85
##DB2PROD from reference A.7b and A.7c in Table 3-1 on page 85
#############################################################
--CREATE tablespaces, tables, and indexes on the Qreplication control tables
--database
GRANT CREATETS ON DATABASE QREPITSO TO QREPADM;
GRANT CREATETAB ON DATABASE QREPITSO TO QREPADM;
GRANT USE OF BUFFERPOOL xxx TO QREPADM;
GRANT DBADM ON DATABASE QREPITSO TO QREPADM;
--SELECT on all Qreplication Control tables, and the DB2 catalog tables.
--The Control tables must exist first, so the GRANTs for tables ITSO.*
--should be done after the tables are created in the first steps of Replication
--Center, or ASNCLP:
GRANT SELECT ON TABLE SYSIBM.SYSTABLES   TO DB2PROD;
GRANT SELECT ON TABLE SYSIBM.SYSTABLES   TO QREPADM;
GRANT SELECT ON TABLE SYSIBM.SYSCATALOG  TO DB2PROD;
GRANT SELECT ON TABLE SYSIBM.SYSCATALOG  TO QREPADM;
GRANT SELECT ON TABLE SYSIBM.SYSDUMMYA   TO DB2PROD;
GRANT SELECT ON TABLE SYSIBM.SYSDUMMYA   TO QREPADM;
GRANT SELECT ON TABLE SYSIBM.SYSDUMMYU   TO DB2PROD;
GRANT SELECT ON TABLE SYSIBM.SYSDUMMYU   TO QREPADM;
GRANT SELECT ON TABLE SYSIBM.SYSDUMMYE   TO DB2PROD;
GRANT SELECT ON TABLE SYSIBM.SYSDUMMYE   TO QREPADM;
GRANT SELECT ON TABLE SYSIBM.SYSPLAN     TO DB2PROD;
GRANT SELECT ON TABLE SYSIBM.SYSPLAN     TO QREPADM;
GRANT SELECT ON TABLE SYSIBM.SYSPACKAGE  TO DB2PROD;
GRANT SELECT ON TABLE SYSIBM.SYSPACKAGE  TO QREPADM;
GRANT SELECT ON TABLE SYSIBM.SYSRELS     TO DB2PROD;
GRANT SELECT ON TABLE SYSIBM.SYSRELS     TO QREPADM;
GRANT SELECT ON TABLE SYSIBM.SYSINDEXES  TO DB2PROD;
GRANT SELECT ON TABLE SYSIBM.SYSINDEXES  TO QREPADM;
GRANT SELECT ON TABLE SYSIBM.SYSKEYS     TO DB2PROD;
GRANT SELECT ON TABLE SYSIBM.SYSKEYS     TO QREPADM;
GRANT SELECT ON TABLE SYSIBM.SYSKEYCOLUSE  TO DB2PROD;
```

```
GRANT SELECT ON TABLE SYSIBM.SYSKEYCOLUSE  TO QREPADM;
GRANT SELECT ON TABLE SYSIBM.SYSFOREIGNKEYS  TO DB2PROD;
GRANT SELECT ON TABLE SYSIBM.SYSFOREIGNKEYS  TO QREPADM;
GRANT SELECT ON TABLE ITSO.IBMQREP_ADMINMSG   TO DB2PROD;
GRANT SELECT ON TABLE ITSO.IBMQREP_APPLYENQ   TO DB2PROD;
GRANT SELECT ON TABLE ITSO.IBMQREP_APPLYMON   TO DB2PROD;
GRANT SELECT ON TABLE ITSO.IBMQREP_APPLYPARMS TO DB2PROD;
GRANT SELECT ON TABLE ITSO.IBMQREP_APPLYTRACE TO DB2PROD;
GRANT SELECT ON TABLE ITSO.IBMQREP_CAPENQ     TO DB2PROD;
GRANT SELECT ON TABLE ITSO.IBMQREP_CAPMON     TO DB2PROD;
GRANT SELECT ON TABLE ITSO.IBMQREP_CAPPARMS   TO DB2PROD;
GRANT SELECT ON TABLE ITSO.IBMQREP_CAPQMON    TO DB2PROD;
GRANT SELECT ON TABLE ITSO.IBMQREP_CAPTRACE   TO DB2PROD;
GRANT SELECT ON TABLE ITSO.IBMQREP_DELTOMB    TO DB2PROD;
GRANT SELECT ON TABLE ITSO.IBMQREP_DONEMSG    TO DB2PROD;
GRANT SELECT ON TABLE ITSO.IBMQREP_EXCEPTIONS TO DB2PROD;
GRANT SELECT ON TABLE ITSO.IBMQREP_RECVQUEUES TO DB2PROD;
GRANT SELECT ON TABLE ITSO.IBMQREP_SAVERI     TO DB2PROD;
GRANT SELECT ON TABLE ITSO.IBMQREP_SENDQUEUES TO DB2PROD;
GRANT SELECT ON TABLE ITSO.IBMQREP_SIGNAL     TO DB2PROD;
GRANT SELECT ON TABLE ITSO.IBMQREP_SPILLEDROW TO DB2PROD;
GRANT SELECT ON TABLE ITSO.IBMQREP_SPILLQS    TO DB2PROD;
GRANT SELECT ON TABLE ITSO.IBMQREP_SRCH_COND  TO DB2PROD;
GRANT SELECT ON TABLE ITSO.IBMQREP_SRC_COLS   TO DB2PROD;
GRANT SELECT ON TABLE ITSO.IBMQREP_SUBS       TO DB2PROD;
GRANT SELECT ON TABLE ITSO.IBMQREP_TARGETS    TO DB2PROD;
GRANT SELECT ON TABLE ITSO.IBMQREP_TRG_COLS   TO DB2PROD;
GRANT SELECT ON TABLE ASN.IBMSNAP_CAPENQ      TO DB2PROD;
GRANT SELECT ON TABLE ASN.IBMSNAP_CAPMON      TO DB2PROD;
GRANT SELECT ON TABLE ASN.IBMSNAP_CAPPARMS    TO DB2PROD;
GRANT SELECT ON TABLE ASN.IBMSNAP_CAPTRACE    TO DB2PROD;
GRANT SELECT ON TABLE ASN.IBMSNAP_PRUNCNTL    TO DB2PROD;
GRANT SELECT ON TABLE ASN.IBMSNAP_PRUNE_LOCK  TO DB2PROD;
GRANT SELECT ON TABLE ASN.IBMSNAP_PRUNE_SET   TO DB2PROD;
GRANT SELECT ON TABLE ASN.IBMSNAP_REGISTER    TO DB2PROD;
GRANT SELECT ON TABLE ASN.IBMSNAP_RESTART     TO DB2PROD;
GRANT SELECT ON TABLE ASN.IBMSNAP_SIGNAL      TO DB2PROD;
GRANT SELECT ON TABLE ASN.IBMSNAP_UOW         TO DB2PROD;
--UPDATE, INSERT, DELETE on all the Qreplication Control tables
GRANT SELECT, INSERT, DELETE ON TABLE ITSO.IBMQREP_ADMINMSG   TO DB2PROD;
GRANT SELECT, INSERT, DELETE ON TABLE ITSO.IBMQREP_APPLYENQ   TO DB2PROD;
GRANT SELECT, INSERT, DELETE ON TABLE ITSO.IBMQREP_APPLYMON   TO DB2PROD;
GRANT SELECT, INSERT, DELETE ON TABLE ITSO.IBMQREP_APPLYPARMS TO DB2PROD;
GRANT SELECT, INSERT, DELETE ON TABLE ITSO.IBMQREP_APPLYTRACE TO DB2PROD;
GRANT SELECT, INSERT, DELETE ON TABLE ITSO.IBMQREP_CAPENQ     TO DB2PROD;
GRANT SELECT, INSERT, DELETE ON TABLE ITSO.IBMQREP_CAPMON     TO DB2PROD;
GRANT SELECT, INSERT, DELETE ON TABLE ITSO.IBMQREP_CAPPARMS   TO DB2PROD;
GRANT SELECT, INSERT, DELETE ON TABLE ITSO.IBMQREP_CAPQMON    TO DB2PROD;
GRANT SELECT, INSERT, DELETE ON TABLE ITSO.IBMQREP_CAPTRACE   TO DB2PROD;
```

```
GRANT SELECT, INSERT, DELETE ON TABLE ITSO.IBMQREP_DELTOMB    TO DB2PROD;
GRANT SELECT, INSERT, DELETE ON TABLE ITSO.IBMQREP_DONEMSG    TO DB2PROD;
GRANT SELECT, INSERT, DELETE ON TABLE ITSO.IBMQREP_EXCEPTIONS TO DB2PROD;
GRANT SELECT, INSERT, DELETE ON TABLE ITSO.IBMQREP_RECVQUEUES TO DB2PROD;
GRANT SELECT, INSERT, DELETE ON TABLE ITSO.IBMQREP_SAVERI     TO DB2PROD;
GRANT SELECT, INSERT, DELETE ON TABLE ITSO.IBMQREP_SENDQUEUES TO DB2PROD;
GRANT SELECT, INSERT, DELETE ON TABLE ITSO.IBMQREP_SIGNAL     TO DB2PROD;
GRANT SELECT, INSERT, DELETE ON TABLE ITSO.IBMQREP_SPILLEDROW TO DB2PROD;
GRANT SELECT, INSERT, DELETE ON TABLE ITSO.IBMQREP_SPILLQS    TO DB2PROD;
GRANT SELECT, INSERT, DELETE ON TABLE ITSO.IBMQREP_SRCH_COND  TO DB2PROD;
GRANT SELECT, INSERT, DELETE ON TABLE ITSO.IBMQREP_SRC_COLS   TO DB2PROD;
GRANT SELECT, INSERT, DELETE ON TABLE ITSO.IBMQREP_SUBS       TO DB2PROD;
GRANT SELECT, INSERT, DELETE ON TABLE ITSO.IBMQREP_TARGETS    TO DB2PROD;
GRANT SELECT, INSERT, DELETE ON TABLE ITSO.IBMQREP_TRG_COLS   TO DB2PROD;
GRANT SELECT, INSERT, DELETE ON TABLE ASN.IBMSNAP_CAPENQ      TO DB2PROD;
GRANT SELECT, INSERT, DELETE ON TABLE ASN.IBMSNAP_CAPMON      TO DB2PROD;
GRANT SELECT, INSERT, DELETE ON TABLE ASN.IBMSNAP_CAPPARMS    TO DB2PROD;
GRANT SELECT, INSERT, DELETE ON TABLE ASN.IBMSNAP_CAPTRACE    TO DB2PROD;
GRANT SELECT, INSERT, DELETE ON TABLE ASN.IBMSNAP_PRUNCNTL    TO DB2PROD;
GRANT SELECT, INSERT, DELETE ON TABLE ASN.IBMSNAP_PRUNE_LOCK  TO DB2PROD;
GRANT SELECT, INSERT, DELETE ON TABLE ASN.IBMSNAP_PRUNE_SET   TO DB2PROD;
GRANT SELECT, INSERT, DELETE ON TABLE ASN.IBMSNAP_REGISTER    TO DB2PROD;
GRANT SELECT, INSERT, DELETE ON TABLE ASN.IBMSNAP_RESTART     TO DB2PROD;
GRANT SELECT, INSERT, DELETE ON TABLE ASN.IBMSNAP_SIGNAL      TO DB2PROD;
GRANT SELECT, INSERT, DELETE ON TABLE ASN.IBMSNAP_UOW         TO DB2PROD;
--UPDATE, INSERT, DELETE on all replicated user tables
GRANT UPDATE, INSERT, DELETE ON TABLE ITSO.yyyyyyyyyyyyyyyyyy TO DB2PROD;
--EXECUTE on Qreplication Plans
GRANT EXECUTE ON PLAN ASNAM820 TO DB2PROD;
GRANT EXECUTE ON PLAN ASNQA820 TO DB2PROD;
GRANT EXECUTE ON PLAN ASNRD820 TO DB2PROD;
GRANT EXECUTE ON PLAN ASNQC820 TO DB2PROD;
--TRACE, MONITOR1, and MONITOR2
GRANT TRACE TO DB2PROD;
GRANT MONITOR1 TO DB2PROD;
GRANT MONITOR2 TO DB2PROD;
```

5. The user IDs that will be used to run the Q replication Apply and Capture programs need to have adequate MQ authorities for all WebSphere MQ objects including:

   – MQCONN to the MQ managers
   – MQOPEN on Send, Admin, Restart, Spill, and Receive queues
   – MQINQ on Send, Admin, Restart, and Receive queues
   – MQGET on Admin, Restart, Spill, and Receive queues
   – MQPUT on Admin, Restart, Spill, and Send queues
   – MQCMIT on Send queue
   – MQBACK on Send queue

– MQCLOSE on Spill queues

For a detailed description, refer to Chapter 7 of the *Replication and Event Publishing Guide and Reference*, SC16-7568-00.

In Z/OS, WebSphere MQ authorities are managed through RACF or ACF2 definitions; only RACF examples are shown here.

Authorization may be accomplished by either:

– Adding the user IDs to the WebSphere administration group on each z/OS system (MQMADM in this environment), as shown in Example 3-9.

> **Note:** This approach is rarely used in z/OS environments.

Example 3-9 Add user IDs QREPADM and DB2PROD to MQMADM group

```
##############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 3-1 on page 85
#############################################################
##DB2PROD from reference A.7b and B.7b in Table 3-1 on page 85
##QREPADM from reference A.9 and B.9 in Table 3-1 on page 85
##############################################################
CONNECT DB2PROD GROUP(MQMADM)
CONNECT QREPADM GROUP(MQMADM)
```

– Granting individual authorities to the specific objects required

> **Note:** This assumes that the WebSphere MQ objects referenced in Example 3-10 and Example 3-11 already exist. In our case, these WebSphere MQ objects are defined in 3.5.5, "Step 5: Configure WebSphere MQ" on page 101.

For full details on these RACF definitions for WebSphere MQ, refer to Chapter 13 of the *WebSphere MQ System Setup Guide*, SC34-6052-00.

Example 3-10 Grant individual privileges on server STLABD1

```
##############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 3-1 on page 85, and Table 3-2 on page 86
#############################################################
##MQS1 from reference A.11 in Table 3-2 on page 86
##DB2PROD from reference A.7b in Table 3-1 on page 85
##QREPADM from reference A.9 in Table 3-1 on page 85
```

```
##############################################################
SETROPTS CLASSACT(MQADMIN,MQQUEUE,MQPROC,MQNLIST,MQCONN,MQCMDS)
RDEFINE MQCONN MQS1.BATCH UACC(NONE)
       PERMIT MQS1.BATCH CLASS(MQCONN) ID(DB2PROD) ACCESS(READ)
RDEFINE MQQUEUE MQS1.QREP.** UACC(NONE)
       PERMIT MQS1.QREP.** CLASS(MQQUEUE) ID(DB2PROD) ACCESS(UPDATE)
RDEFINE MQQUEUE MQS1.IBMQREP.SPILL.** UACC(NONE)
       PERMIT MQS1.IBMQREP.SPILL.** CLASS(MQQUEUE) ID(DB2PROD) ACCESS(ALTER)
RDEFINE MQCONN MQS1.BATCH UACC(NONE)
       PERMIT MQS1.BATCH CLASS(MQCONN) ID(QREPADM) ACCESS(READ)
RDEFINE MQQUEUE MQS1.QREP.** UACC(NONE)
       PERMIT MQS1.QREP.** CLASS(MQQUEUE) ID(QREPADM) ACCESS(UPDATE)
RDEFINE MQQUEUE MQS1.IBMQREP.SPILL.** UACC(NONE)
       PERMIT MQS1.IBMQREP.SPILL.** CLASS(MQQUEUE) ID(QREPADM) ACCESS(ALTER)
```

Example 3-11 Grant individual privileges on server WTSC53

```
##############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 3-1 on page 85, and Table 3-2 on page 86
#############################################################
##MQZ1 from reference B.11 in Table 3-2 on page 86
##DB2PROD from reference B.7b in Table 3-1 on page 85
##QREPADM from reference B.9 in Table 3-1 on page 85
##############################################################
SETROPTS CLASSACT(MQADMIN,MQQUEUE,MQPROC,MQNLIST,MQCONN,MQCMDS)
RDEFINE MQCONN MQZ1.BATCH UACC(NONE)
       PERMIT MQZ1.BATCH CLASS(MQCONN) ID(DB2PROD) ACCESS(READ)
RDEFINE MQQUEUE MQZ1.QREP.** UACC(NONE)
      PERMIT MQZ1.QREP.** CLASS(MQQUEUE) ID(DB2PROD) ACCESS(UPDATE)
RDEFINE MQQUEUE MQZ1.IBMQREP.SPILL.** UACC(NONE)
      PERMIT MQZ1.IBMQREP.SPILL.** CLASS(MQQUEUE) ID(DB2PROD) ACCESS(ALTER)
RDEFINE MQCONN MQZ1.BATCH UACC(NONE)
      PERMIT MQZ1.BATCH CLASS(MQCONN) ID(QREPADM) ACCESS(READ)
RDEFINE MQQUEUE MQZ1.QREP.** UACC(NONE)
      PERMIT MQZ1.QREP.** CLASS(MQQUEUE) ID(QREPADM) ACCESS(UPDATE)
RDEFINE MQQUEUE MQZ1.IBMQREP.SPILL.** UACC(NONE)
      PERMIT MQZ1.IBMQREP.SPILL** CLASS(MQQUEUE) ID(QREPADM) ACCESS(ALTER)
```

Database servers

The following actions need to be performed to set up the database servers:

1. We assume that the DB2 subsystems have been genned on both servers (DT11 reference A.7a on STLABD1) and (D8G1 reference B.7a on WTSC53), respectively. The logical database, which will contain the Q replication control tables, must be predefined, as shown in Example 3-5 on page 91.

2. If Q replication uses the automatic load feature, then the two DB2 subsystems must be known to one another via a DRDA® connection. This is accomplished by updating the system catalog Connection Database (CDB) tables on each server. The SQL commands to accomplish this on WTSC53 (D8G1) to allow connection to STLABD1(DT11) are shown in Example 3-12, and the SQL commands to allow connection on STLABD1(DT11) to WTSC53(D8G1) are shown in Example 3-13.

   For details on setting up DRDA connectivity, refer to the *Distributed Functions of DB2 for z/OS and OS/390*, SG24-6952-00.

Example 3-12 DRDA connection definition on WTSC53 (D8G1) to DB2 subsystem DT11

```
#############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 3-1 on page 85
############################################################
##STLABD1 is a made up name - not referenced in the template
##9.30.132.94 from reference A.2 in Table 3-1 on page 85
##8010 from reference B.6 in Table 3-1 on page 85
##DT11 from reference A.7a in Table 3-1 on page 85
##DSNT1 from reference A.5 in Table 3-1 on page 85
#############################################################
INSERT INTO SYSIBM.LOCATIONS (LOCATION, LINKNAME, IBMREQD, PORT, TPN)
      VALUES (
-- ENTER VALUES BELOW         COLUMN NAME    DATA TYPE       LENGTH   NULLS
      'DSNT1    '              , -- LOCATION       CHAR            16       NO
      'DT11     '              , -- LINKNAME       CHAR            8        NO
      ' '                      , -- IBMREQD        CHAR            1        NO
      '8010 '                  , -- PORT           CHAR            32       NO
      ' '                    );  -- TPN            VARCHAR         64       NO

INSERT INTO SYSIBM.IPNAMES (LINKNAME, SECURITY_OUT, USERNAMES, IBMREQD,IPADDR)
      VALUES (
-- ENTER VALUES BELOW         COLUMN NAME    DATA TYPE       LENGTH   NULLS
      'DSNT1    '            , -- LINKNAME       CHAR            8        NO
      'P'                    , -- SECURITY_OUT   CHAR            1        NO
      'O'                    , -- USERNAMES      CHAR            1        NO
      ' '                    , -- IBMREQD        CHAR            1        NO
      '9.30.132.94'         ); -- IPADDR         VARCHAR         254      NO

INSERT INTO SYSIBM.USERNAMES (TYPE, AUTHID, LINKNAME,
      NEWAUTHID, PASSWORD,IBMREQD)
      VALUES (
-- ENTER VALUES BELOW         COLUMN NAME    DATA TYPE       LENGTH   NULLS
      'O'                    , -- TYPE           CHAR            1        NO
      ' '                    , -- AUTHID         CHAR            8        NO
      'DSNT1    '            , -- LINKNAME       CHAR            8        NO
```

```
          'DB2PROD '            , -- NEWAUTHID      CHAR          8         NO
          'xxxxxxx '            , -- PASSWORD       CHAR          8         NO
          ' '                 );  -- IBMREQD        CHAR          1         NO
```

Example 3-13 DRDA connection definition on STLABD1(DT11) to DB2 subsystem D8G1

```
#############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 3-1 on page 85
############################################################
##WTSC53 is a madeup name - not referenced in the template
##9.12.6.77 from reference B.2 in Table 3-1 on page 85
##38060 from reference A.6 in Table 3-1 on page 85
##D8G1 from reference B.7a in Table 3-1 on page 85
##DB8G from reference A.5 in Table 3-1 on page 85
#############################################################
INSERT INTO SYSIBM.LOCATIONS (LOCATION, LINKNAME, IBMREQD, PORT, TPN)
      VALUES (
-- ENTER VALUES BELOW            COLUMN NAME    DATA TYPE       LENGTH   NULLS
      'DB8G    '                , -- LOCATION       CHAR          16        NO
      'D8G1    '                , -- LINKNAME       CHAR          8         NO
      ' '                       , -- IBMREQD        CHAR          1         NO
      '38060'                   , -- PORT           CHAR          32        NO
      ' '                     );  -- TPN            VARCHAR       64        NO

INSERT INTO SYSIBM.IPNAMES (LINKNAME, SECURITY_OUT, USERNAMES, IBMREQD,IPADDR)
      VALUES (
-- ENTER VALUES BELOW            COLUMN NAME    DATA TYPE       LENGTH   NULLS
      'DB8G   '               , -- LINKNAME       CHAR          8         NO
      'P'                     , -- SECURITY_OUT   CHAR          1         NO
      'O'                     , -- USERNAMES      CHAR          1         NO
      ' '                     , -- IBMREQD        CHAR          1         NO
      '9.12.6.77'           );  -- IPADDR         VARCHAR       254       NO

INSERT INTO SYSIBM.USERNAMES (TYPE, AUTHID, LINKNAME,
      NEWAUTHID, PASSWORD,IBMREQD)
      VALUES (
-- ENTER VALUES BELOW            COLUMN NAME    DATA TYPE       LENGTH   NULLS
      'O'                     , -- TYPE           CHAR          1         NO
      ' '                     , -- AUTHID         CHAR          8         NO
      'DB8G   '               , -- LINKNAME       CHAR          8         NO
      'DB2PROD '              , -- NEWAUTHID      CHAR          8         NO
      'xxxxxxx '              , -- PASSWORD       CHAR          8         NO
      ' '                   );  -- IBMREQD        CHAR          1         NO
```

3. We tested the DRDA connectivity to subsystem D8G1 from host STLABD1 and to subsystem DT11 from host WTSC53 using the SPUFI ISPF panels on each side, as shown in Example 3-14 and Example 3-15, respectively.

Example 3-14 Test connectivity to D8G1 from server STLABD1 (DT11)

```
#############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 3-1 on page 85
############################################################
##DT11 from reference A.3 in Table 3-1 on page 85
##DB8G from reference B.5 in Table 3-1 on page 85
#############################################################
SPUFI                              SSID: DT11
===>

Enter the input data set name:        (Can be sequential or partitioned)
1  DATA SET NAME ... ===> SPUFI(USERNAME)
2  VOLUME SERIAL ... ===>            (Enter if not cataloged)
3  DATA SET PASSWORD ===>            (Enter if password protected)

Enter the output data set name:       (Must be a sequential data set)
4  DATA SET NAME ... ===> SPUFIOUT

Specify processing options:
5  CHANGE DEFAULTS   ===> YES        (Y/N - Display SPUFI defaults panel?)
6  EDIT INPUT ...... ===> YES        (Y/N - Enter SQL statements?)
7  EXECUTE ......... ===> YES        (Y/N - Execute SQL statements?)
8  AUTOCOMMIT ...... ===> YES        (Y/N - Commit after successful run?)
9  BROWSE OUTPUT ... ===> YES        (Y/N - Browse output data set?)

For remote SQL processing:
10  CONNECT LOCATION  ===> DB8G
```

Example 3-15 Test connectivity to DT11 from server WTSC53 (D8G1)

```
#############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 3-1 on page 85
############################################################
##D8G1 from reference B.3 in Table 3-1 on page 85
##DSNT1 from reference A.5 in Table 3-1 on page 85
#############################################################
SPUFI                              SSID: D8G1
===>

Enter the input data set name:        (Can be sequential or partitioned)
```

```
1  DATA SET NAME ... ===> SPUFI(USERNAME)
2  VOLUME SERIAL ... ===>            (Enter if not cataloged)
3  DATA SET PASSWORD ===>            (Enter if password protected)

Enter the output data set name:       (Must be a sequential data set)
4  DATA SET NAME ... ===> SPUFIOUT

Specify processing options:
5  CHANGE DEFAULTS   ===> YES        (Y/N - Display SPUFI defaults panel?)
6  EDIT INPUT ...... ===> YES        (Y/N - Enter SQL statements?)
7  EXECUTE ......... ===> YES        (Y/N - Execute SQL statements?)
8  AUTOCOMMIT ...... ===> YES        (Y/N - Commit after successful run?)
9  BROWSE OUTPUT ... ===> YES        (Y/N - Browse output data set?)

For remote SQL processing:
10  CONNECT LOCATION  ===> DSNT1
```

3.5.5 Step 5: Configure WebSphere MQ

Several WebSphere MQ objects need to be created in support of Q replication, and configured to suit an organization's particular workload. The WebSphere MQ objects are:

- Queue Manager
- Channels
- Local queues
- Remote queues

There is a hierarchical relationship between these objects—the Queue Manager is created at the highest level and all objects are managed by the Queue Manager.

To support a typical Q replication bidirectional set up, the following objects must be defined on each system:

- Queue Manager
- Sender Channel
- Receiver Channel
- Transmit Queue (Local)
- Capture Administration Queue (Local)
- Capture Restart Queue (Local)
- Capture Send Queue (Remote)
- Apply Receive Queue (Local)
- Apply Spill Queue (Local Model)
- Apply Administration Queue (Remote)

Configure WebSphere MQ objects

The ASNQDEF member of the SASNSAMP contains a script for creating the required WebSphere MQ objects. Example 3-16 shows the JCL script for creating all the WebSphere MQ objects necessary for bidirectional replication on STLABD1 MQ manager MQS1 (reference A.11), while Example 3-17 shows the JCL script for creating the objects on WTSC53 MQ manager MQZ1 (reference B.11).

> **Attention:** The creation of an MQ manager on z/OS is not covered here. For full details on creating an MQ manager, refer to Chapter 1 of the *WebSphere MQ System Setup Guide*, SC34-6052-00.

> **Important:** The scripts should be run by a user ID with MQ Administration authority.

> **Note:** The values for the names of the WebSphere MQ objects and the configuration parameters are substituted from the templates listed in Table 3-1 on page 85, Table 3-2 on page 86, and Table 3-3 on page 88. The specific references are listed in the script.

Example 3-16 Create necessary MQ objects for bidirectional replication on STLABD1

```
//ASNQDEF  JOB NOTIFY=&SYSUID,
//         MSGCLASS=H,MSGLEVEL=(1,1),
//         REGION=0M,TIME=NOLIMIT
//******************************************************************/
//*                                                                */
//*    IBM DB2 Information Integrator Replication Version 8.2      */
//*    for z/OS (5655-I60)                                         */
//*                                                                */
//*    Sample MQSeries queue definitions for Q Replication.        */
//*    Licensed Materials - Property of IBM                        */
//*                                                                */
//*    (C) Copyright IBM Corp. 1993, 2004. All Rights Reserved     */
//*                                                                */
//*    US Government Users Restricted Rights - Use, duplication    */
//*    or disclosure restricted by GSA ADP Schedule Contract       */
//*    with IBM Corp.                                              */
//******************************************************************/
//*
//*
//* This is an example of MQSeries queue definitions for
//* Q Replication.
```

```
//*
//* Need to define ADMINQ, RESTARTQ, RECVQ and a model SPILLQ.
//* You might also need to define a Dead letter Q.
//*
//* Locate and change all occurrences of the following strings
//*   .  MQS!!0 to the name of your MQSeries target library
//*
//* The STEPLIB must include the MQSeries libraries
//* if they are not installed in the LNKLST.
//*
//*********************************************************************/
//*
//*##############################################################
//*##The variables in the commands are shown unbold, and
//*##these values are substituted from the templates described
//*##in Table 3-1 on page 85, and Table 3-2 on page 86
//*#############################################################
//*##PARM from reference A.11 in Table 3-2 on page 86
//*##############################################################
//*

//ASNQMQD1 EXEC PGM=CSQUTIL,PARM='MQS1'
//STEPLIB   DD DSN=MQS1.USERAUTH,DISP=SHR
//          DD DSN=MQ531.SCSQANLE,DISP=SHR
//          DD DSN=MQ531.SCSQAUTH,DISP=SHR
//          DD DSN=CEE.SCEERUN,DISP=SHR
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
  COMMAND DDNAME(CMDINP)
/*
//CMDINP   DD *

*
* Following are the definitions which are required to send data
* from the STLA capture instance to the POKA apply instance.
* Similar definitions are required for each capture/apply pair.
*
*Local AdminQ from reference A.17 in Table 3-2 on page 86
DEFINE REPLACE                                                       +
       QLOCAL(QREP.STLA.ADMINQ)                                      +
       DESCR('LOCAL DEFN OF ADMINQ FOR STLA CAPTURE')                +
       PUT(ENABLED)                                                  +
       GET(ENABLED)                                                  +
       SHARE                                                         +
       DEFSOPT(SHARED)                                               +
       DEFPSIST(YES)

*Local RestartQ from reference A.16 in Table 3-2 on page 86
DEFINE REPLACE                                                       +
```

```
        QLOCAL(QREP.STLA.RESTARTQ)                                    +
        DESCR('LOCAL DEFN OF RESTART FOR STLA CAPTURE')               +
        PUT(ENABLED)                                                  +
        GET(ENABLED)                                                  +
        SHARE                                                         +
        DEFSOPT(SHARED)                                               +
        DEFPSIST(YES)
*Remote SendQ QREMOTE from reference A.18 in Table 3-2 on page 86
*Remote SendQ XMITQ from reference A.18a in Table 3-2 on page 86
*Remote SendQ RNAME from reference A.18b in Table 3-2 on page 86
*Remote SendQ RQMNAME from reference A.18c in Table 3-2 on page 86
DEFINE REPLACE                                                        +
        QREMOTE(QREP.STLA.TO.POKA.SENDQ)                              +
        DESCR('REMOTE DEFN OF SEND QUEUE FROM STLA TO POKA')          +
        PUT(ENABLED)                                                  +
        XMITQ(MQZ1XMIT)                                               +
        RNAME(QREP.STLA.TO.POKA.RECVQ)                                +
        RQMNAME(MQZ1)                                                 +
        DEFPSIST(YES)


* Following are the definitions which are required to send data
* from an MQSeries queue manager named MQS1 to a Queue manager
* named MQZ1.
*Local TransmitQ QLOCAL from reference A.13 in Table 3-2 on page 86
*Local TransmitQ TRIGDATA from reference A.13a in Table 3-2 on page 86
*Local TransmitQ INITQ from reference A.13b in Table 3-2 on page 86
DEFINE REPLACE                                                        +
        QLOCAL(MQZ1XMIT)                                              +
        DESCR('TRANSMISSION QUEUE TO MQZ1')                           +
        USAGE(XMITQ)                                                  +
        PUT(ENABLED)                                                  +
        GET(ENABLED)                                                  +
        TRIGGER                                                       +
        TRIGTYPE(FIRST)                                               +
        TRIGDATA(MQS1.TO.MQZ1)                                        +
        INITQ(SYSTEM.CHANNEL.INITQ)

*
* Note that a listener must be started, on the other queue
* manager, for the port number identified in the channel
* sender definition.
*Local SDR channel CHANNEL from reference A.14 in Table 3-2 on page 86
*Local SDR channel XMITQ from reference A.14b in Table 3-2 on page 86
*Local SDR channel TRPTYPE from reference A.14a in Table 3-2 on page 86
*Local SDR channel CONNAME from reference A.14c in Table 3-2 on page 86

DEFINE REPLACE                                                        +
        CHANNEL(MQS1.TO.MQZ1)                                         +
```

```
       CHLTYPE(SDR)                                          +
       TRPTYPE(TCP)                                          +
       DESCR('SENDER CHANNEL TO MQZ1')                       +
       XMITQ(MQZ1XMIT)                                       +
       CONNAME('9.12.6.77(1540)')
*Local RCV channel CHANNEL from reference A.15 in Table 3-2 on page 86
*Local RCV channel TRPTYPE from reference A.15a in Table 3-2 on page 86
DEFINE REPLACE                                               +
       CHANNEL(MQZ1.TO.MQS1)                                 +
       CHLTYPE(RCVR)                                         +
       TRPTYPE(TCP)                                          +
       DESCR('RECEIVER CHANNEL FROM MQZ1')

DISPLAY QUEUE(*) ALL

*
* Following are the definitions which are required to receive data
* at the STLA apply instance from the POKA capture instance.
*
*Local SpillQ from reference A.20 in Table 3-2 on page 86
DEFINE REPLACE                                               +
       QMODEL('IBMQREP.SPILL.MODELQ')                        +
       DEFSOPT(SHARED)                                       +
       MAXDEPTH(999,999,999)                                    +
       MSGDLVSQ(FIFO)                                        +
       DEFTYPE(PERMDYN)
*Remote AdminQ QREMOTE from reference A.21 in Table 3-2 on page 86
*Remote AdminQ XMITQ from reference A.21a in Table 3-2 on page 86
*Remote AdminQ RNAME from reference A.21 in Table 3-2 on page 86
*Remote AdminQ RQMNAME from reference A.21c in Table 3-2 on page 86
DEFINE REPLACE                                               +
       QREMOTE(QREP.POKA.ADMINQ)                             +
       DESCR('REMOTE DEFN OF ADMINQ FOR POKA CAPTURE')       +
       PUT(ENABLED)                                          +
       XMITQ(MQZ1XMIT)                                       +
       RNAME(QREP.POKA.ADMINQ)                               +
       RQMNAME(MQZ1)                                         +
       DEFPSIST(YES)

*Local ReceiveQ QLOCAL from reference A.19 in Table 3-2 on page 86
DEFINE REPLACE                                               +
       QLOCAL(QREP.POKA.TO.STLA.RECVQ)                       +
       DESCR('LOCAL RECEIVE QUEUE - POKA TO STLA')           +
       PUT(ENABLED)                                          +
       GET(ENABLED)                                          +
       SHARE                                                 +
       DEFSOPT(SHARED)                                       +
       DEFPSIST(YES)                                         +
       INDXTYPE(MSGID)
```

```
DISPLAY QUEUE(*) ALL
//********************************************************************/
//* Starting channels                                                */
//* For the remote set up, before you start replication you need to   */
//* start WebSphere MQ interchannel communication.                   */
//* First make sure that the queue manager is running, then start the */
//* listener and then start the channel:                             */
//* From MVS console, issue the command to start the qmgr            */
//* +MQS1 start qmgr parm(MQS1parm)                                  */
//* where +MQS1 is the recognition character for the Queue manager    */
//* and MQS1parm is the parms you want to use to start the qmgr      */
//*                                                              */
//* +MQS1 sta lstr(1414) <<this starts the listener                  */
//* using tcpip protocol for queue manager CSQ1, port 1414>>         */
//*                                                               */
//* +MQS1 start channel (MQS1.TO.MQZ1)                               */
//* <<this command starts the channel>>                              */
//*                                                        */
//* In some instances it might be necessary to start the channel      */
//*  initiator before starting the channel . If so use               */
//* +MQS1 start chinit                                               */
//* Note: You need to use the same commands at the MZQ1 server.      */
//********************************************************************/
```

Note: The same listener port can be used on both host servers but it may be easier to debug if one uses a different port on each.

Example 3-17 Create necessary MQ objects for bidirectional replication on WTSC53

```
//ASNQDEF  JOB NOTIFY=&SYSUID,
//         MSGCLASS=H,MSGLEVEL=(1,1),
//         REGION=OM,TIME=NOLIMIT
/*JOBPARM S=SC53
//***************************************************************/
//*                                                             */
//*    IBM DB2 Information Integrator Replication Version 8.2    */
//*    for z/OS (5655-I60)                                      */
//*                                                             */
//*    Sample MQSeries queue definitions for Q Replication.     */
//*    Licensed Materials - Property of IBM                     */
//*                                                             */
//*    (C) Copyright IBM Corp. 1993, 2004. All Rights Reserved   */
//*                                                             */
//*    US Government Users Restricted Rights - Use, duplication  */
//*    or disclosure restricted by GSA ADP Schedule Contract     */
//*    with IBM Corp.                                           */
```

```
//*****************************************************************/
//*
//*
//* This is an example of MQSeries queue definitions for
//* Q Replication.
//*
//* Need to define ADMINQ, RESTARTQ, RECVQ and a model SPILLQ .
//* You might also need to define a Dead letter Q.
//*
//* Locate and change all occurrences of the following strings
//*   .  MQS!!0 to the name of your MQSeries target library
//*
//* The STEPLIB must include the MQSeries libraries
//* if they are not installed in the LNKLST.
//*
//*
//**********************************************************************/
//*
//*################################################################
//*##The variables in the commands are shown unbold, and
//*##these values are substituted from the templates described
//*##in Table 3-1 on page 85, and Table 3-2 on page 86
//*################################################################
//*##PARM from reference B.11 in Table 3-2 on page 86
//*################################################################
//*

//ASNQMQD1 EXEC PGM=CSQUTIL,PARM='MQZ1'                <== B.11
//STEPLIB    DD DSN=MQZ1.USERAUTH,DISP=SHR
//           DD DSN=MQ531.SCSQANLE,DISP=SHR
//           DD DSN=MQ531.SCSQAUTH,DISP=SHR
//           DD DSN=CEE.SCEERUN,DISP=SHR
//SYSPRINT   DD SYSOUT=*
//SYSIN      DD *
  COMMAND DDNAME(CMDINP)
/*
//CMDINP    DD *

*
* Following are the definitions which are required to send data
* from the POKA capture instance to the STLA apply instance.
* Similar definitions are required for each capture/apply pair.
*
*

*Local AdminQ from reference B.17 in Table 3-2 on page 86
DEFINE REPLACE                                                    +
       QLOCAL(QREP.POKA.ADMINQ)                                   +
       DESCR('LOCAL DEFN OF ADMINQ FOR POKA CAPTURE')             +
```

```
        PUT(ENABLED)                                                    +
        GET(ENABLED)                                                    +
        SHARE                                                           +
        DEFSOPT(SHARED)                                                 +
        DEFPSIST(YES)

*Local RestartQ from reference B.16 in Table 3-2 on page 86
DEFINE REPLACE                                                          +
        QLOCAL(QREP.POKA.RESTARTQ)                                      +
        DESCR('LOCAL DEFN OF RESTART FOR POKA CAPTURE')                 +
        PUT(ENABLED)                                                    +
        GET(ENABLED)                                                    +
        SHARE                                                           +
        DEFSOPT(SHARED)                                                 +
        DEFPSIST(YES)

*Remote SendQ QREMOTE from reference B.18 in Table 3-2 on page 86
*Remote SendQ XMITQ from reference B.18a in Table 3-2 on page 86
*Remote SendQ RNAME from reference B.18b in Table 3-2 on page 86
*Remote SendQ RQMNAME from reference B.18c in Table 3-2 on page 86
DEFINE REPLACE                                                          +
        QREMOTE(QREP.POKA.TO.STLA.SENDQ)                                +
        DESCR('REMOTE DEFN OF SEND QUEUE FROM POKA TO STLA')             +
        PUT(ENABLED)                                                    +
        XMITQ(MQS1XMIT)                                                 +
        RNAME(QREP.POKA.TO.STLA.RECVQ)                                  +
        RQMNAME(MQS1)                                                   +
        DEFPSIST(YES)


* Following are the definitions which are required to send data
* from an MQSeries queue manager named MQZ1 to a Queue manager
* named MQS1.  These definitions are required only once.
*
*Local TransmitQ QLOCAL from reference B.13 in Table 3-2 on page 86
*Local TransmitQ TRIGDATA from reference B.13a in Table 3-2 on page 86
*Local TransmitQ INITQ from reference B.13b in Table 3-2 on page 86
DEFINE REPLACE                                                          +
        QLOCAL(MQS1XMIT)                                                +
        DESCR('TRANSMISSION QUEUE TO MQS1')                             +
        USAGE(XMITQ)                                                    +
        PUT(ENABLED)                                                    +
        GET(ENABLED)                                                    +
        TRIGGER                                                         +
        TRIGTYPE(FIRST)                                                 +
        TRIGDATA(MQZ1.TO.MQS1)                                          +
        INITQ(SYSTEM.CHANNEL.INITQ)

*
```

```
* Note that a listener must be started, on the other queue
* manager, for the port number identified in the channel
* sender definition.
*
*Local SDR channel CHANNEL from reference B.14 in Table 3-2 on page 86
*Local SDR channel XMITQ from reference B.14b in Table 3-2 on page 86
*Local SDR channel TRPTYPE from reference B.14a in Table 3-2 on page 86
*Local SDR channel CONNAME from reference B.14c in Table 3-2 on page 86
DEFINE REPLACE                                                      +
       CHANNEL(MQZ1.TO.MQS1)                                        +
       CHLTYPE(SDR)                                                 +
       TRPTYPE(TCP)                                                 +
       DESCR('SENDER CHANNEL TO MQS1')                              +
       XMITQ(MQS1XMIT)                                              +
       CONNAME('9.30.132.94(1414)')

*Local RCV channel CHANNEL from reference B.15 in Table 3-2 on page 86
*Local RCV channel TRPTYPE from reference B.15a in Table 3-2 on page 86

DEFINE REPLACE                                                      +
       CHANNEL(MQS1.TO.MQZ1)                                        +
       CHLTYPE(RCVR)                                                +
       TRPTYPE(TCP)                                                 +
       DESCR('RECEIVER CHANNEL FROM MQS1')

DISPLAY QUEUE(*) ALL

*
* Following are the definitions which are required to receive data
* at the POKA apply instance from the STLA capture instance.
*
*Local SpillQ from reference B.20 in Table 3-2 on page 86
DEFINE REPLACE                                                      +
       QMODEL('IBMQREP.SPILL.MODELQ')                               +
       DEFSOPT(SHARED)                                              +
       MAXDEPTH(999,999,999)                                          +
       MSGDLVSQ(FIFO)                                               +
       DEFTYPE(PERMDYN)
*Remote AdminQ QREMOTE from reference B.21 in Table 3-2 on page 86
*Remote AdminQ XMITQ from reference B.21a in Table 3-2 on page 86
*Remote AdminQ RNAME from reference B.21 in Table 3-2 on page 86
*Remote AdminQ RQMNAME from reference B.21c in Table 3-2 on page 86
DEFINE REPLACE                                                      +
       QREMOTE(QREP.STLA.ADMINQ)                                    +
       DESCR('REMOTE DEFN OF ADMINQ FOR STLA CAPTURE')              +
       PUT(ENABLED)                                                 +
       XMITQ(MQS1XMIT)                                              +
       RNAME(QREP.STLA.ADMINQ)                                      +
       RQMNAME(MQS1)                                                +
```

```
      DEFPSIST(YES)

*Local ReceiveQ QLOCAL from reference B.19 in Table 3-2 on page 86
DEFINE REPLACE                                                     +
       QLOCAL(QREP.STLA.TO.POKA.RECVQ)                             +
       DESCR('LOCAL RECEIVE QUEUE - STLA TO POKA')                 +
       PUT(ENABLED)                                                +
       GET(ENABLED)                                                +
       SHARE                                                       +
       DEFSOPT(SHARED)                                             +
       DEFPSIST(YES)                                               +
       INDXTYPE(MSGID)


DISPLAY QUEUE(*) ALL
//*******************************************************************/
//* Starting channels                                               */
//* For the remote set up, before you start replication you need to   */
//* start WebSphere MQ interchannel communication.                  */
//* First make sure that the queue manager is running, then start the */
//* listener and then start the channel:                            */
//* From MVS console, issue the command to start the qmgr           */
//* =MQZ1 start qmgr parm(MQZ1parm)                                 */
//* where =MQZ1 is the recognition character for the Queue manager  */
//* and MQZ1parm is the parms you want to use to start the qmgr     */
//*
*/
//* =MQZ1 sta lstr(1540) <<this starts the listener                 */
//* using tcpip protocol for queue manager CSQ1, port 1540>>        */
//*
*/
//* =MQZ1 start channel (MQZ1.TO.MQS1)                              */
//* <<this command starts the channel>>                             */
//*
*/
//* In some instances it might be necessary to start the channel    */
//*  initiator before starting the channel . If so use              */
//* =MQZ1 start chinit                                              */
//* Note: You need to use the same commands at the MQS1 server.     */
//*******************************************************************/
```

Note: The same listener port can be used on both host servers, but it may be easier to debug if one uses a different port on each.

Verify successful WebSphere MQ configuration

WebSphere MQ on z/OS does not provide a utility for inserting (via a PUT) and retrieving (via a GET) test messages on a queue to test the placement or movement of messages.

Therefore, one can only verify the correctness of the channel definitions by starting the channels (from ISPF MQ panels) on each queue manager and then verifying that they enter the RUN status.

Example 3-18 and Example 3-19 show MVS™ console commands that start the sender channel on STLABD1(MQS1) and WTSC53(MQZ1), respectively. Alternatively these channels can be started from the WebSphere MQ ISPF panels (not shown here). In z/OS, the channel start commands are usually included in the MQ manager startup procedure.

Example 3-18 Start the sender channel on STLABD1

```
//*##############################################################
//*##The variables in the commands are shown unbold, and
//*##these values are substituted from the templates described
//*##in Table 3-2 on page 86
//*#############################################################
//*##Local SDR channel CHANNEL from reference A.14 in Table 3-2 on page 86
//*##Queue manager from reference A.11 in Table 3-2 on page 86
//*## '+' is the z/OS recognition character for the Queue manager on STLABD1

+MQS1 start channel (MQS1.TO.MQZ1)
```

Example 3-19 Start the sender channel on WTSC53

```
//*##############################################################
//*##The variables in the commands are shown unbold, and
//*##these values are substituted from the templates described
//*##in Table 3-2 on page 86
//*#############################################################
//*##Local SDR channel CHANNEL from reference B.14 in Table 3-2 on page 86
//*##Queue manager from reference B.11 in Table 3-2 on page 86
//*## '=' is the z/OS recognition character for the Queue manager on WTSC53
=MQZ1 start channel (MQZ1.TO.MQS1)
```

3.5.6 Step 6: Configure and activate Q replication using GUI

In this section we document the step-by-step configuration of Q replication for the bidirectional replication topology in our fictitious company. Figure 3-5 expands “STEP 6: Configure & activate Q replication (GUI or commands)” (in Figure 3-2 on page 77) into a number of substeps involved in configuring Q replication. Each of these substeps is described in detail in the following subsections.

Very important: The Replication Center is typically used to configure and manage an SQL or Q replication environment because of its ease-of-use GUI interface. In order for a Replication Center client to be aware of the database servers that need to be defined as a first and second server in Figure 3-35 on page 142 in a replication environment, the Q Capture and Q Apply control tables *must* be created from the *same* Replication Center client. If the Q Capture and Q Apply control tables are created by ASNCLP scripts, or using *another* Replication Center client, then those database servers will *not* appear in the list of available servers for the first and second server selection. In such cases, you must catalog them in this Replication Center client using the process described in Appendix H, "Cataloging remote database servers" on page 881.

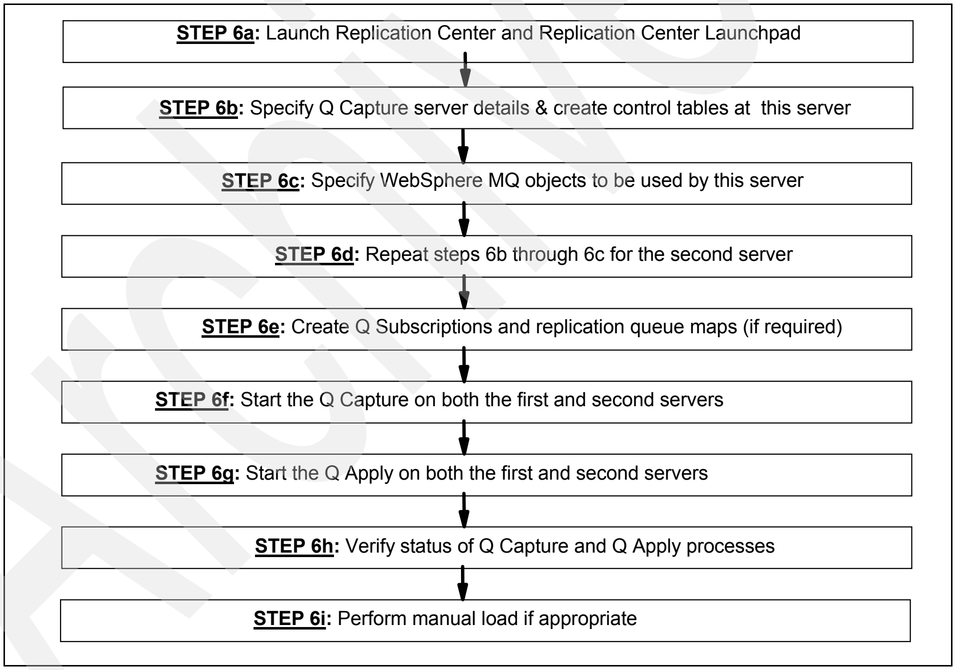


Figure 3-5 Overview of Q replication configuration steps

Step 6a: Launch Replication Center

Figure 3-6 shows the launching of the Replication Center from the DB2 Control Center by clicking the **Tools** tab and selecting **Replication Center**. This displays the Replication Center Launchpad screen shown in Figure 3-7.

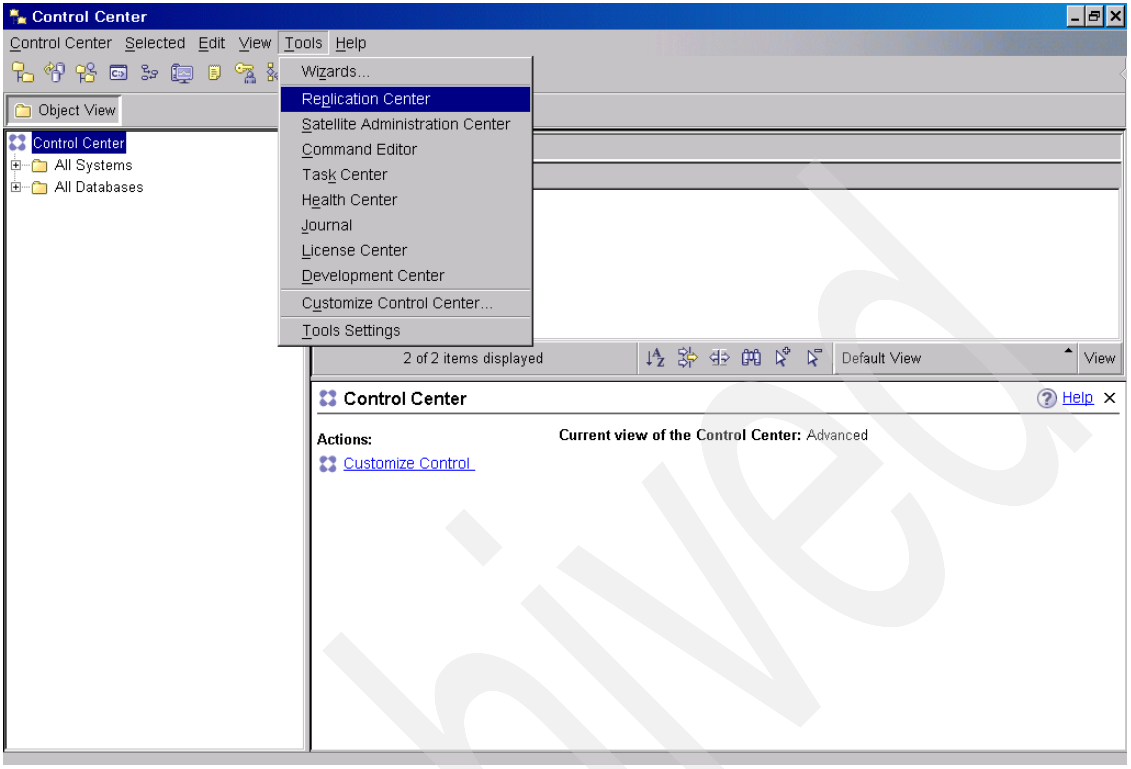


Figure 3-6 Launch Replication Center

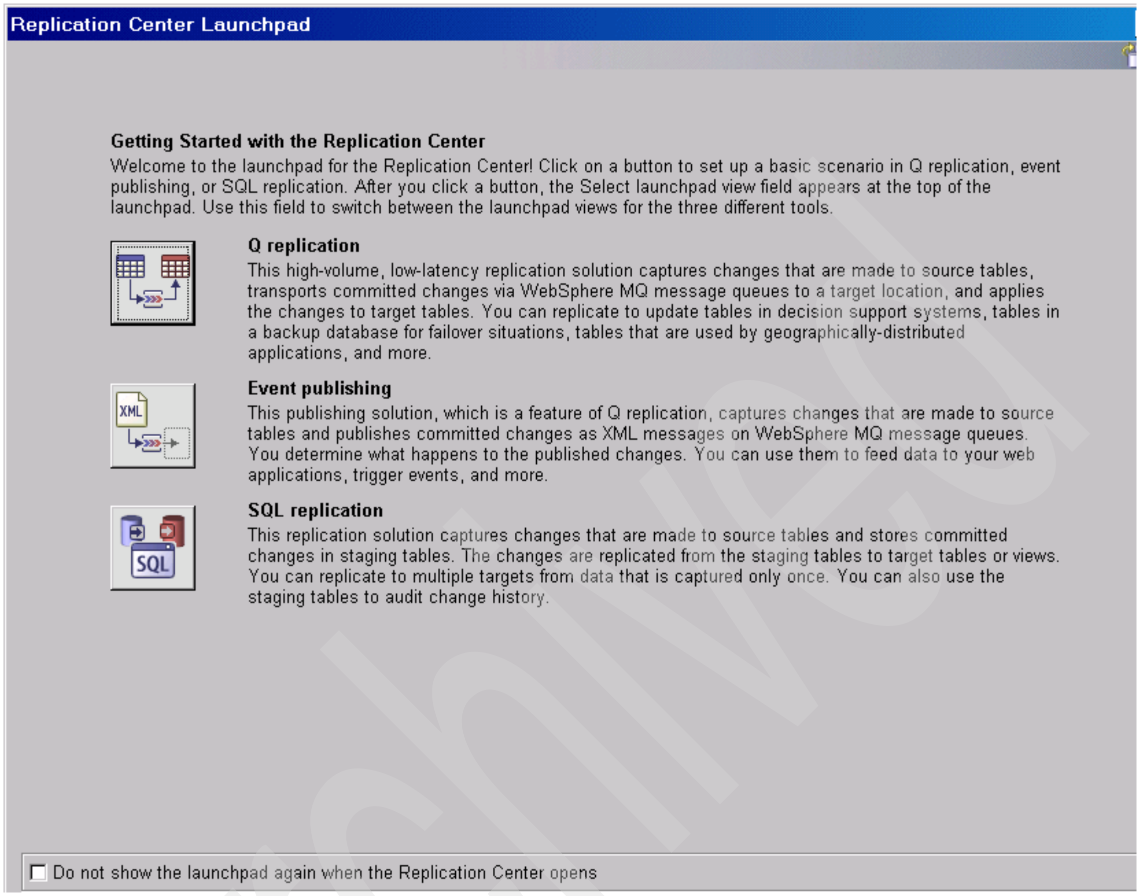


Figure 3-7 Replication Center Launchpad

> **Note:** Figure 3-7 can be skipped in the future by selecting the **Do not show this launchpad again when the Replication Center opens** option.

Step 6b: Specify Q Capture details

Click **Q replication** in Figure 3-7 to display Figure 3-8, which describes the five steps in setting up and activating Q replication, as follows:

1. Create Q Capture Control Tables.
2. Create Q Apply Control Tables.
3. Create a Q Subscription.
4. Start a Q Capture Program.
5. Start a Q Apply Program.

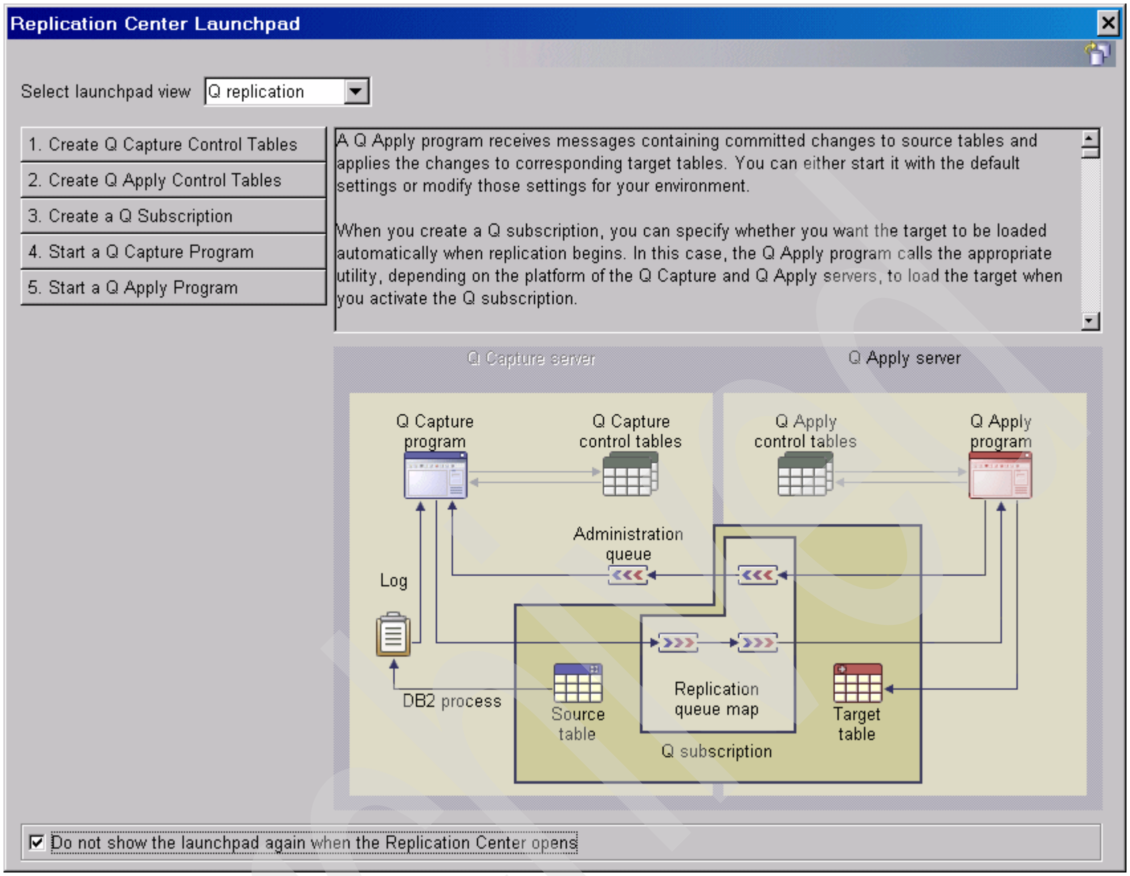


Figure 3-8 Five steps in setting up Q replication infrastructure

Click **Create Q Capture Control Tables** to display Figure 3-9, which enables the specification of details for the Q Capture infrastructure.

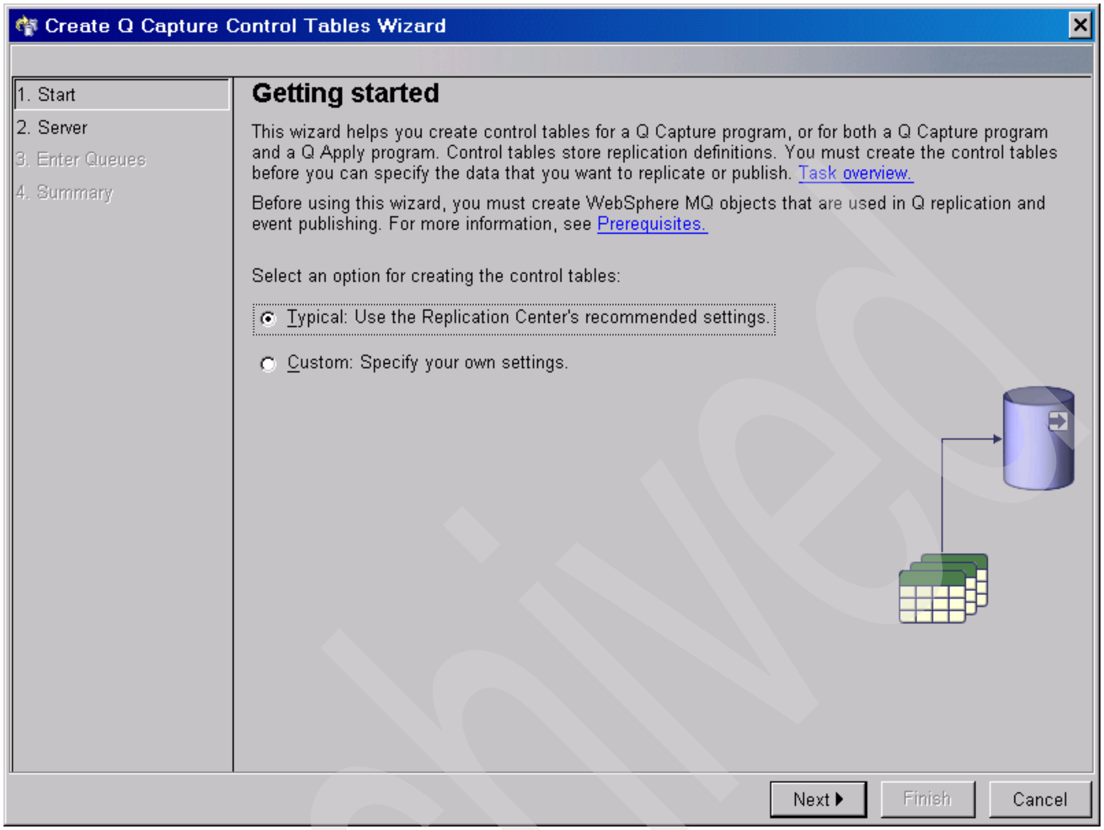


Figure 3-9 Getting started - Create Q Capture Control Tables Wizard

We selected the **Typical** setting and clicked **Next** to display Figure 3-10, where the Q Capture server details are specified.

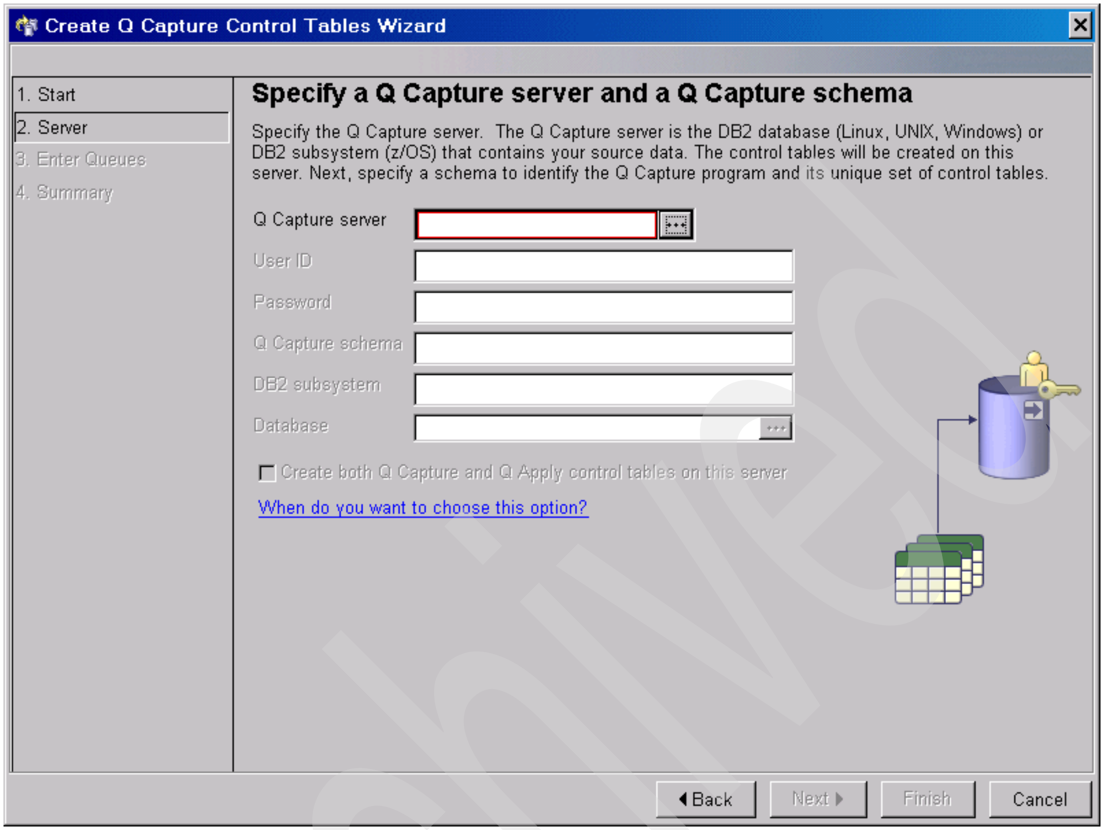


Figure 3-10 Server - Create Q Capture Control Tables Wizard

Click on the **...** tab to display the list of available servers, as shown in Figure 3-11.

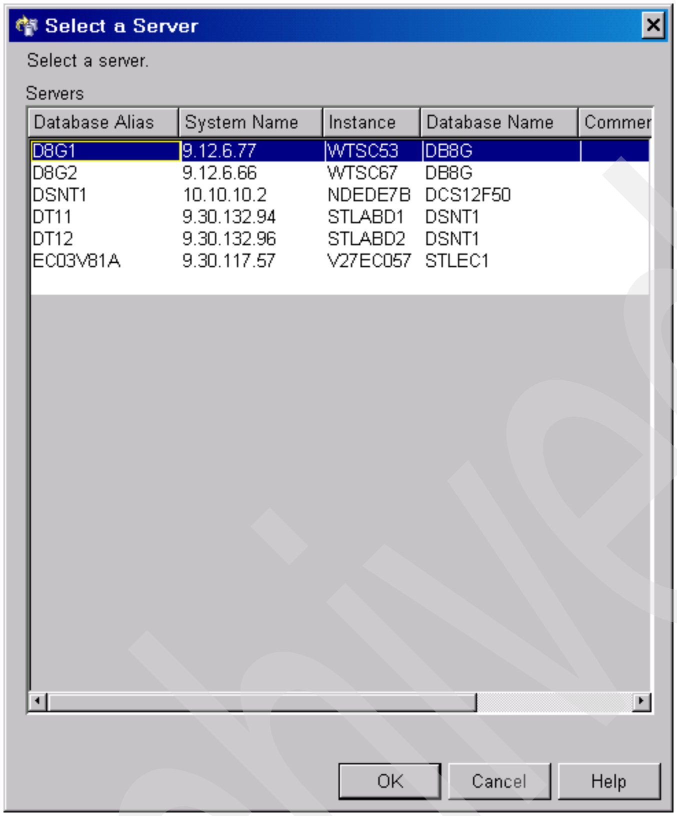


Figure 3-11 Select a Server

Select the **D8G1** (reference B.7 in Table 3-1 on page 85) database alias on instance WTSC53 as the database server for Q Capture, and click **OK** to display Figure 3-12 for providing Q Capture details.

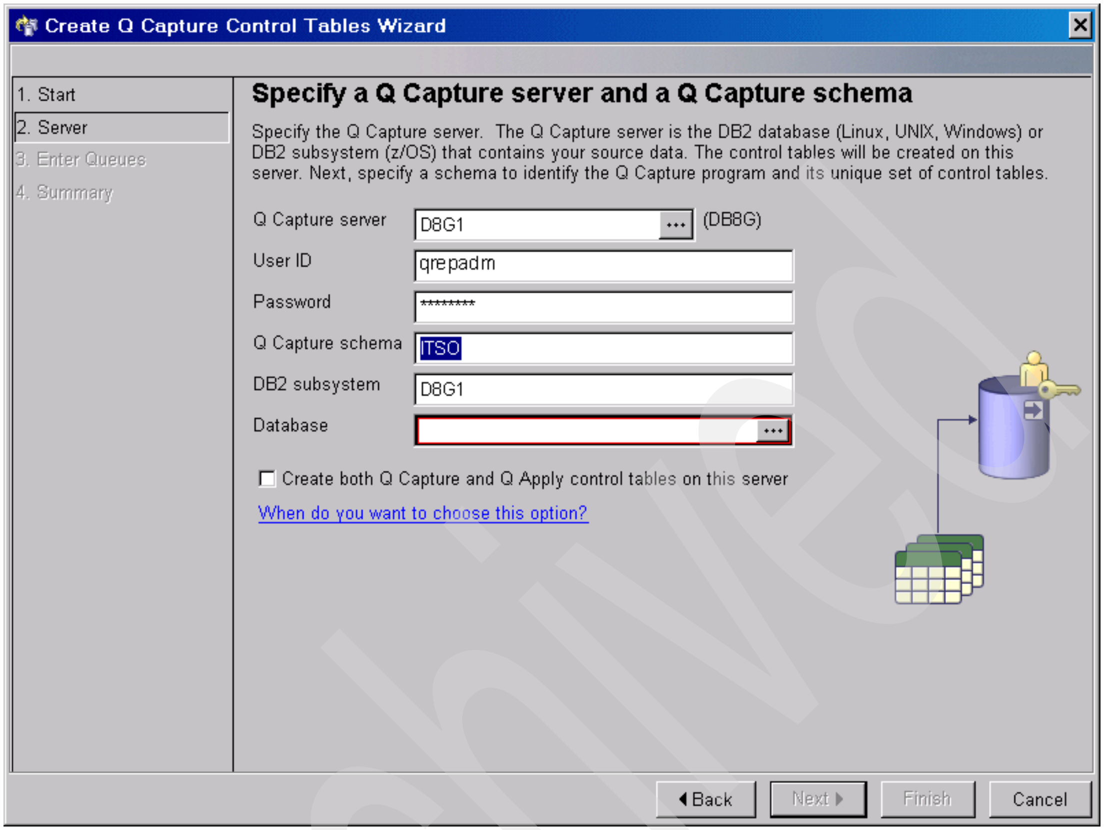


Figure 3-12 Q Capture server and schema details

Supply the Q Capture user ID and password “qrepadm/xxx” (reference B.9b in Table 3-2 on page 86) and Q Capture schema “ITSO” (reference B.22 in Table 3-3 on page 88). Click the **...** button to obtain a list of available databases, as shown in Figure 3-13.

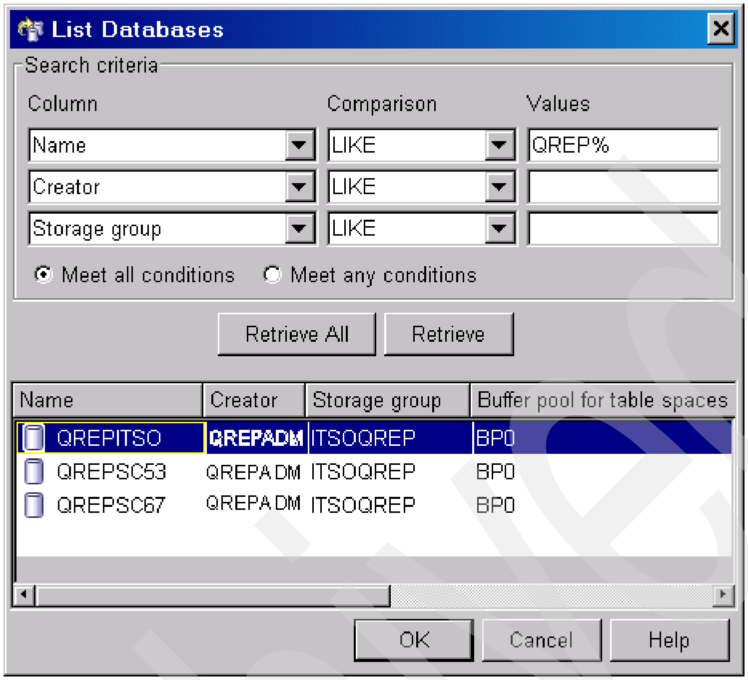


Figure 3-13 List Databases

Specify the selection criteria for the names of databases beginning with the characters QREP and click the **Retrieve** button to the list of qualifying databases as shown. Highlight the database named **QREPITSO** (which will contain the control tables—reference B.10 in Table 3-1 on page 85) and click **OK** to proceed to Figure 3-14.

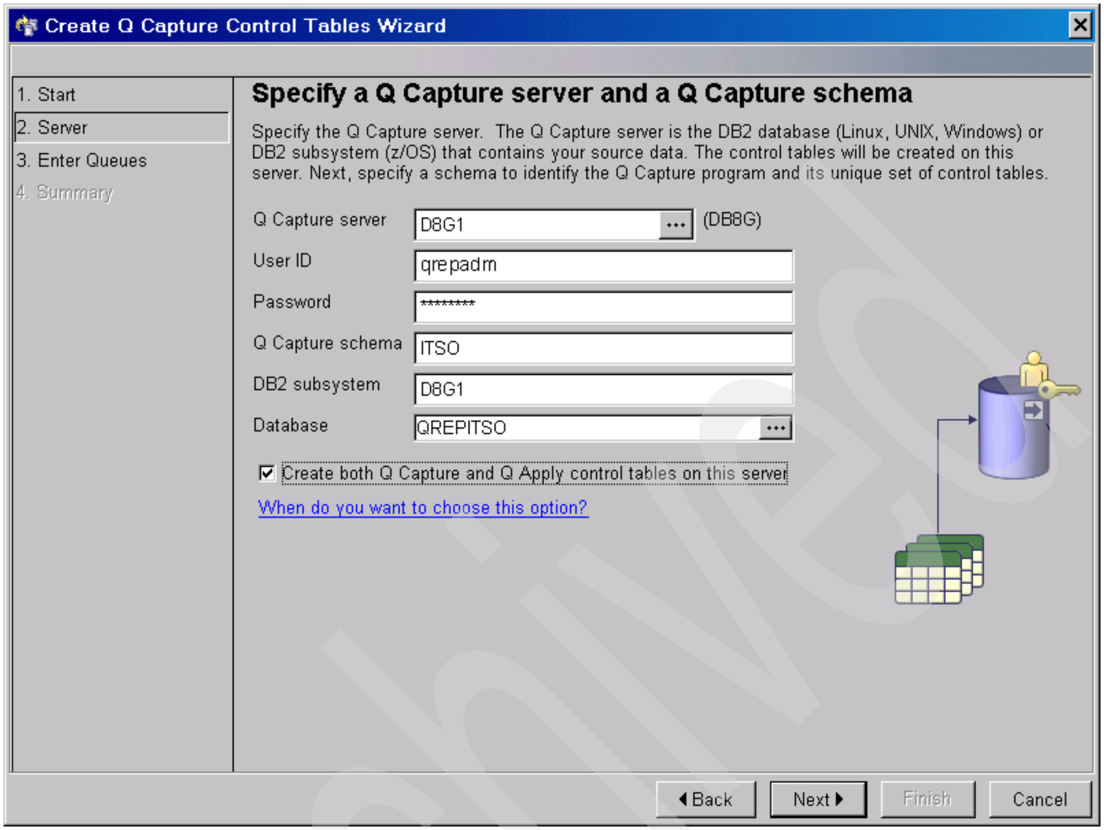


Figure 3-14 Q Capture server and schema details

Select the **Create both Q Capture and Q Apply control tables on this server** option in Figure 3-14.

Click **Next** to display Figure 3-15 for defining WebSphere MQ objects.

Step 6c: Specify WebSphere MQ objects

Supply WebSphere MQ queues for this server. This includes the Queue Manager "MQZ1" (reference B.11 in Table 3-3 on page 88), Administration queue "QREP.POKA.ADMINQ" (reference B.17 in Table 3-3 on page 88), and Restart queue "QREP.POKA.RESTARTQ" (reference B.16 in Table 3-3 on page 88).

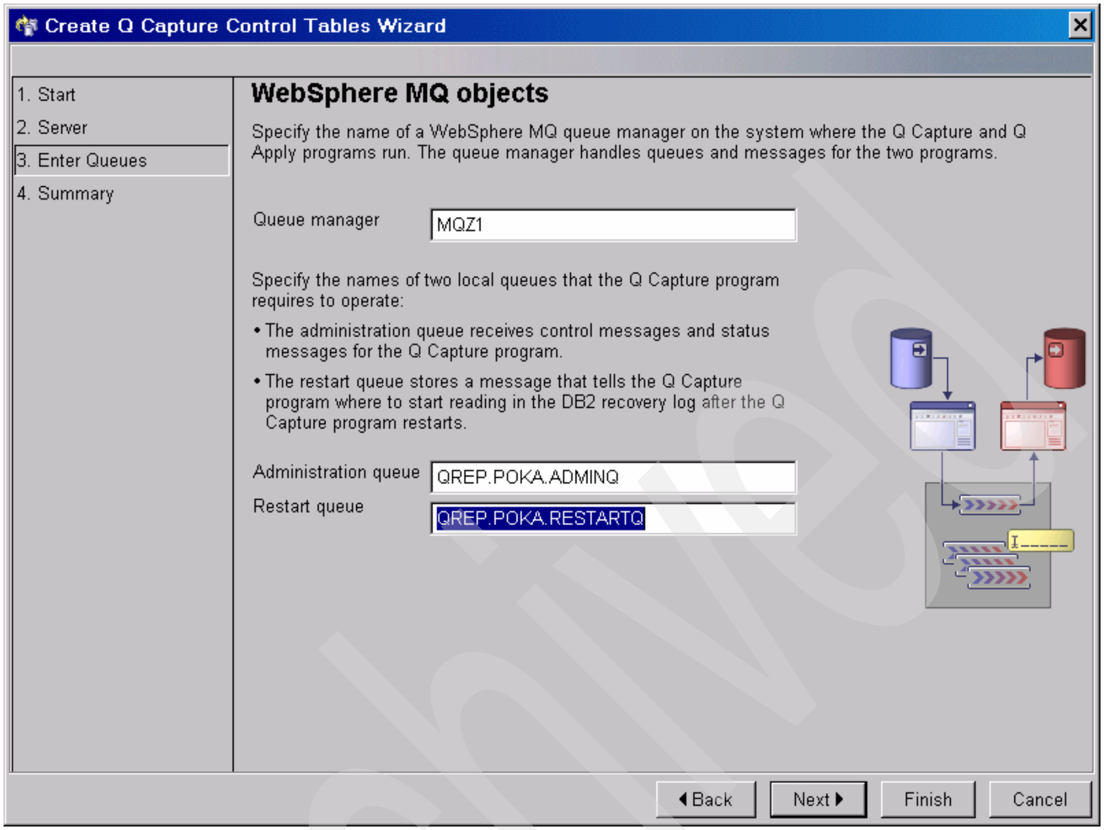


Figure 3-15 Queue information for Q Capture server

Click **Next** to continue to Figure 3-16, which summarizes the information provided in the previous screens.

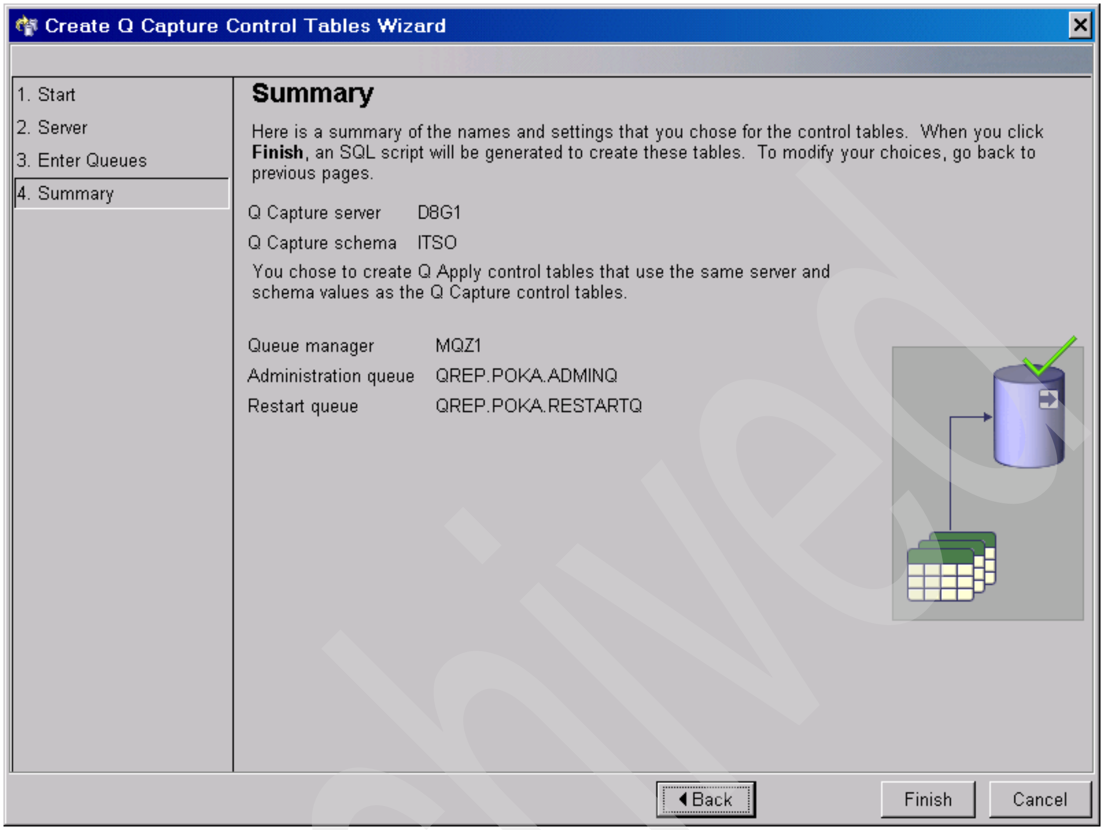


Figure 3-16 Q Capture server summary information

Verify accuracy of the information supplied, and click **Finish** to generate the scripts that need to be executed on the Q Capture server to define the specified objects, as shown in Figure 3-17.

Click the **Back** button to go back and correct any errors.

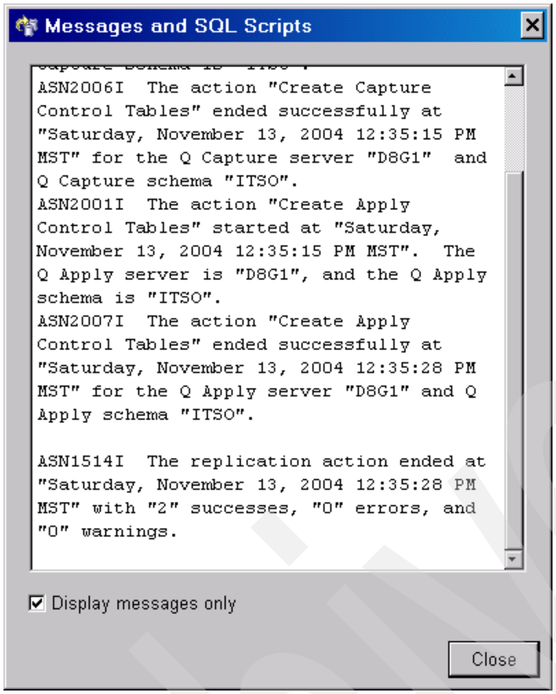


Figure 3-17 Messages and SQL Scripts screen for Q Capture server

Check that the errors are “0”. Click **Close** to display Figure 3-18, and decide whether to execute the script or save it for later execution.

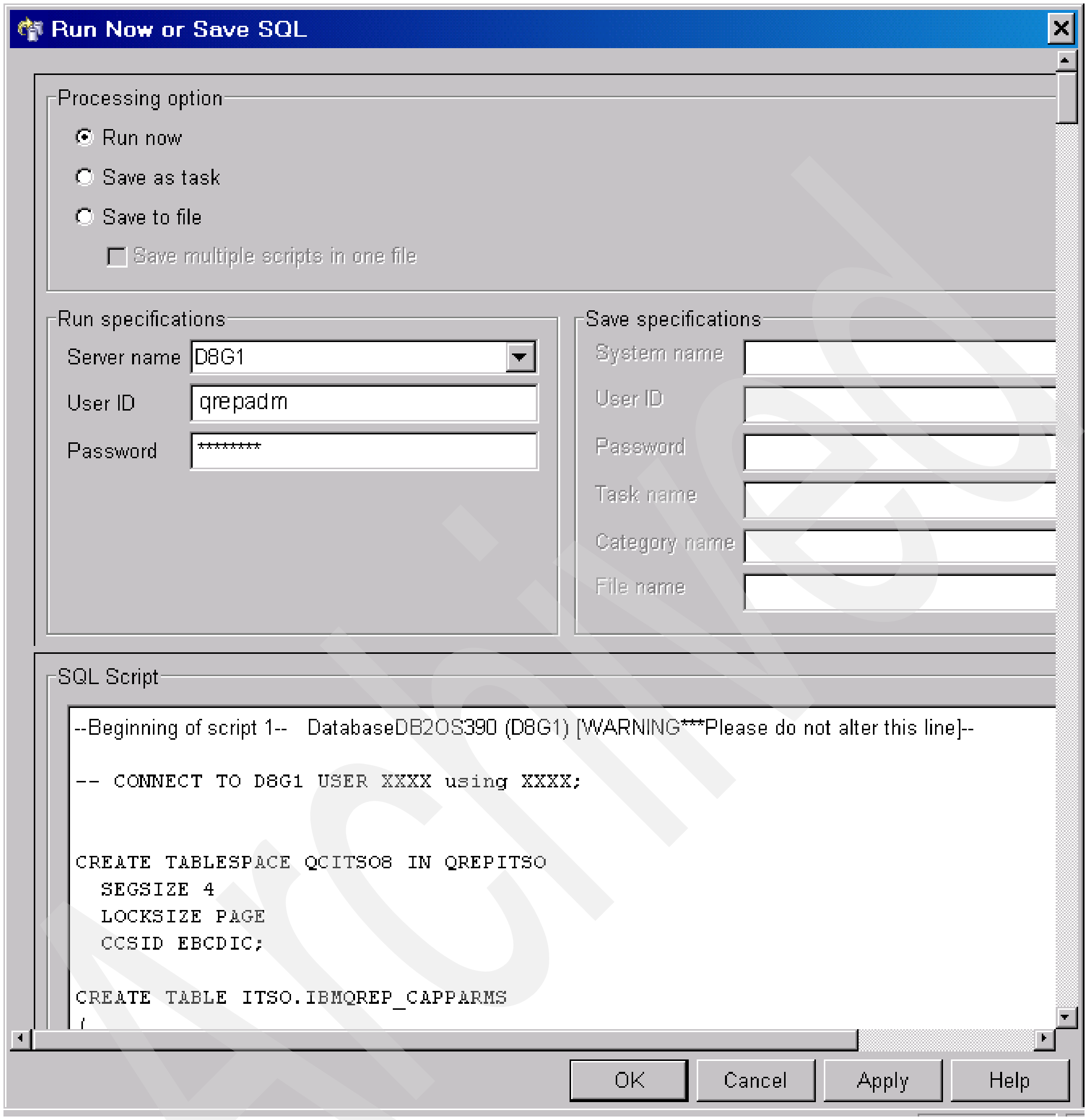


Figure 3-18 Run Now or Save SQL screen

> **Note:** At this point, one may choose to save the generated SQL to a file by selecting the **Save to file** option. Port it to MVS and execute it via SPUFI or DSNTEP2. If one chooses to save and execute the SQL scripts under MVS control, one must execute it before generating further SQL scripts, as the tables need to exist prior to the generation of the next SQL script.

However, we chose to execute the generated scripts immediately by selecting the **Run now** option.

Attention: Database Administration Server (DAS) is *not* required to execute these scripts from the workstation. DAS is only required to start and communicate with the Capture, Apply, and Monitor programs from the workstation.

Click **OK** to execute the SQL scripts and wait for the successful completion message, as shown in Figure 3-19.

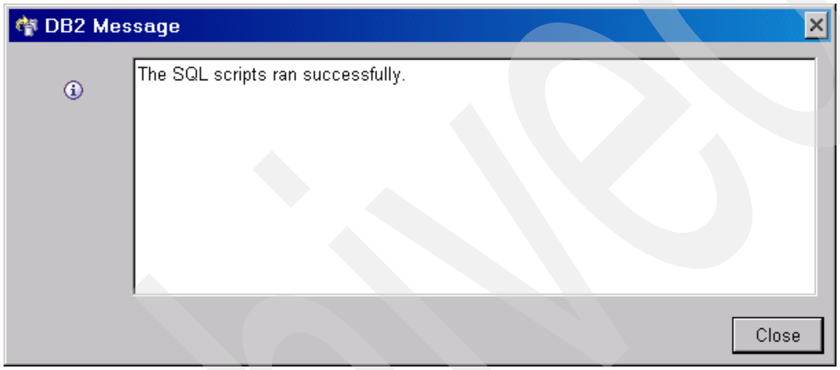


Figure 3-19 SQL scripts ran successfully message

Click **Close.** The control tables for both Q Capture and Q Apply on D8G1 (reference B.7 in template Table 3-1 on page 85) have been created.

Step 6d: Repeat steps 6b through 6c for second server

The same steps for the Q Capture server on WTSC53 need to be repeated for the STLABD1 using the appropriate reference information in Table 3-1 on page 85, Table 3-2 on page 86, and Table 3-3 on page 88, as shown in Figure 3-20 through Figure 3-31 on page 138.

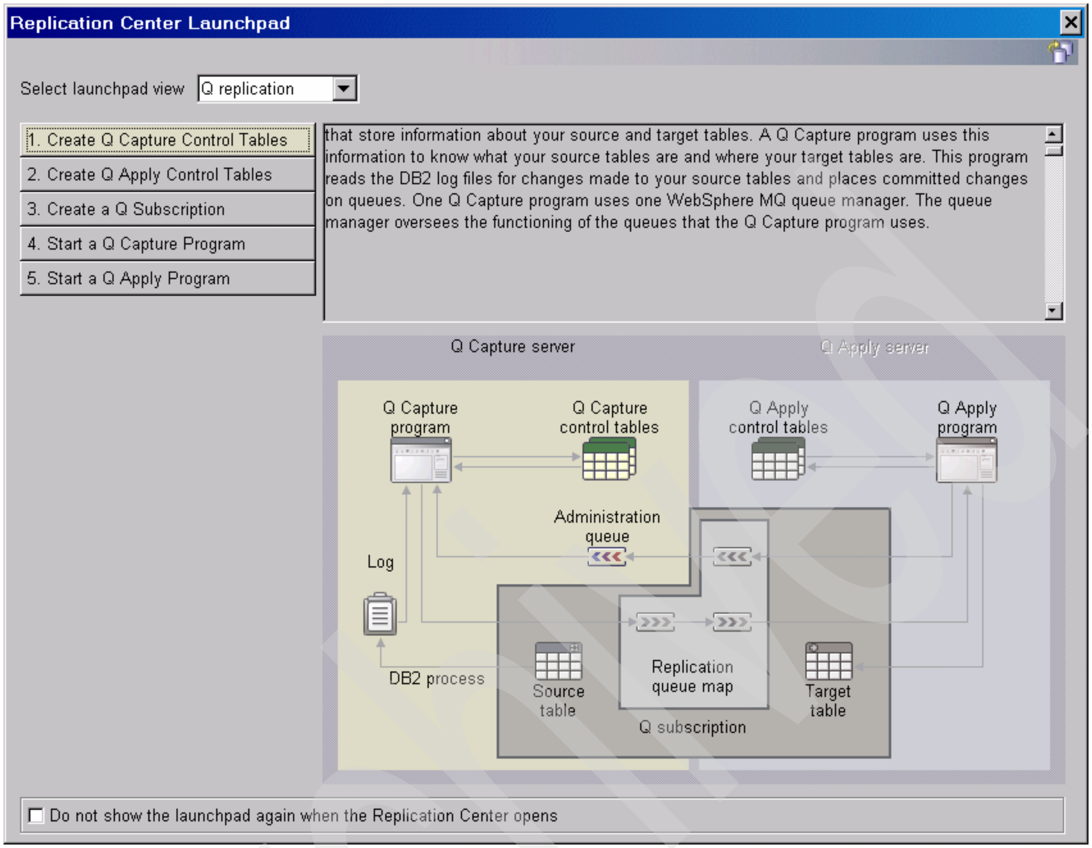


Figure 3-20 Five steps in setting up Q replication infrastructure

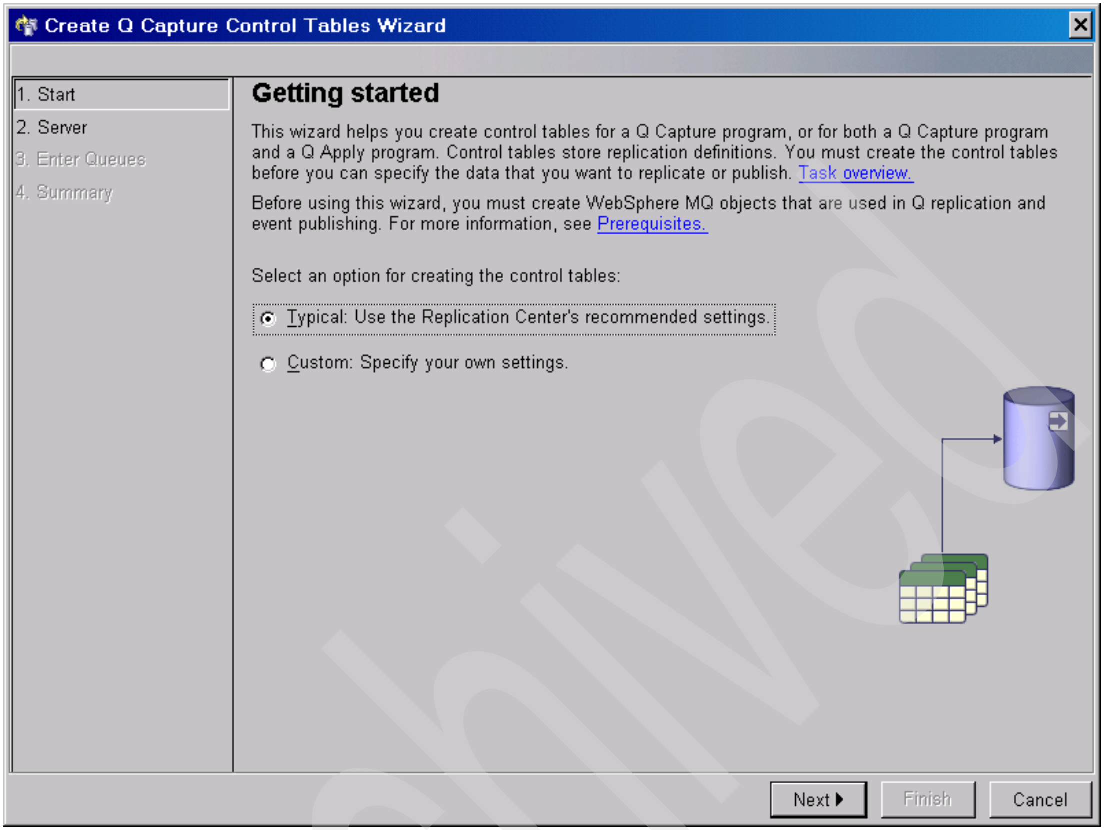


Figure 3-21 Getting started - Create Q Capture Control Tables Wizard

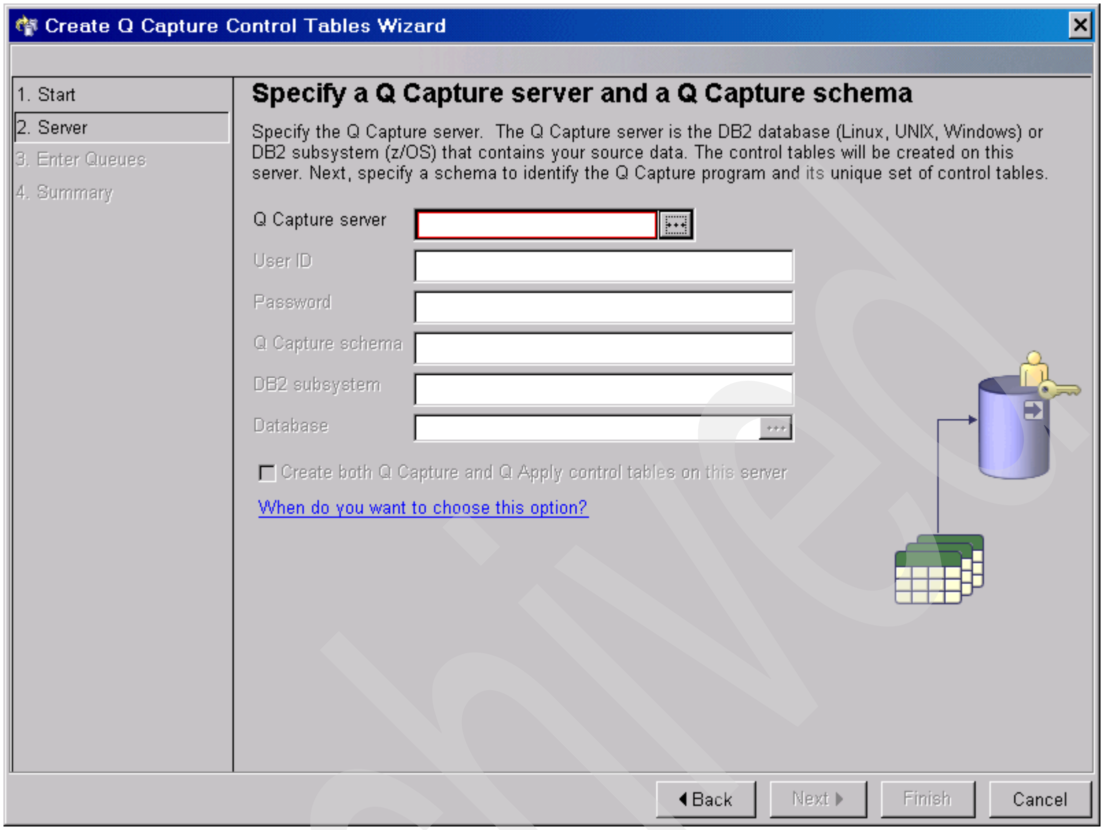


Figure 3-22 Server - Create Q Capture Control Tables Wizard

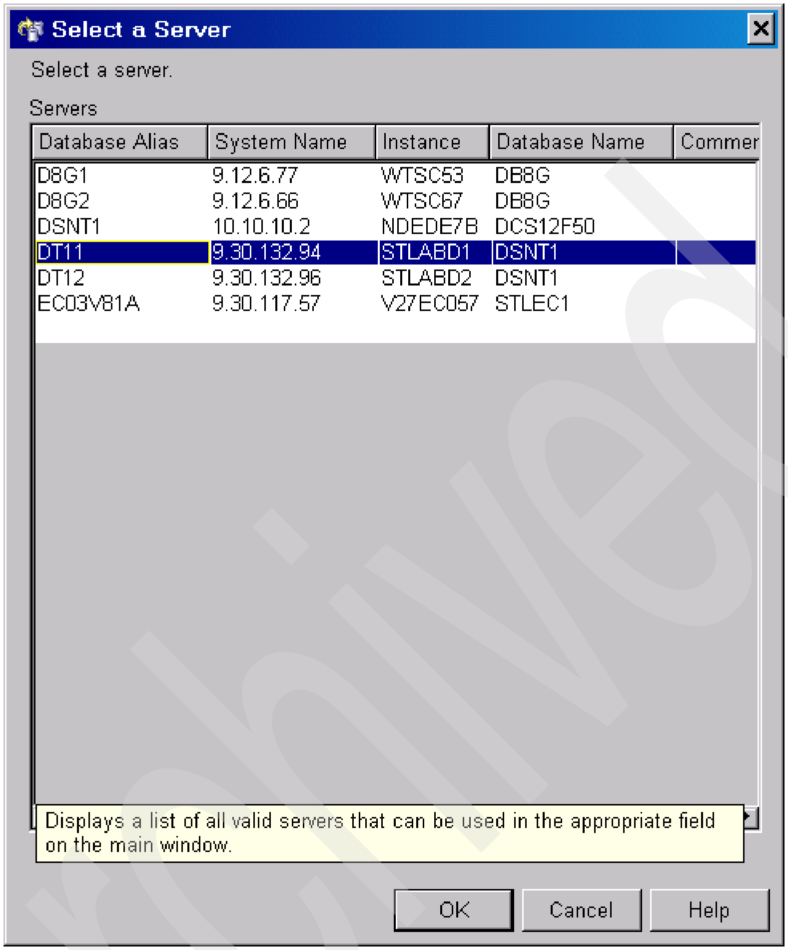


Figure 3-23 Select a Server

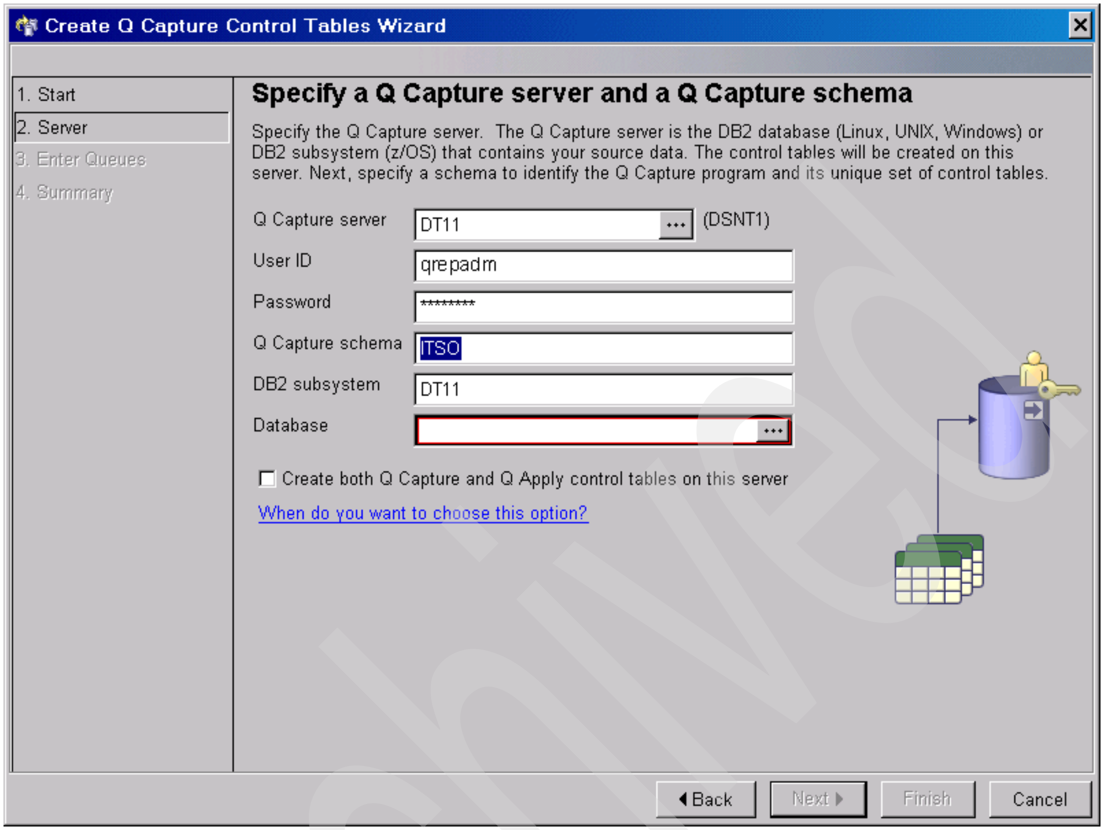


Figure 3-24 Q Capture server and schema details

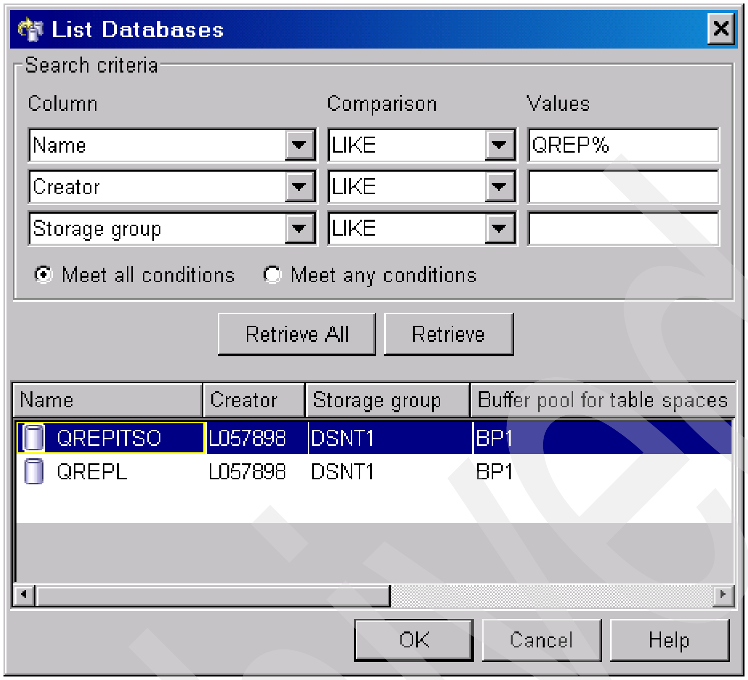


Figure 3-25 List Databases

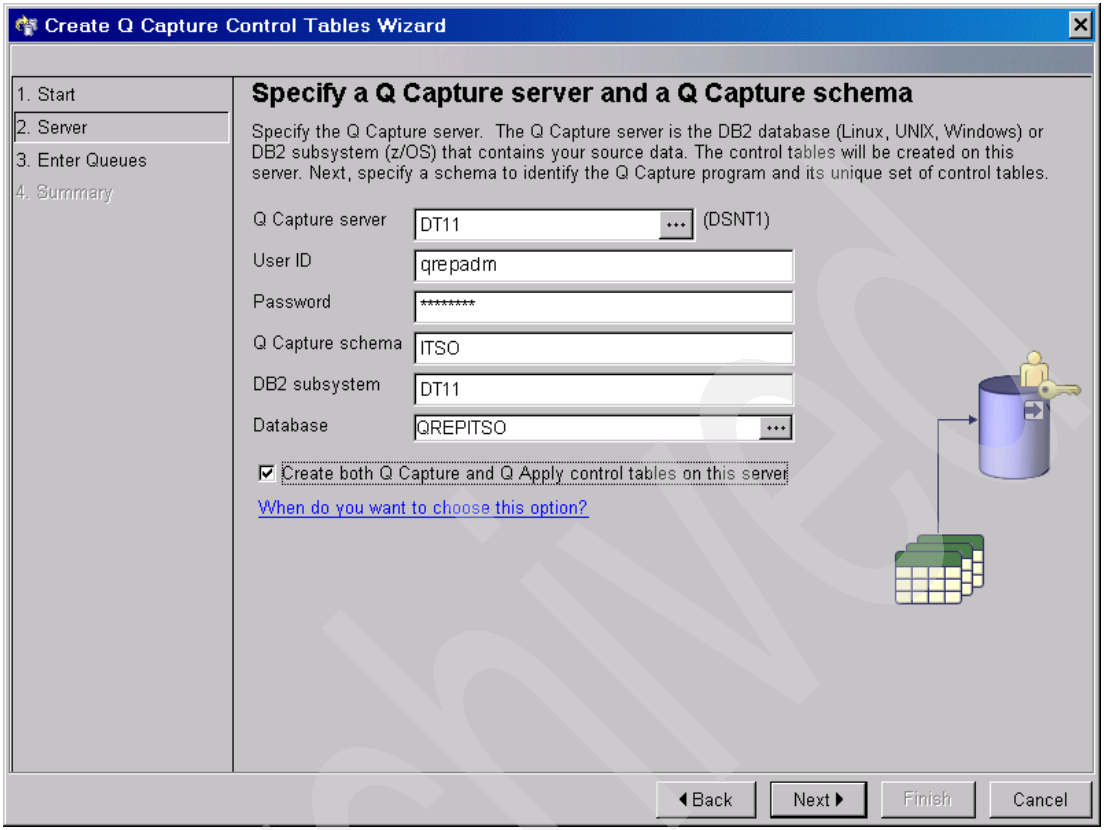


Figure 3-26 Q Capture server and schema details

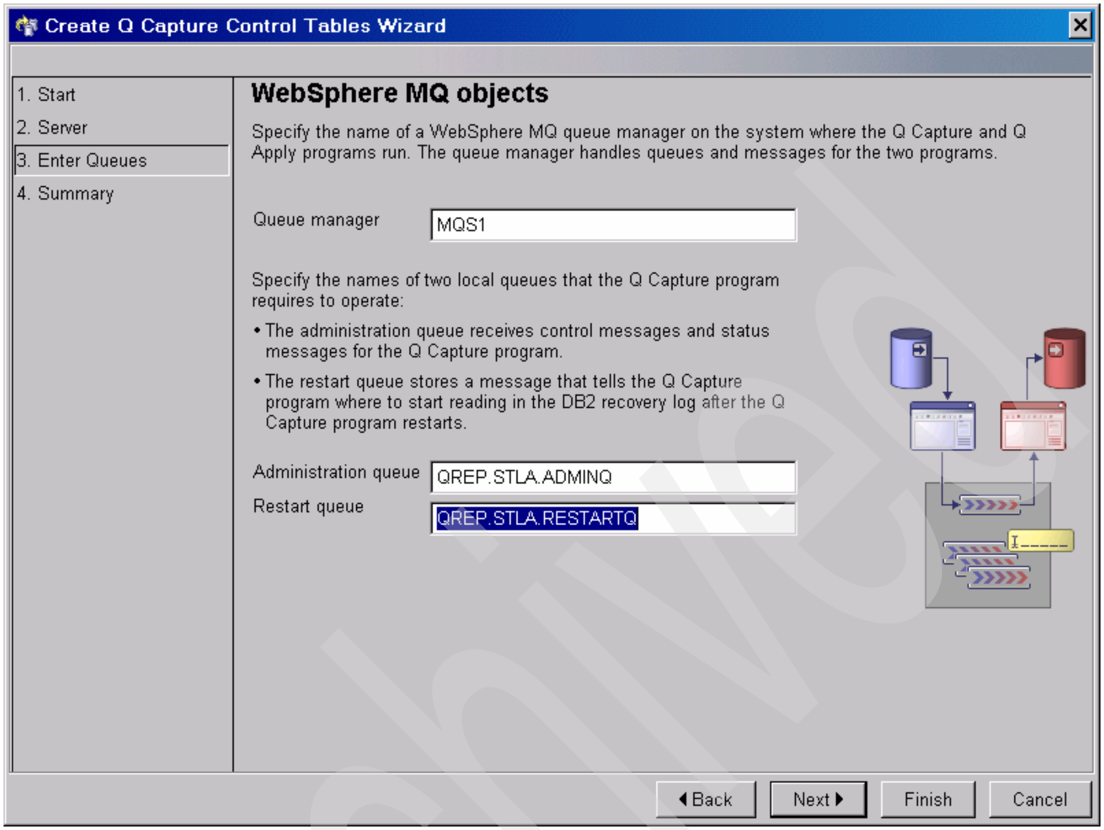


Figure 3-27 Queue information for Q Capture server

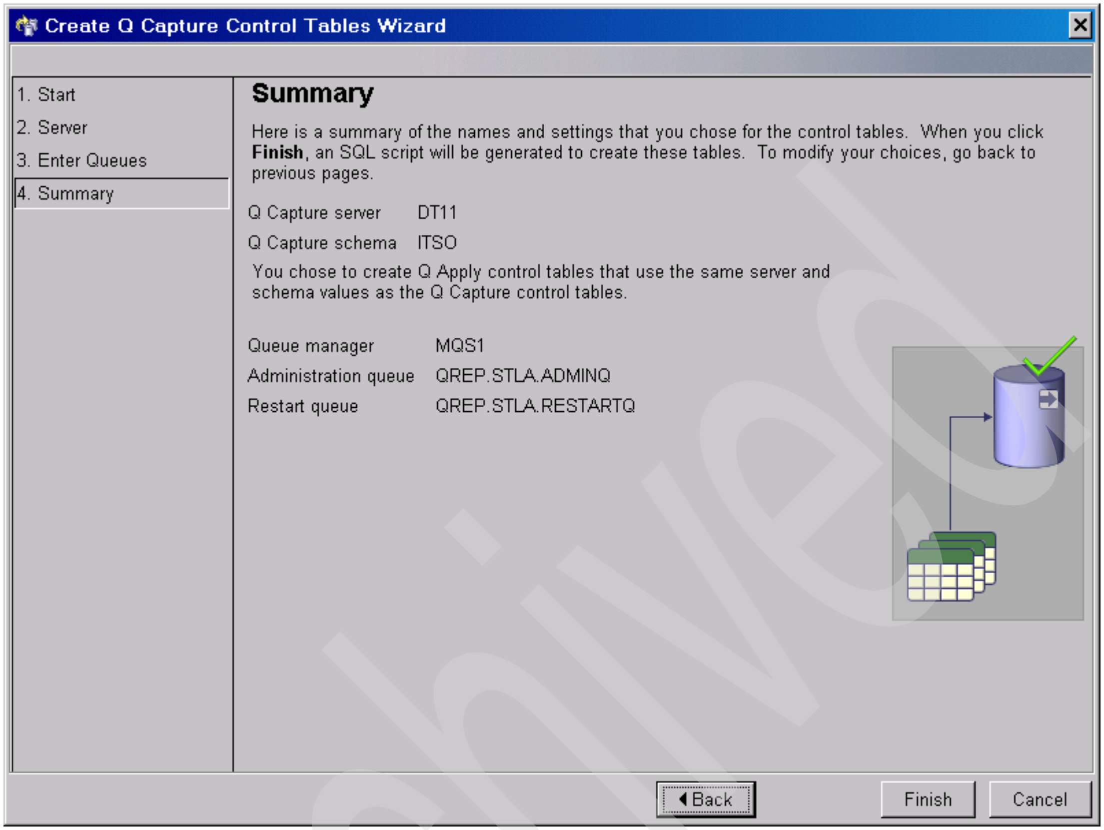


Figure 3-28 Q Capture server summary information

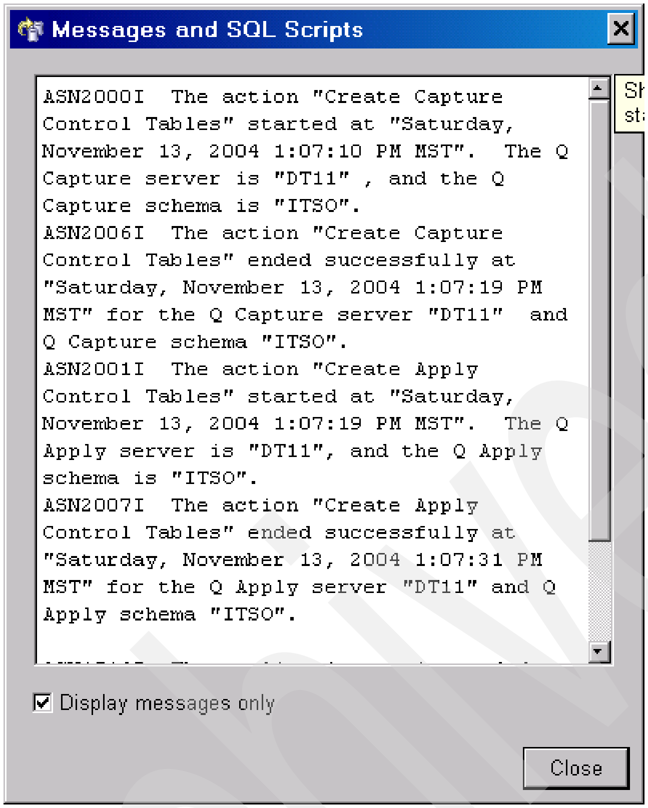


Figure 3-29 Messages and SQL Scripts screen for Q Capture server

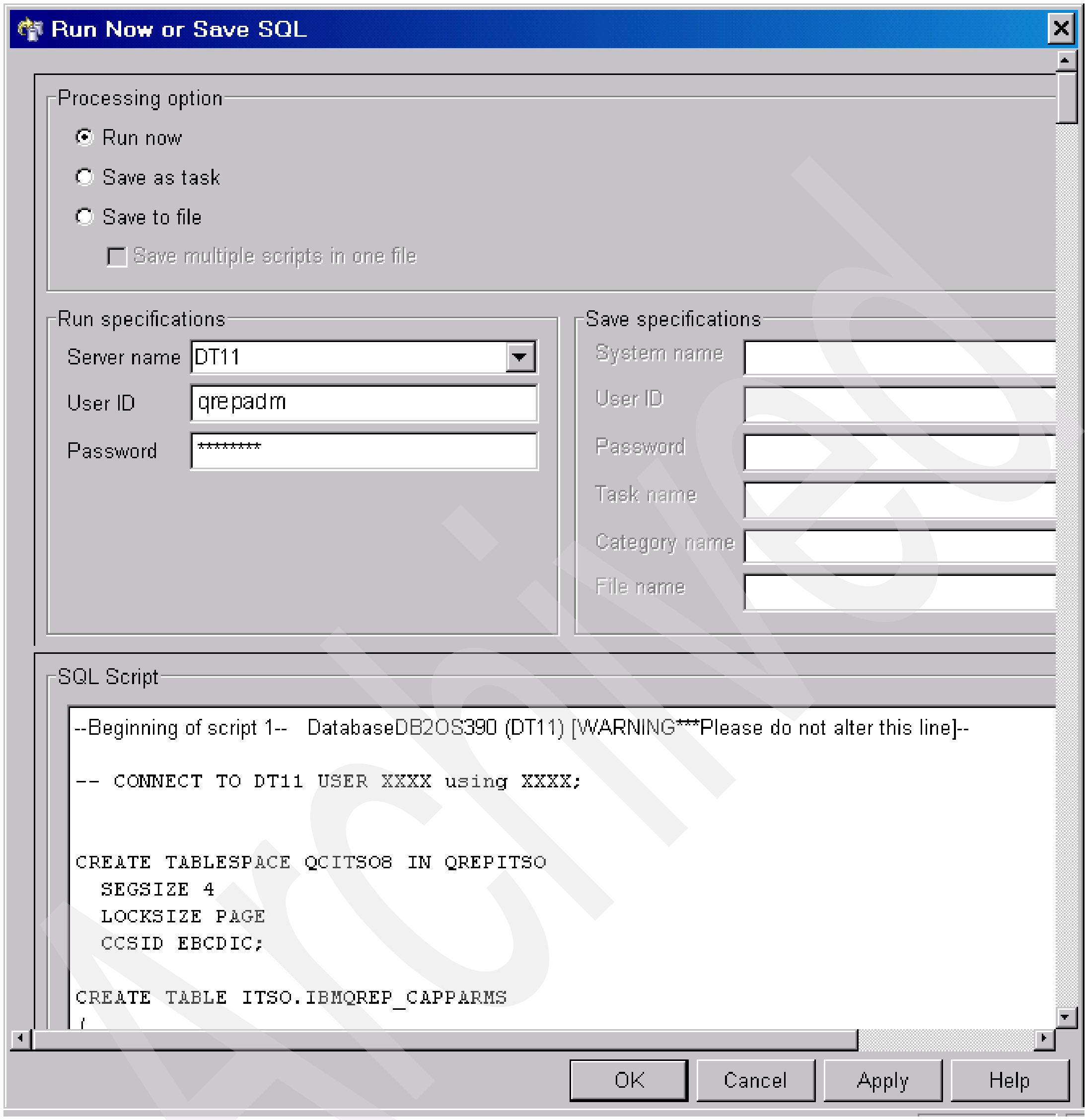


Figure 3-30 Run Now or Save SQL screen

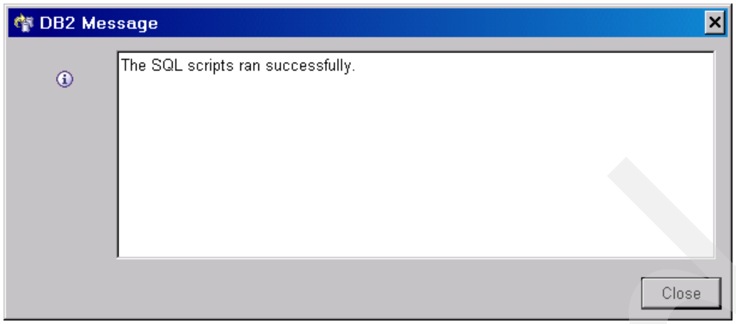


Figure 3-31 SQL scripts ran successfully message

Step 6e: Create Q Subscriptions and replication queue maps

After the Q Capture and Q Apply Control Tables have been created, the Q subscriptions and replication queue maps need to be created. Click the **Create a Q Subscription** option in Figure 3-32 to display Figure 3-33 to begin the process.

> **Note:** Since we created the Q Apply control tables earlier by selecting the **Create both Q Capture and Q Apply control tables on this server** in Figure 3-24 on page 131, we skipped the Create Q Apply Control Tables option in Figure 3-32.

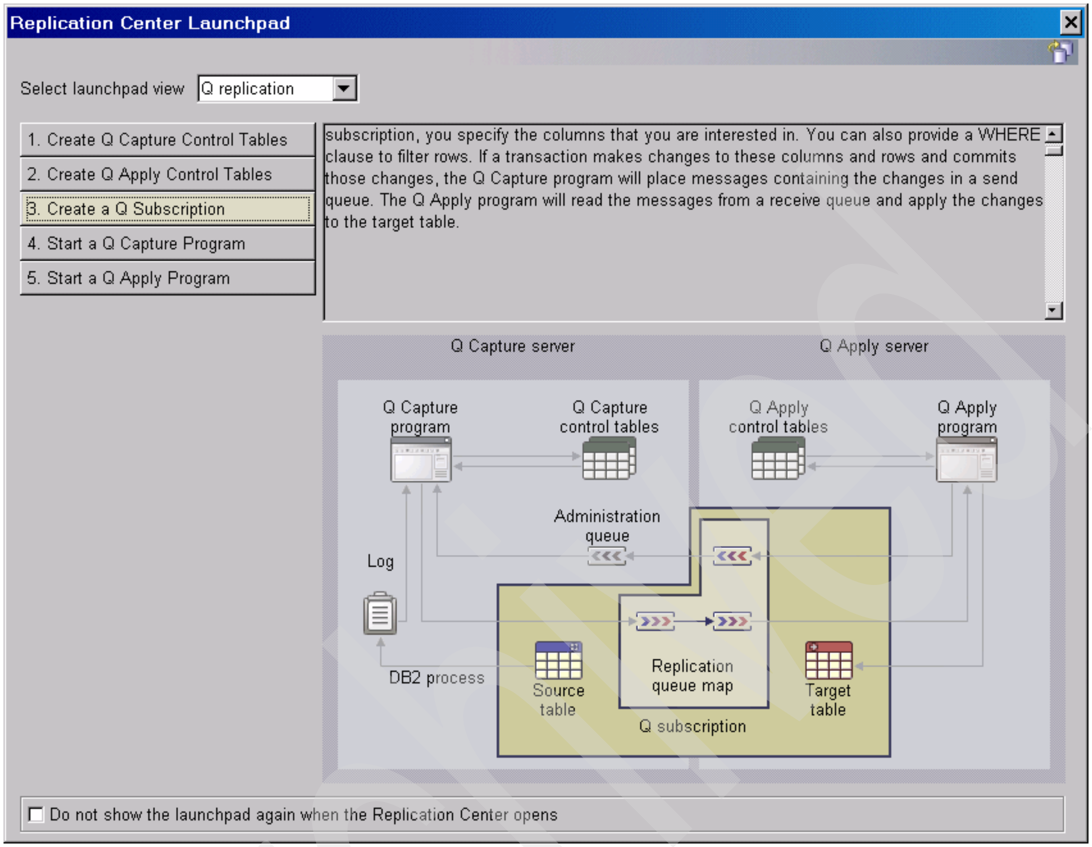


Figure 3-32 Five steps in setting up Q replication infrastructure

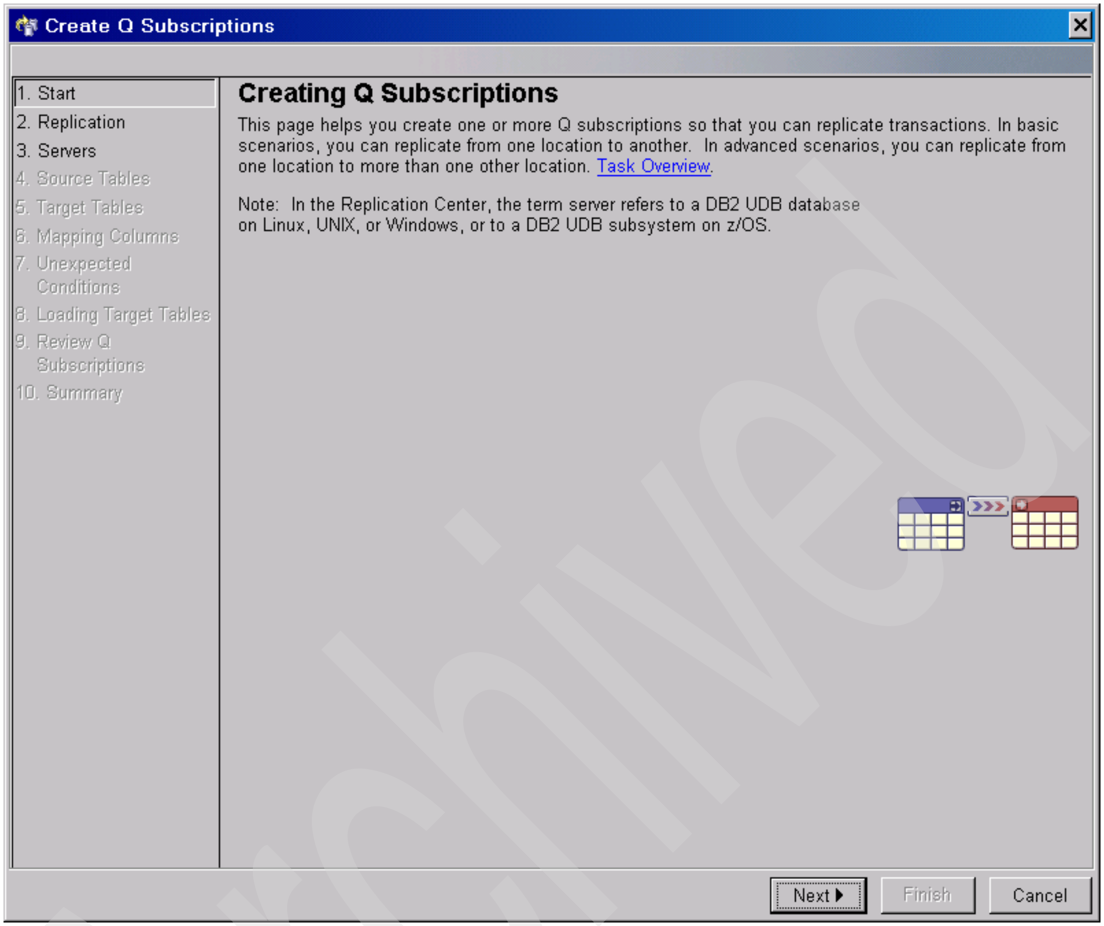


Figure 3-33 Create Q Subscriptions

Click **Next** to continue to Figure 3-34 to choose the type of replication topology desired.

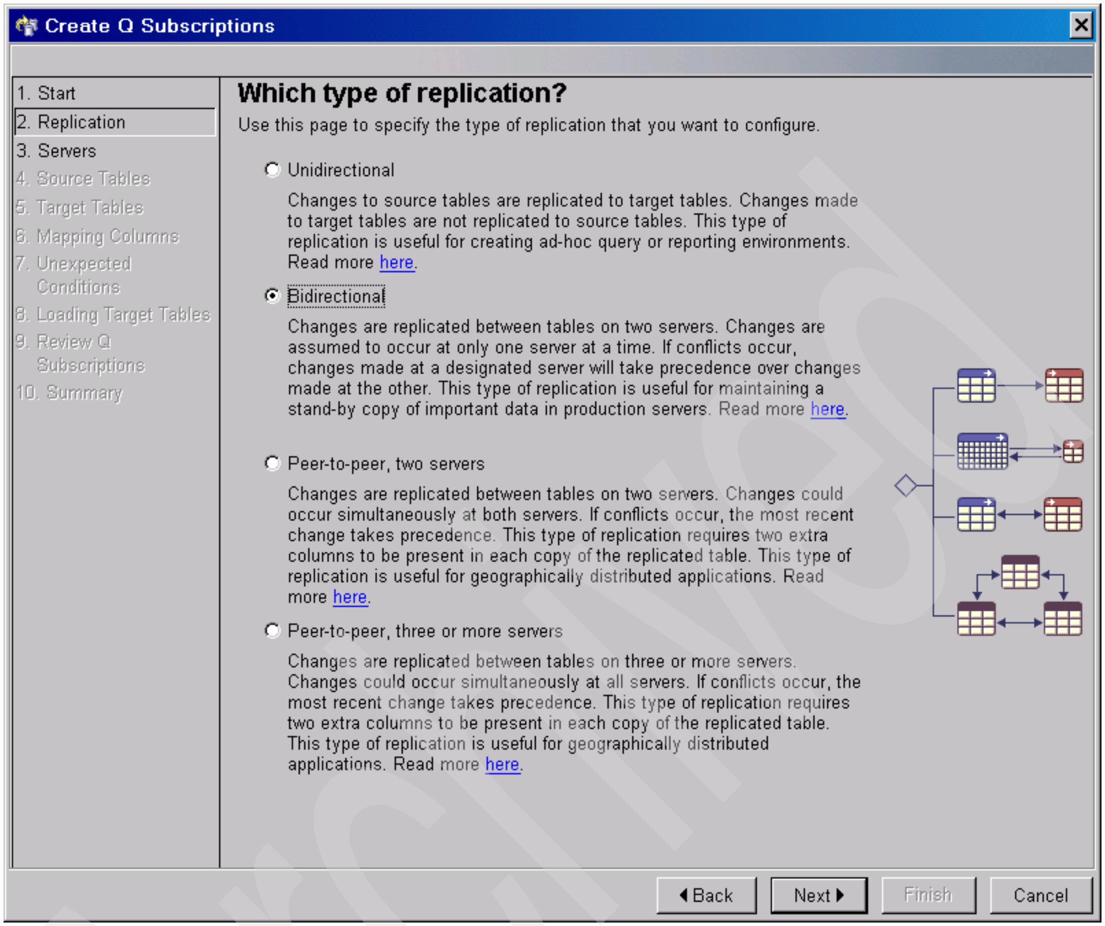


Figure 3-34 Which type of replication

Select the **Bidirectional** setting and click **Next** to provide details about the servers involved in the replication topology, as shown in Figure 3-35.

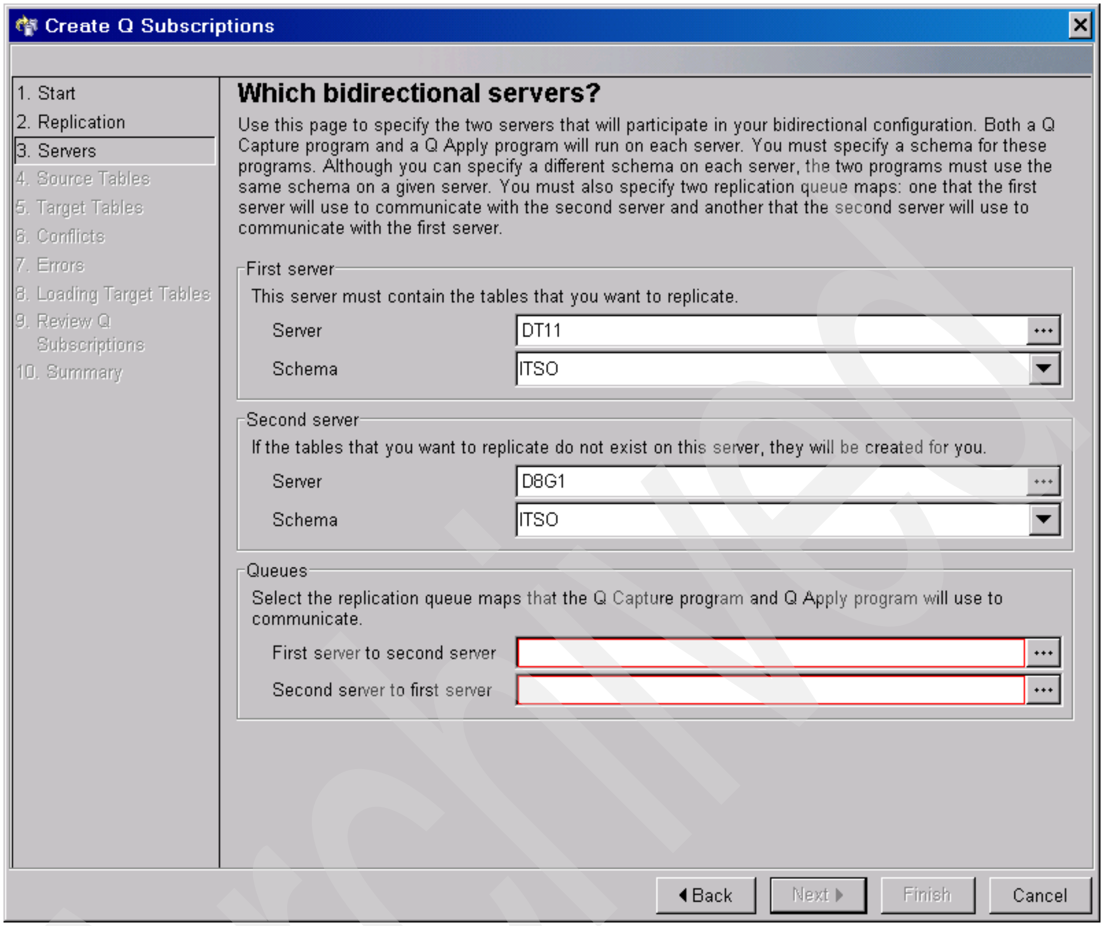


Figure 3-35 Which bidirectional servers

Supply the values for the First server (Server value “DT11” reference A.7a in Table 3-1 on page 85, Schema value “ITSO” reference A.22 in Table 3-3 on page 88) and Second server (Server value “D8G1” reference B.7a in Table 3-1 on page 85, Schema value “ITSO” reference B.22 in Table 3-3 on page 88).

Click the **...** button for the “First server to second server” to get a list of the available replication queue maps, as shown in Figure 3-36.

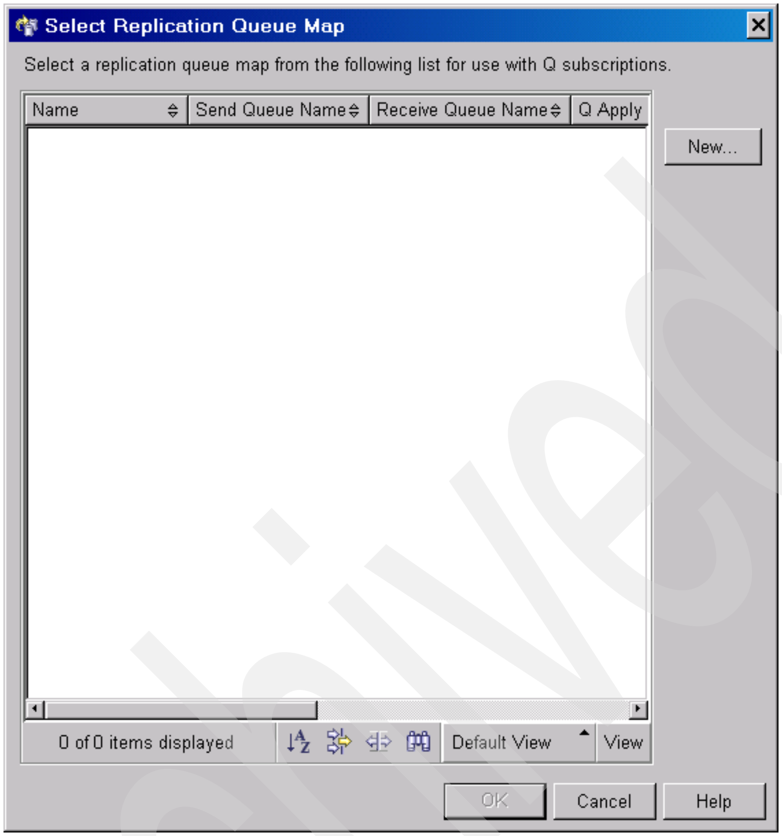


Figure 3-36 Select Replication Queue Map

Since no replication queue maps have been defined as yet, Figure 3-36 has no list of existing replication queue maps. A new replication queue map needs to be created by clicking the **New...** button. This displays Figure 3-37.

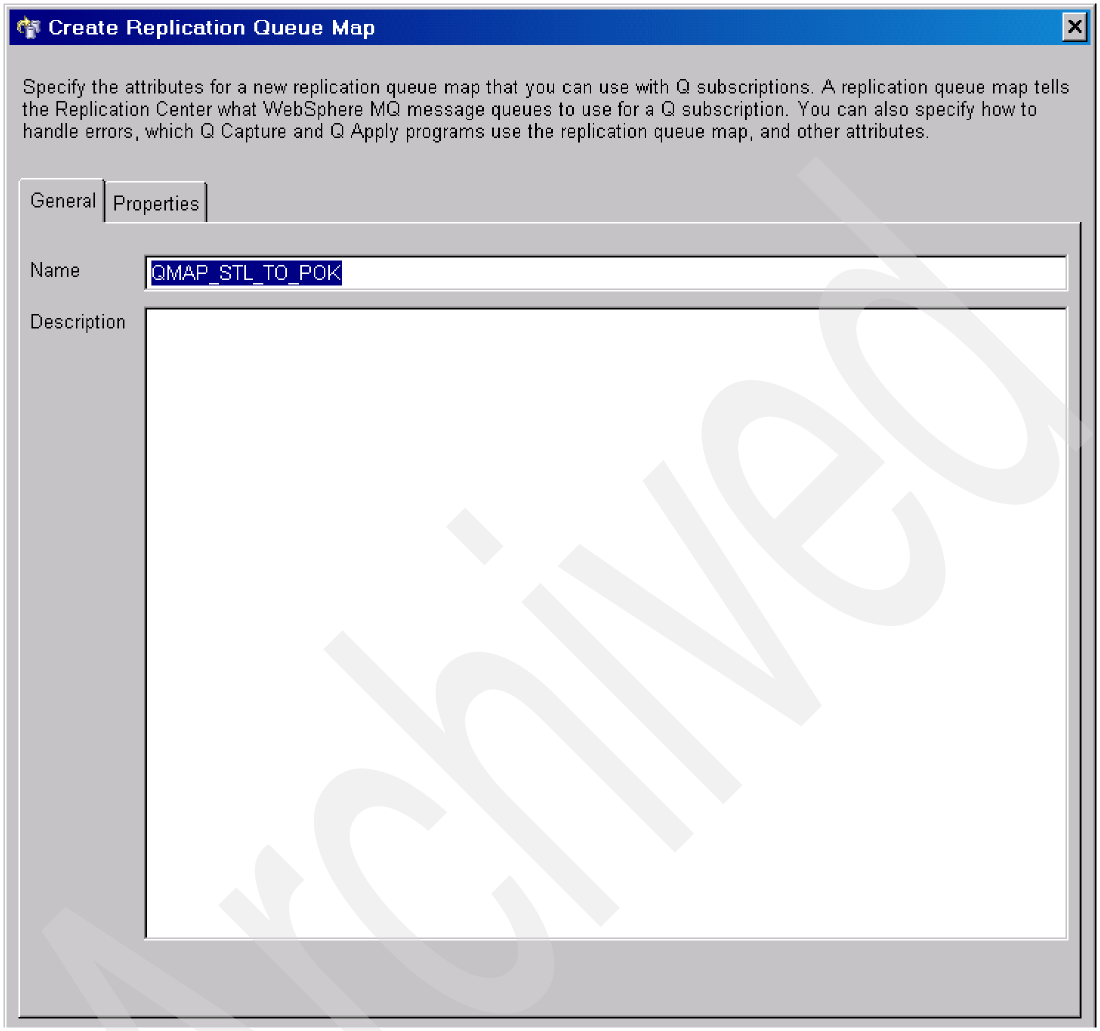


Figure 3-37   Create Replication Queue Map

Supply the Name “QMAP_STL_TO_POK” (reference A.27 in Table 3-3 on page 88), and then choose the **Properties** tab to display Figure 3-38.

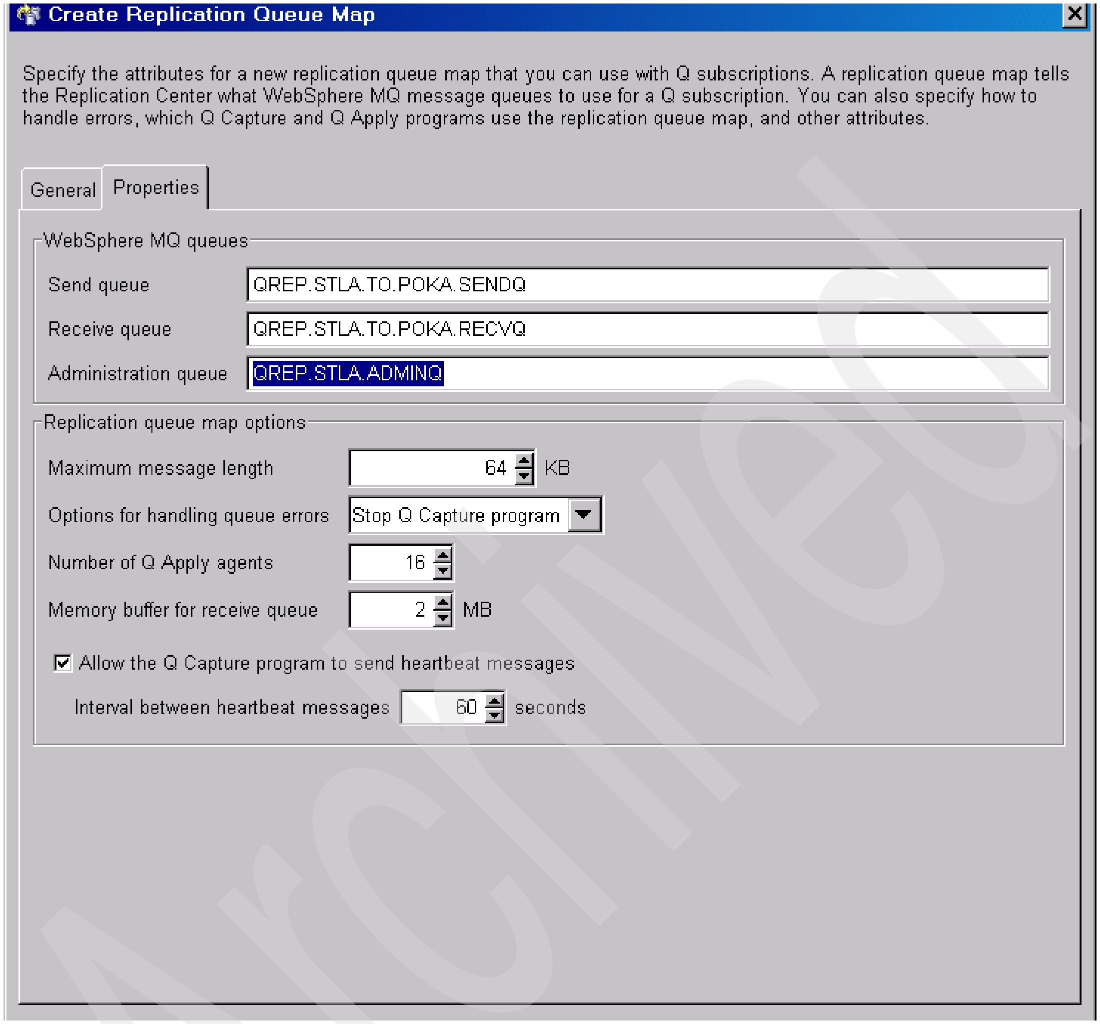


Figure 3-38 Create Replication Queue Map - Properties

Supply values for the Send queue ("QREP.STLA.TO.POKA.SENDQ" reference A.18 in Table 3-3 on page 88), Receive queue ("QREP.STLA.TO.POKA.RECVQ" reference B.19 in Table 3-3 on page 88), Administration queue ("QREP.STLA.ADMINQ" reference A.17 in Table 3-3 on page 88), and let other values default.

Click **OK** to complete the definition, and generate the scripts for creating the replication queue map objects on the servers, as shown in Figure 3-41.

Attention: In fixpak 9, which is due out in April 2005, the screens shown in Figure 3-37 and Figure 3-38 have been enhanced to appear as shown in Figure 3-39 and Figure 3-40, respectively. Please ignore the values shown therein, since the screens have been shown to only highlight the difference in the new screen layout.

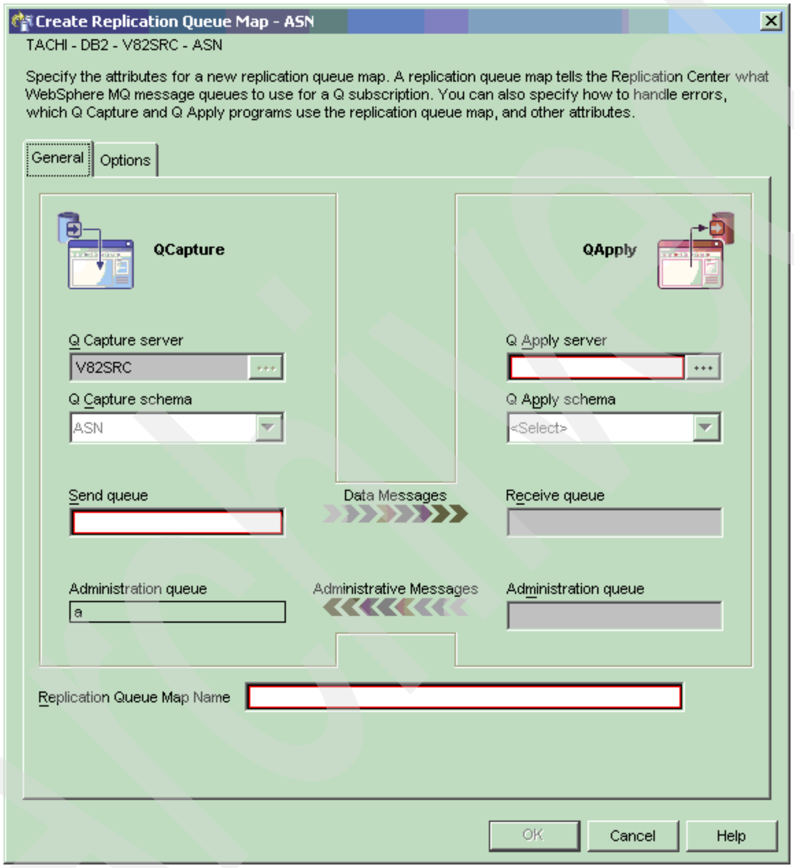


Figure 3-39 Fixpak 9 Create Replication Queue Map screen

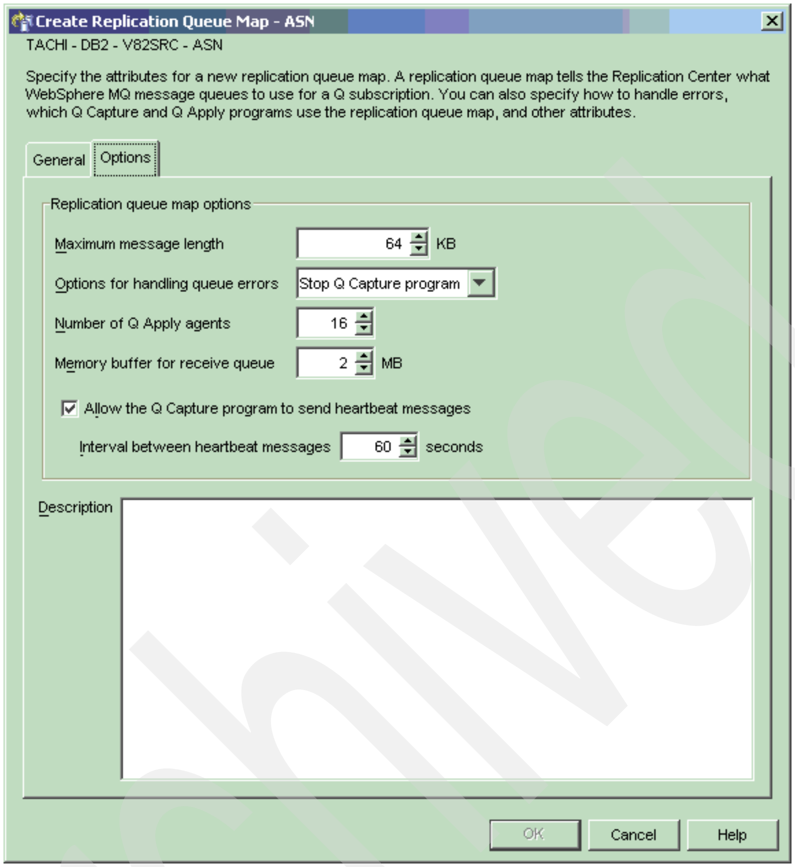


Figure 3-40 Fixpak 9 Create Replication Queue Map - Options screen

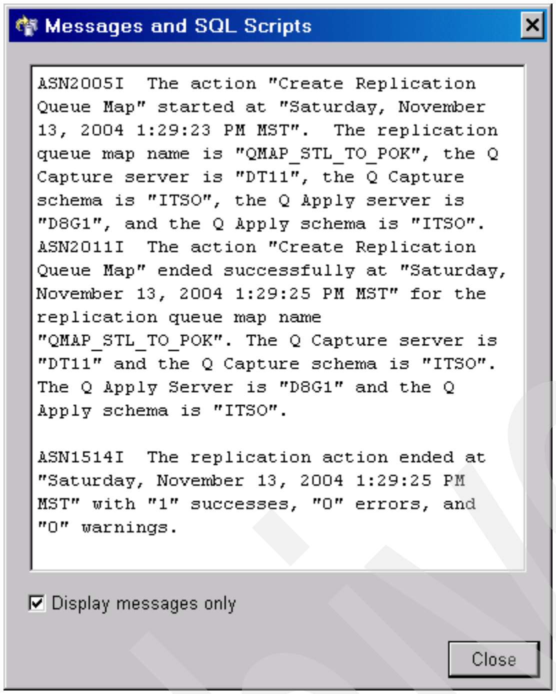


Figure 3-41 Messages and SQL Scripts

Click **Close** to bring up Figure 3-42 for choosing whether to execute the scripts right away or save the scripts for later execution.

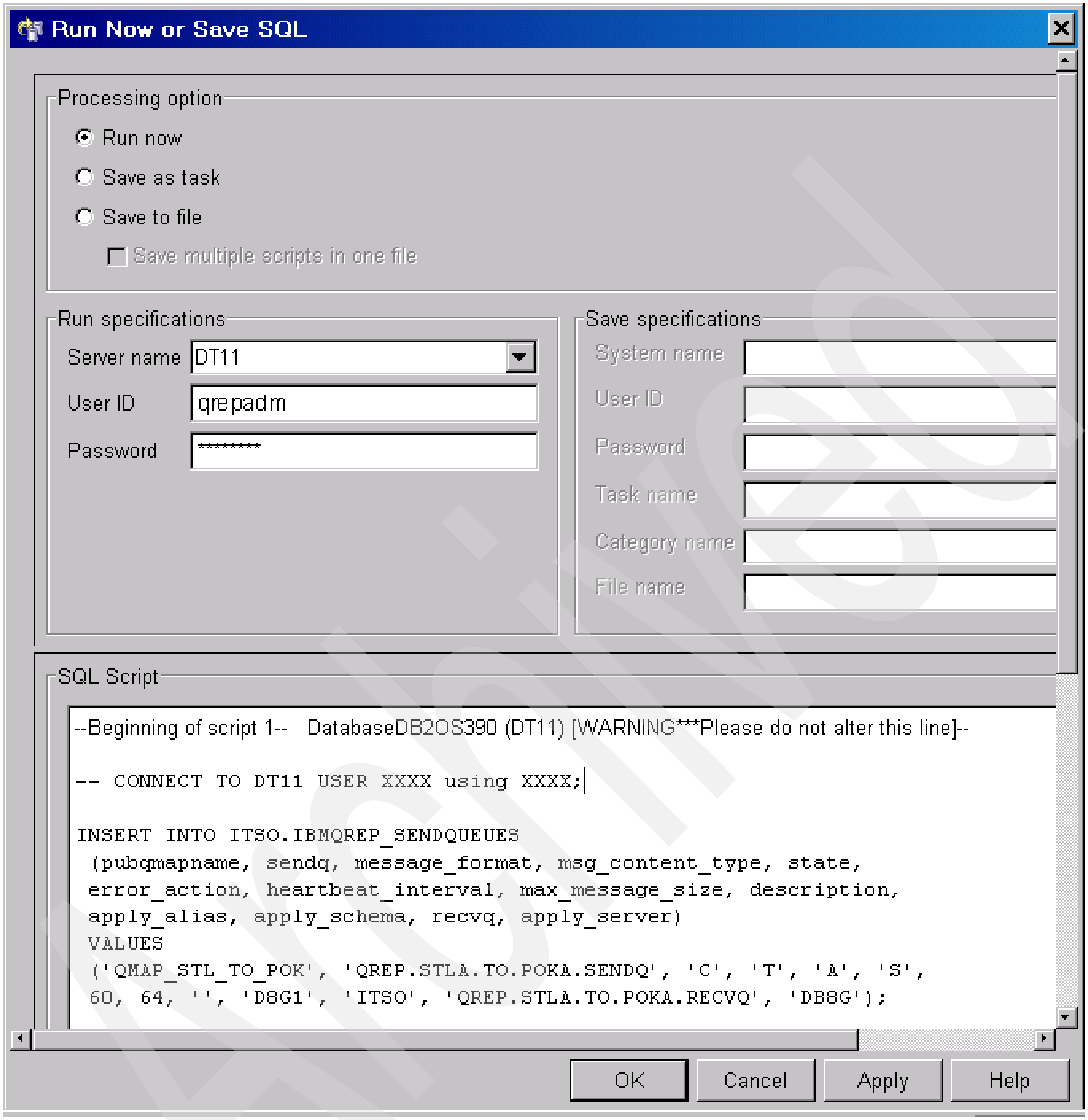


Figure 3-42 Run Now or Save SQL

Select the **Run now** option and click **OK** to execute the script immediately. Figure 3-43 is displayed when the scripts run successfully.

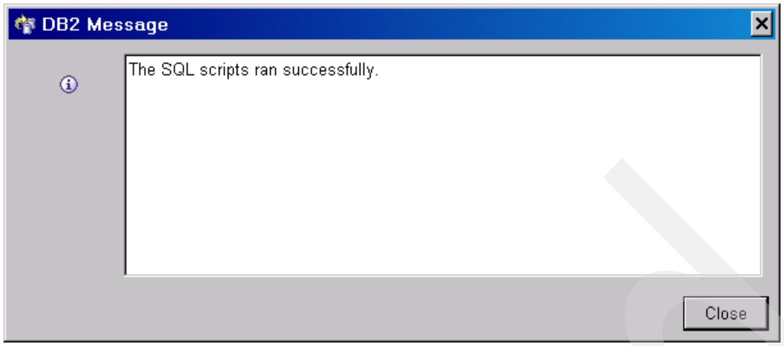


Figure 3-43 SQL scripts ran successfully message

Click **Close** to proceed to Figure 3-44, which displays the just created Replication Queue Map.

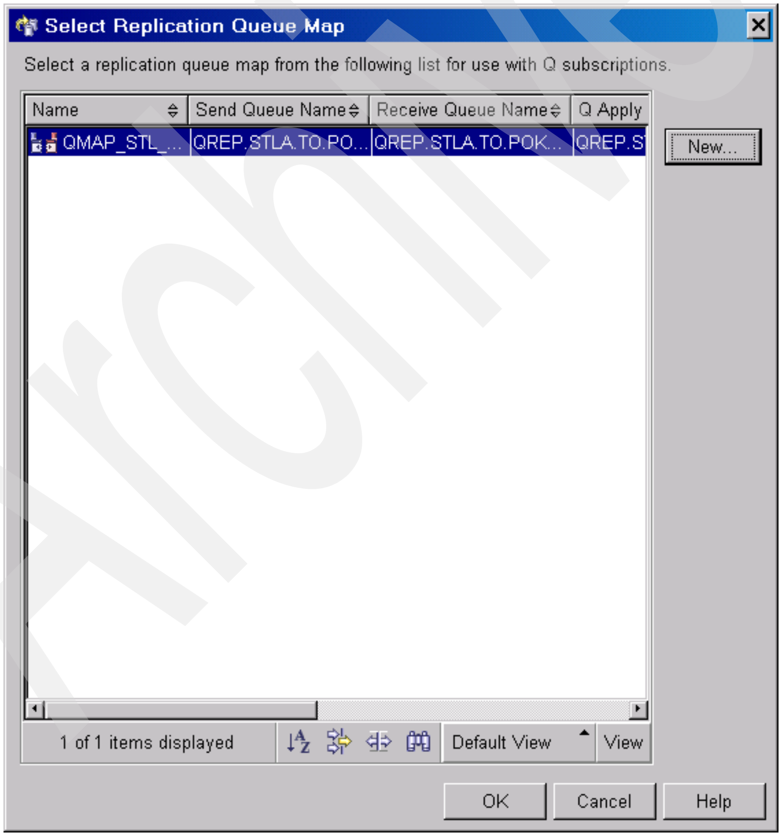


Figure 3-44 Select Replication Queue Map

Highlight the created replication queue map and click **OK** to proceed to Figure 3-45.

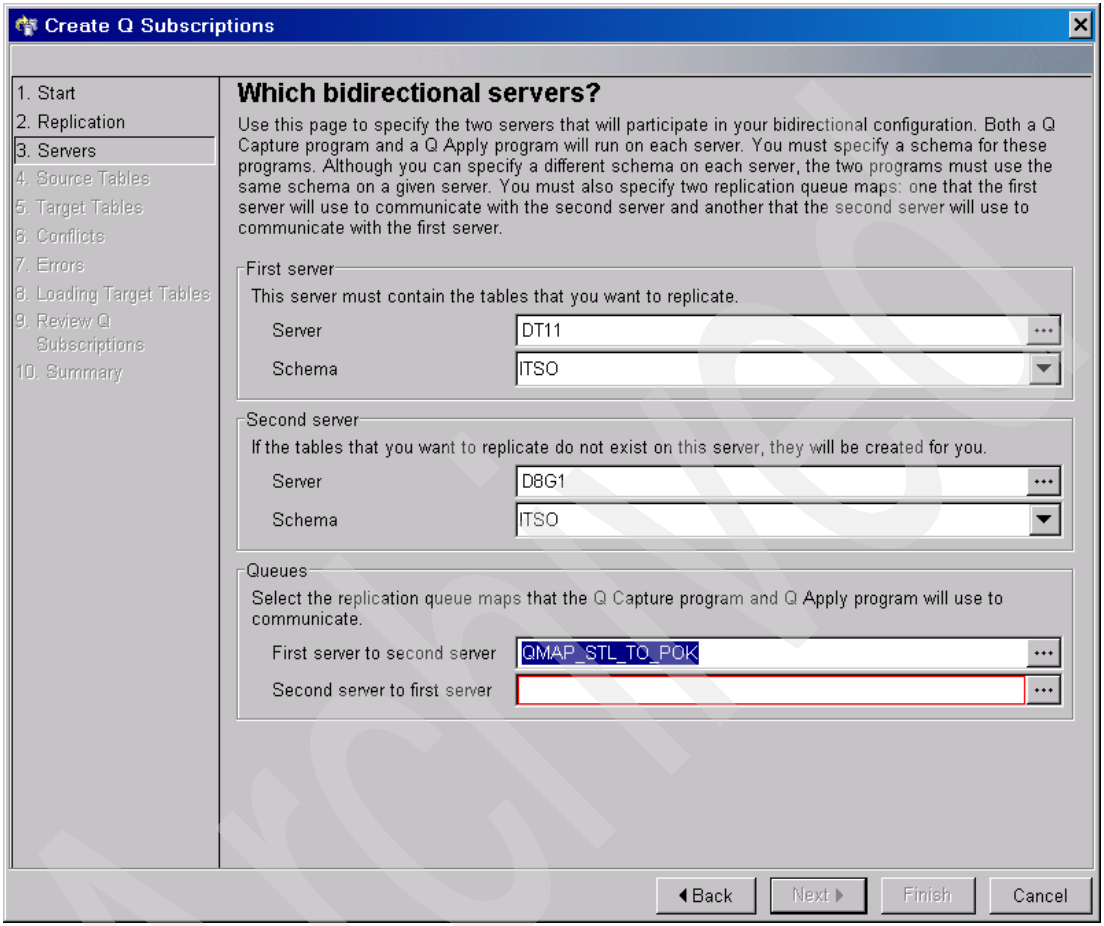


Figure 3-45 Which bidirectional servers

The next step is to select the replication queue map for communication between the “Second server to first server.”

Repeat the steps performed earlier for the replication queue map definition from the “First server to second server” (as described in Figure 3-35 on page 142 through Figure 3-44 on page 150) using the appropriate template values corresponding to WTSC53 in Table 3-2 on page 86 and Table 3-3 on page 88. This is described in Figure 3-45 through Figure 3-52 on page 158.

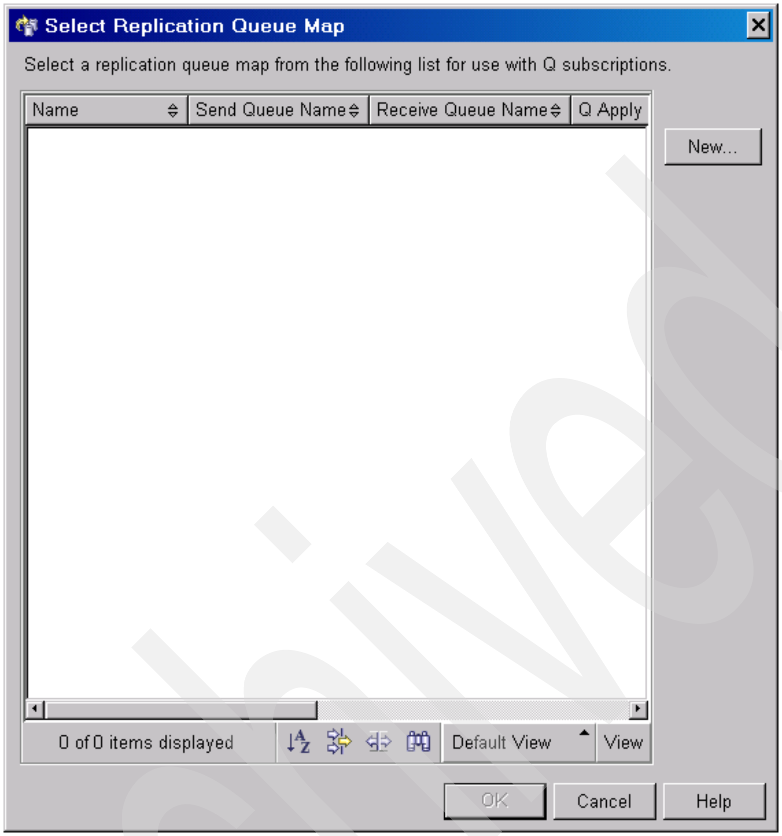


Figure 3-46 Select Replication Queue Map

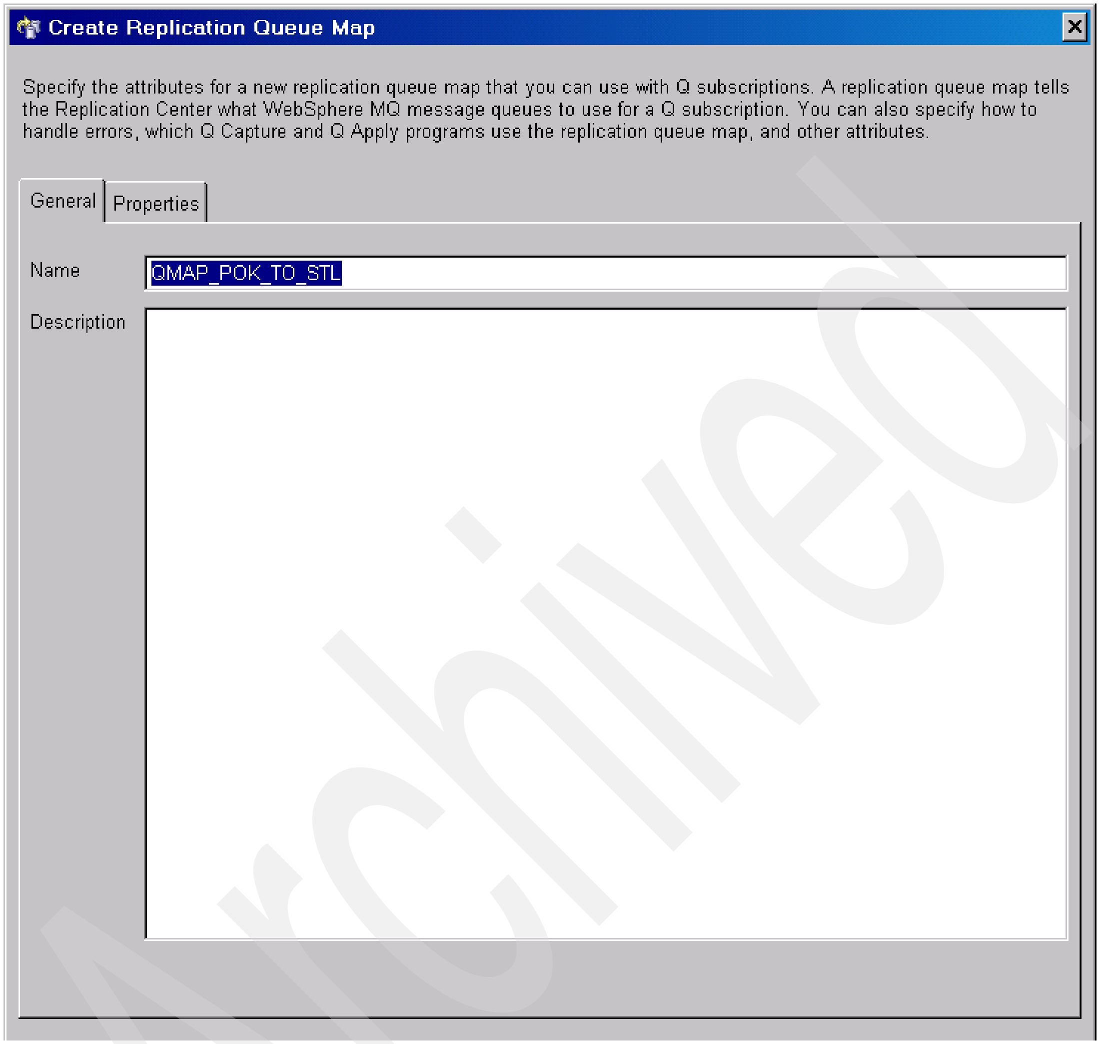


Figure 3-47 Create Replication Queue Map

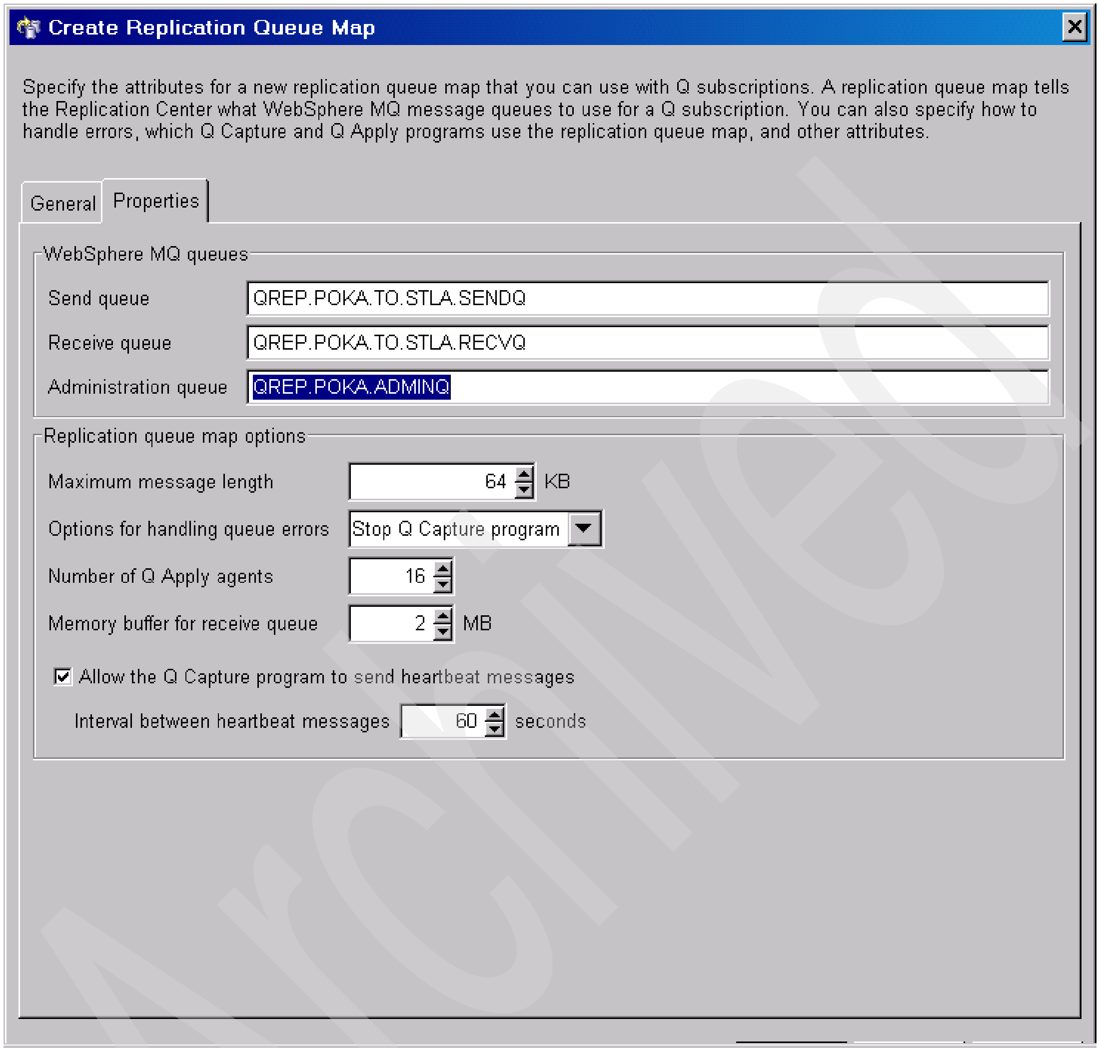


Figure 3-48 Create Replication Queue Map - Properties

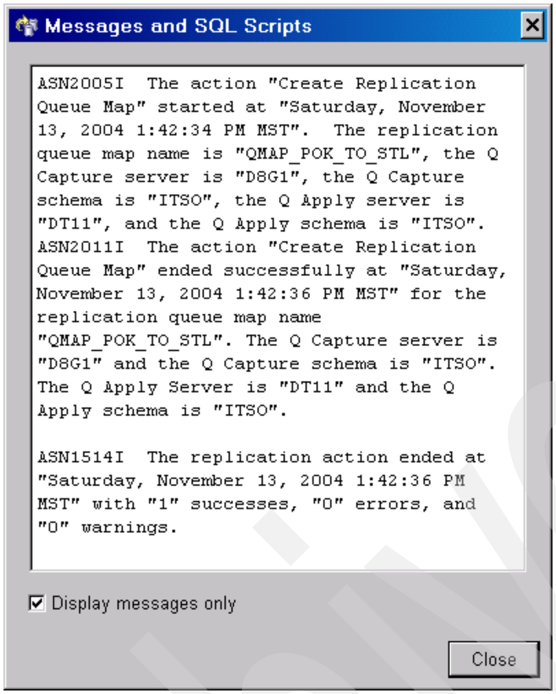


Figure 3-49 Messages and SQL Scripts

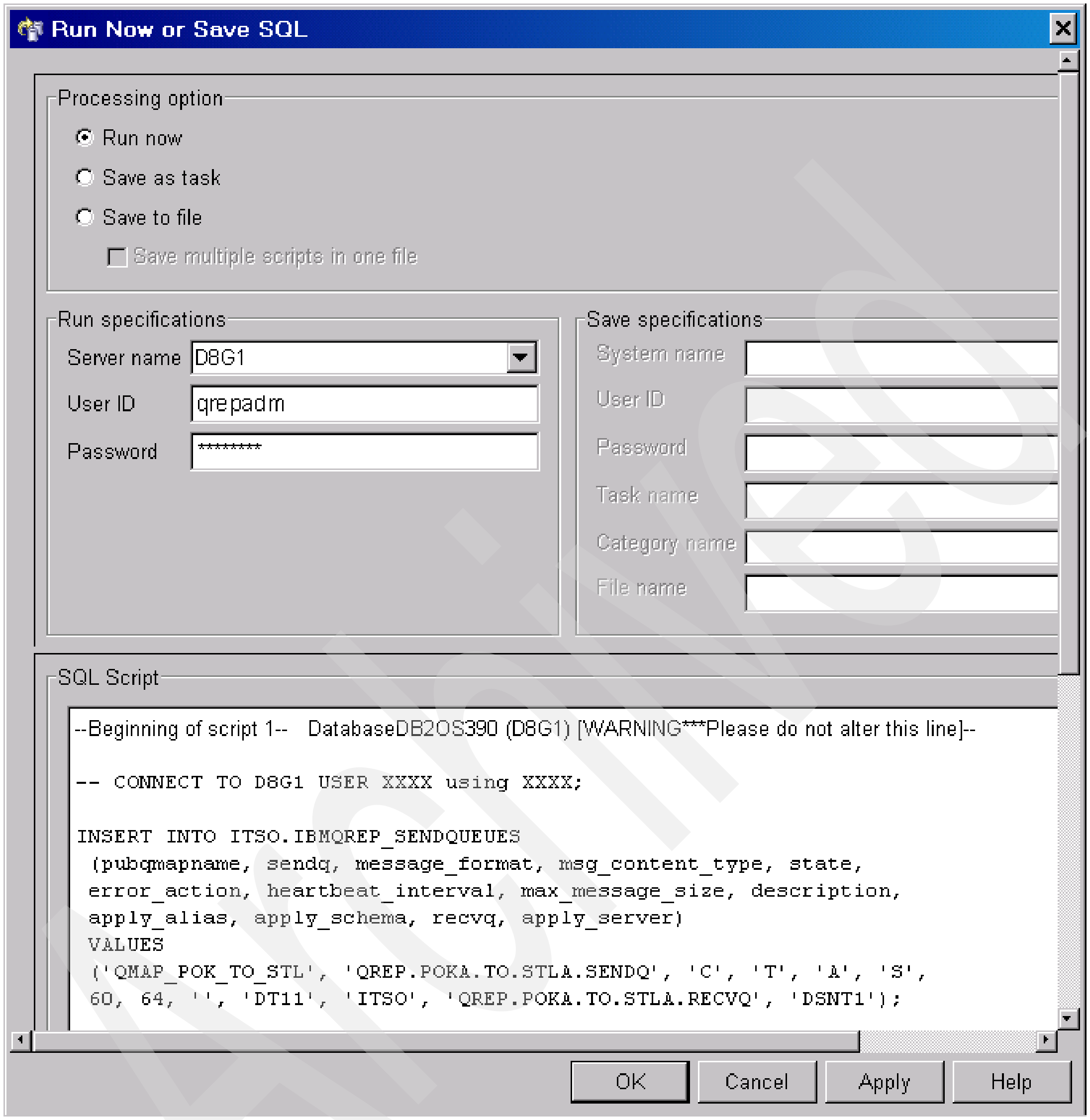


Figure 3-50 Run Now or Save SQL

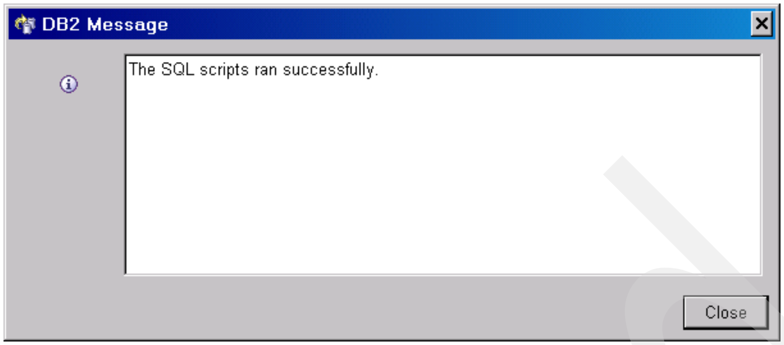


Figure 3-51 SQL scripts ran successfully message

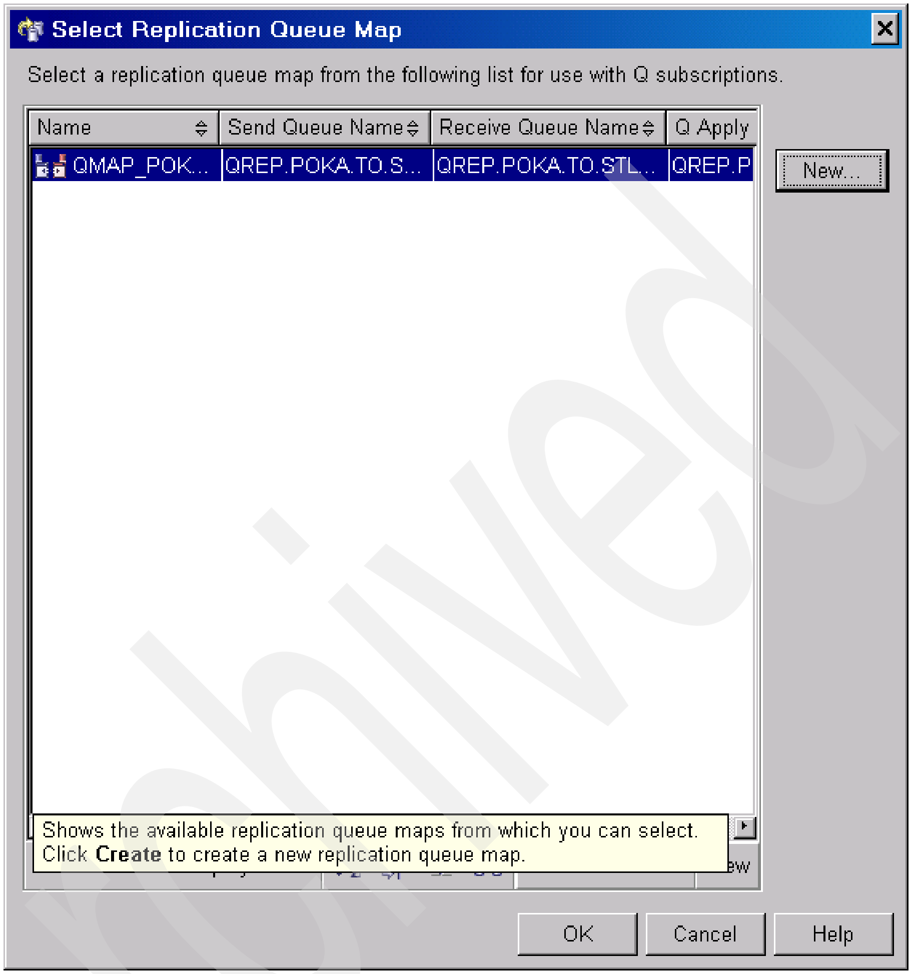


Figure 3-52 Select Replication Queue Map

At the completion of creating the replication queue maps for communication between the servers (see Figure 3-53), the individual subscriptions need to be defined.

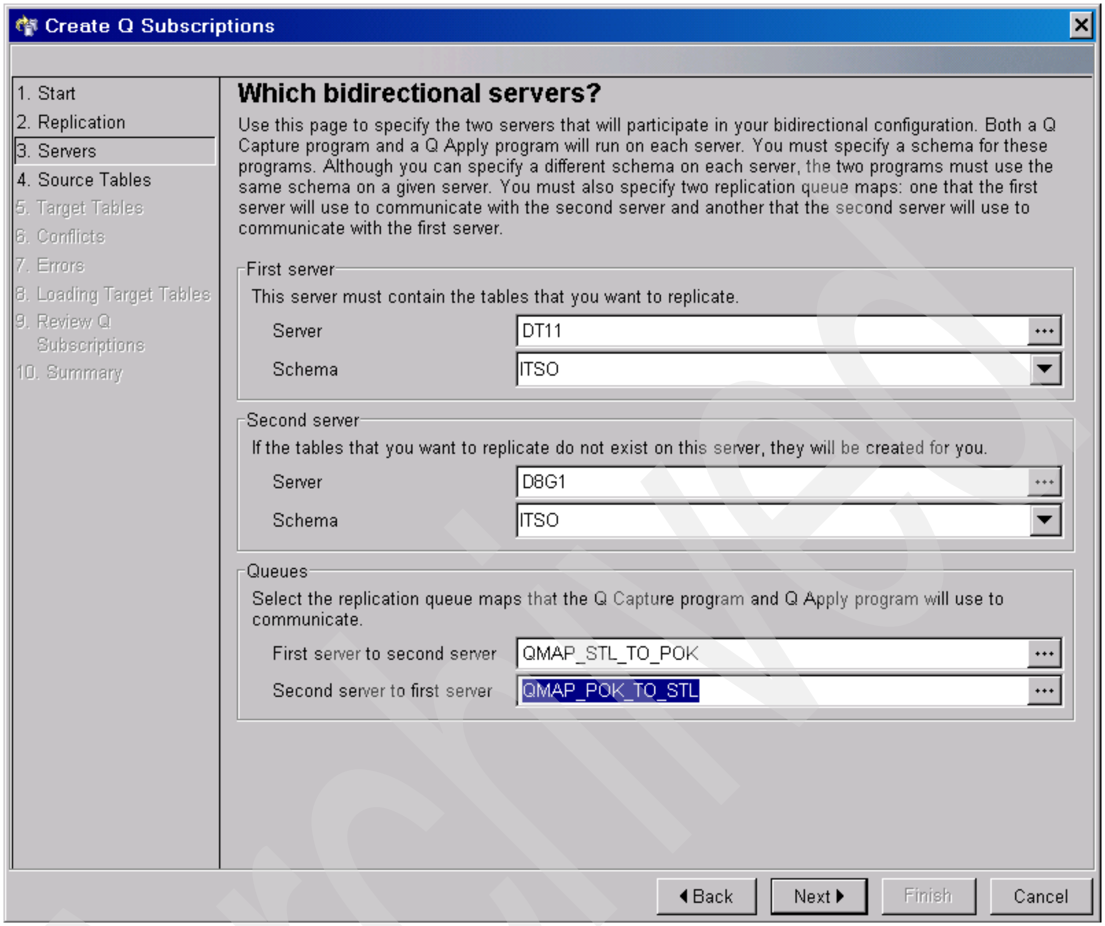


Figure 3-53 Completion of replication queue maps selection

Click **Next** (in the screen shown in Figure 3-53) to begin the process of defining the subscriptions, as shown in Figure 3-54.

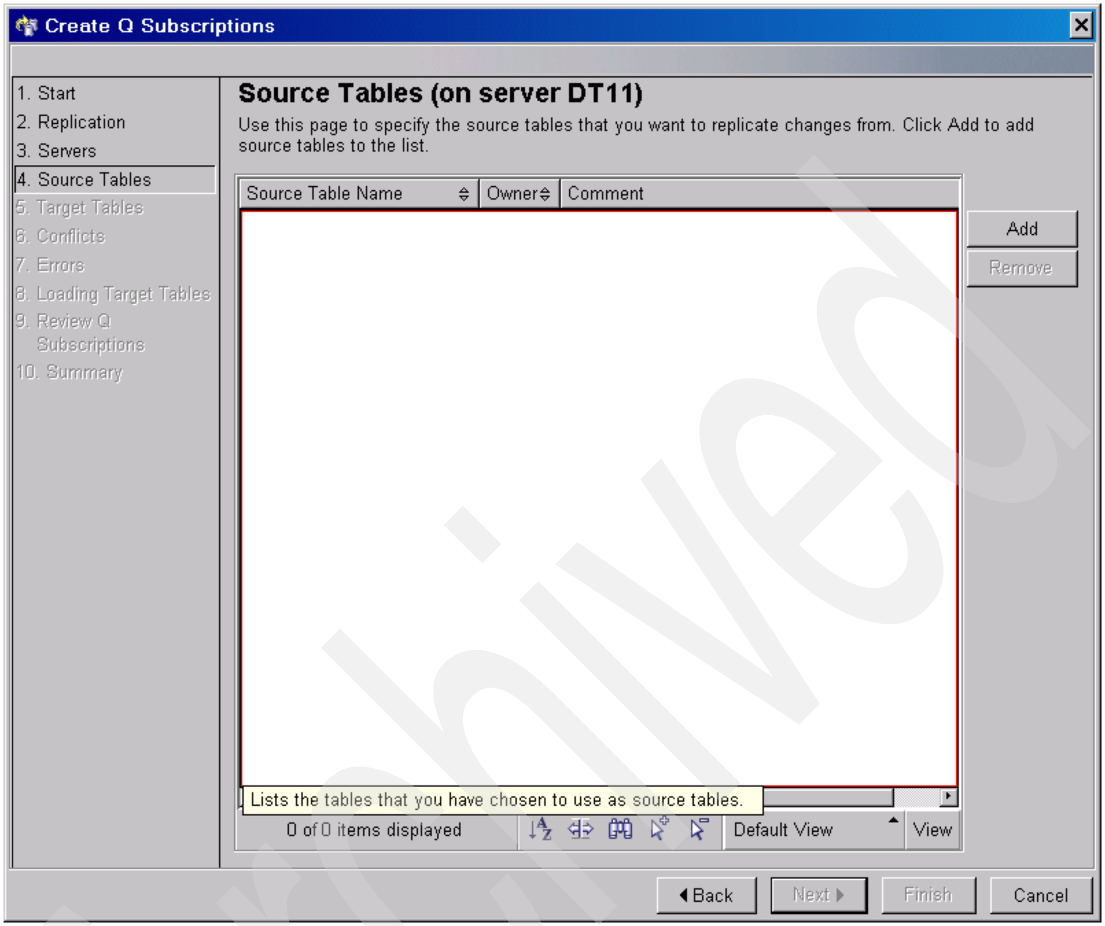


Figure 3-54 Create Q Subscriptions - Source Tables (on server DT11)

Since there are no tables to select from, click the **Add** button (Figure 3-54) to add to the list of tables available for subscriptions (as shown in Figure 3-55).

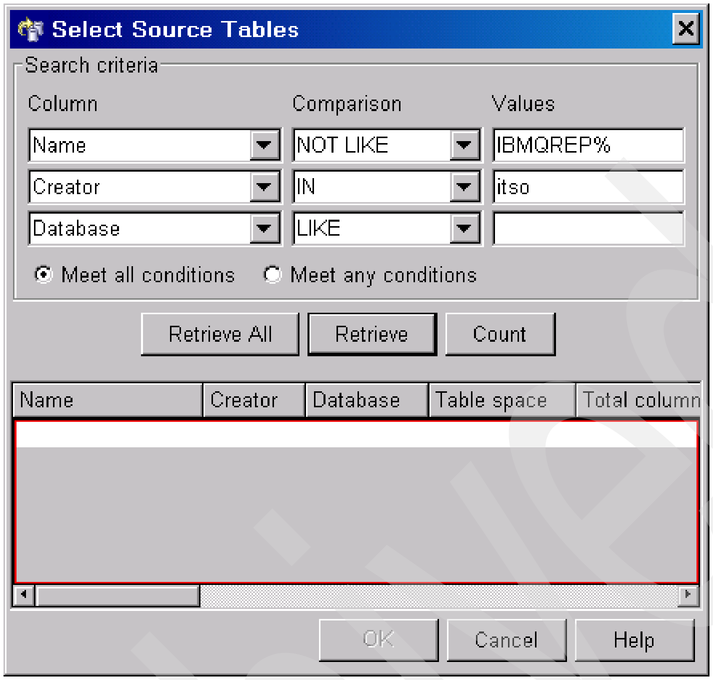


Figure 3-55   Select Source Tables

Exclude all tables that have a Name of "IBMQREP%", and include tables with Creator name of "itso" (the application tables for this configuration). Click **Retrieve** to list all the tables that qualify, as shown in Figure 3-56. Clicking **Retrieve All** ignores the criteria.

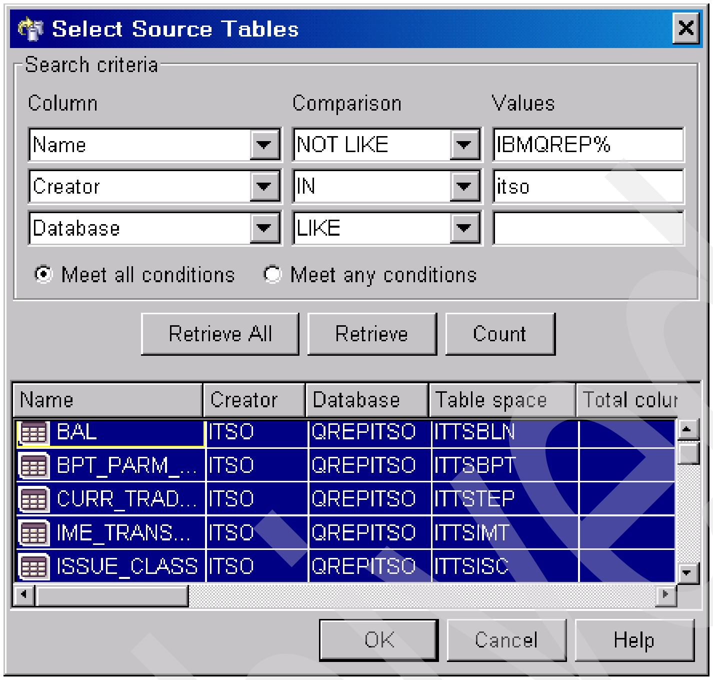


Figure 3-56 Select Source Tables

Go back and forth between Figure 3-56 and Figure 3-55 until all 23 tables in the application have been selected (23 tables in reference A.29 in Table 3-3 on page 88). The complete list is shown in Figure 3-57.

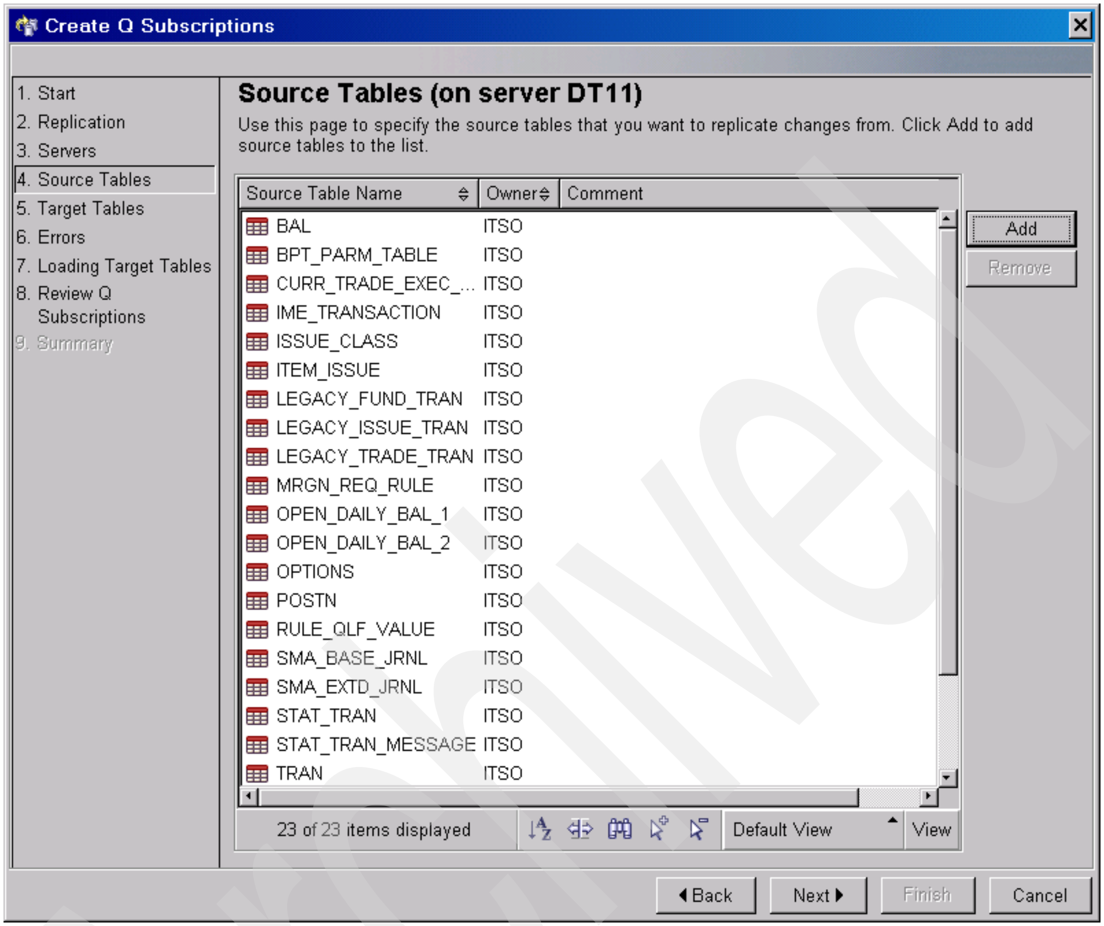


Figure 3-57 Complete list of tables

Click **Next** to set the profiles for the target tables, as shown in Figure 3-58.

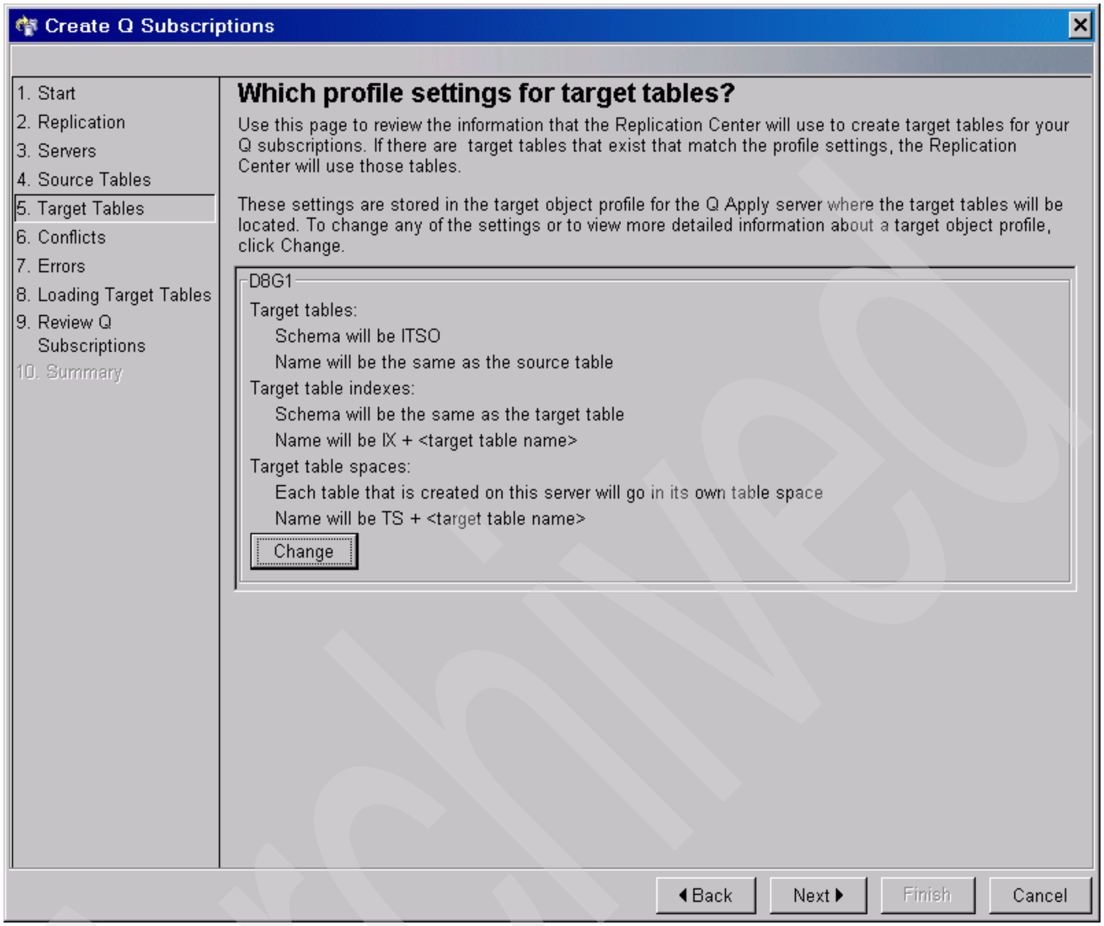


Figure 3-58 Create Q Subscriptions - Which profile settings for target tables

Figure 3-58 provides the option for changing the default settings for the target tables (reference A.28a, A.28b, and A.28c in Table 3-3 on page 88). We chose to go with the defaults. Click **Next** to specify the conflict resolution rules for the bidirectional topology in Figure 3-59.

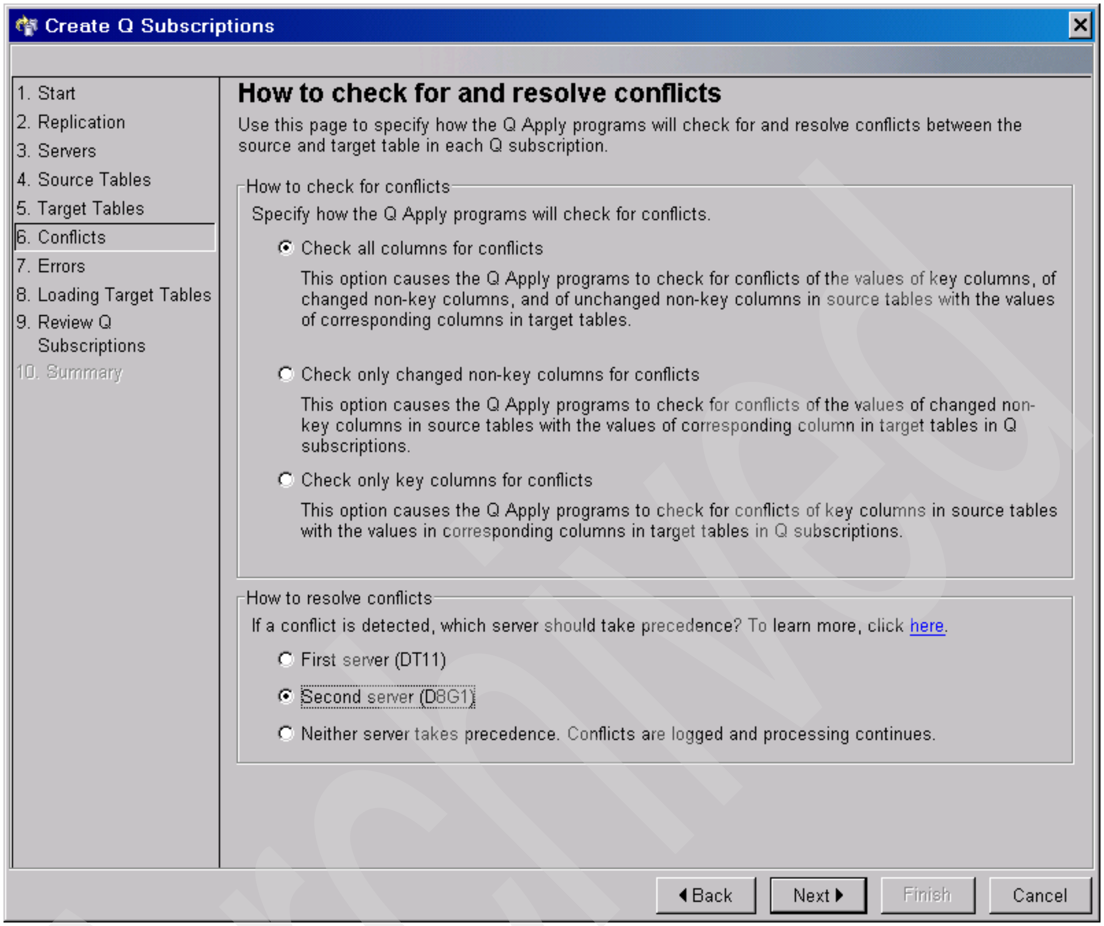


Figure 3-59 How to check for and resolve conflicts

In Figure 3-59, select **Check all columns for conflicts** and **Second server (D8G1)** to take precedence when a conflict is detected (reference A.28d, A.28e, and A.28f in Table 3-3 on page 88). Click **Next** to specify the action to be taken in the event of errors, as shown in Figure 3-60.

> **Note:** We chose the **Check all columns for conflicts** option instead of the other options to allow for the greatest possible scrutiny to ensure that the data remains synchronized correctly. Selection of the Second server (D8G1) as the winner for conflicts was made because this is the server that will be the failover server whose changes must override the primary server at switchback time.

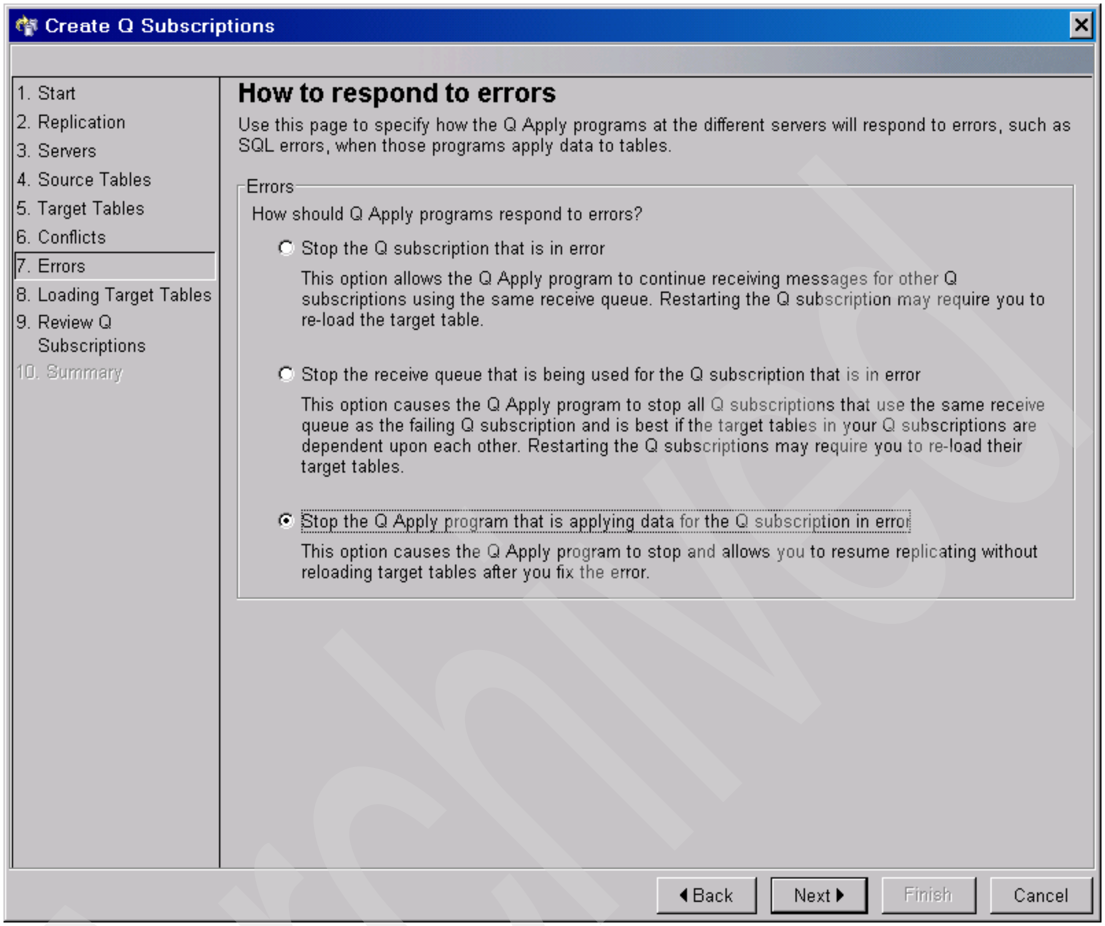


Figure 3-60 How to respond to errors

We selected the **Stop the Q Apply program that is applying data for the Q subscription in error** setting (reference A.28f in Table 3-3 on page 88).

We chose this option for the following reason:

- "Stop the Q subscription that is in error" would mean that Q Capture would continue through the DB2 log and ignore further changes to the table affected. This would force the reload/resynch of the table before the subscription can be reactivated, which can be very time consuming.
- "Stop the receive queue that is being used for the Q subscription that is in error" would stop all subscriptions that use this particular qmap. This would force the reload/resynch of *all* the tables in the replication queue map, which is undesirable.

- “Stop the Q Apply program that is applying data for the Q subscription in error” affects latency by introducing a delay; however, it permits correction without having to reload/resynch.

Click **Next** to specify how the target tables should be loaded in Figure 3-61.

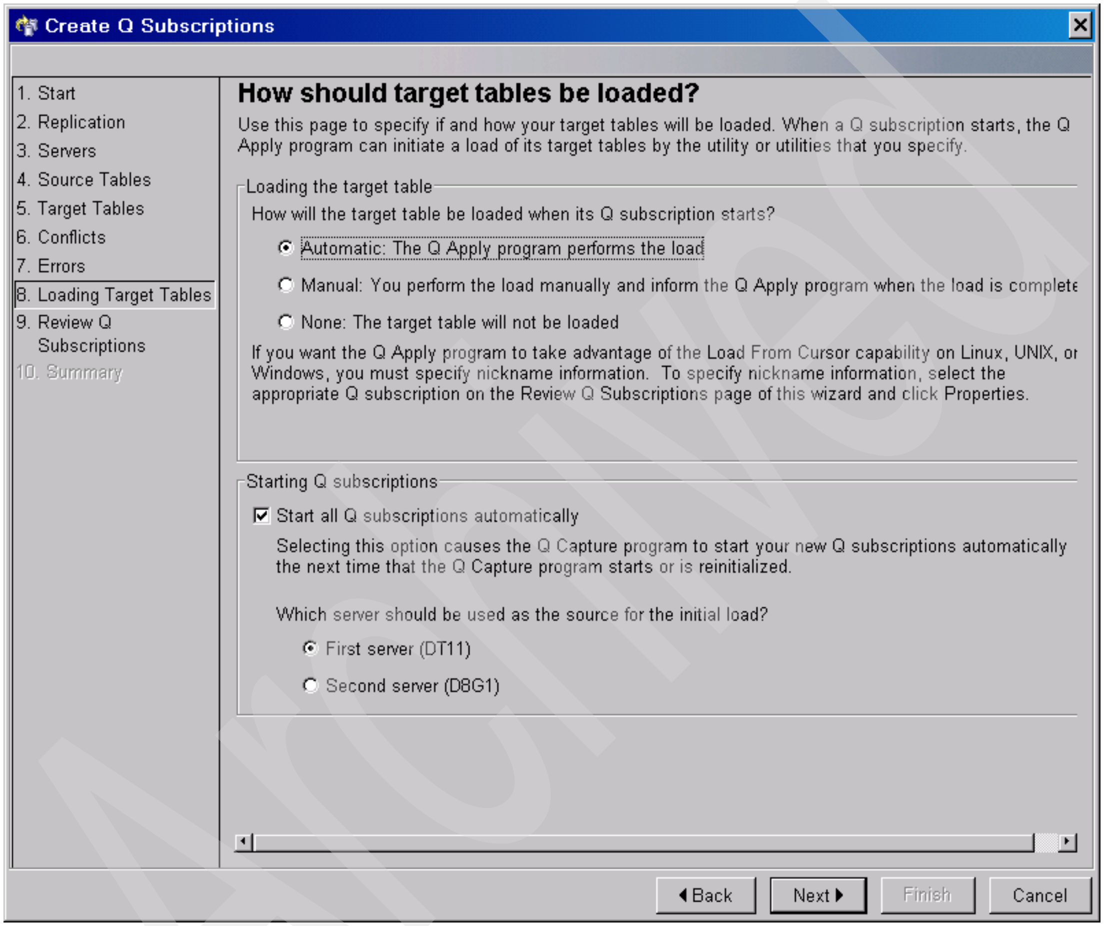


Figure 3-61 How should target tables be loaded

We chose the **Automatic: The Q Apply program performs the load** option (reference A.28g and A.28h in Table 3-3 on page 88), the **Start all Q subscriptions automatically** option, and selected the **First server (DT11)** to be the source for the initial load, as shown in Figure 3-61.

Important: Since the automatic loading of tables was selected, the initialization of the Q Apply program on the Second Server (D8G1) will cause Q Capture on the first server (DT11) to prompt Q Apply to initiate loading of the tables. This is accomplished on z/OS systems using the DRDA connectivity from the second server(D8G1) to the first server(DT11), as described in Example 3-12 on page 98 and Example 3-13 on page 99 in 3.5.4, "Step 4: Set up user IDs, privileges, and database servers" on page 89.

Click **Next** to review the subscriptions defined.

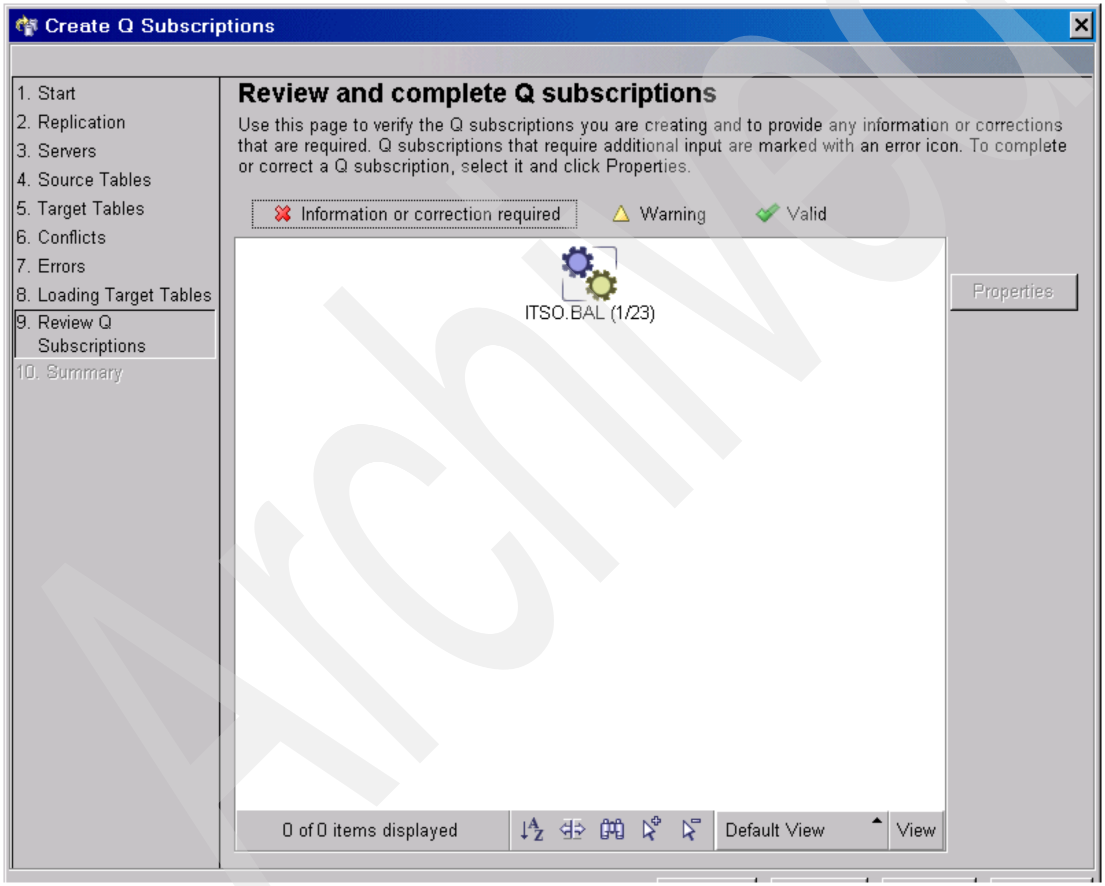


Figure 3-62 Review and complete Q subscriptions

Figure 3-62 is displayed while both z/OS databases are checked for accuracy of the replication request, and it is verified that the tables can be replicated. At the end of this Figure 3-63 is displayed, showing the 46 subscriptions—two for each of the twenty three tables.

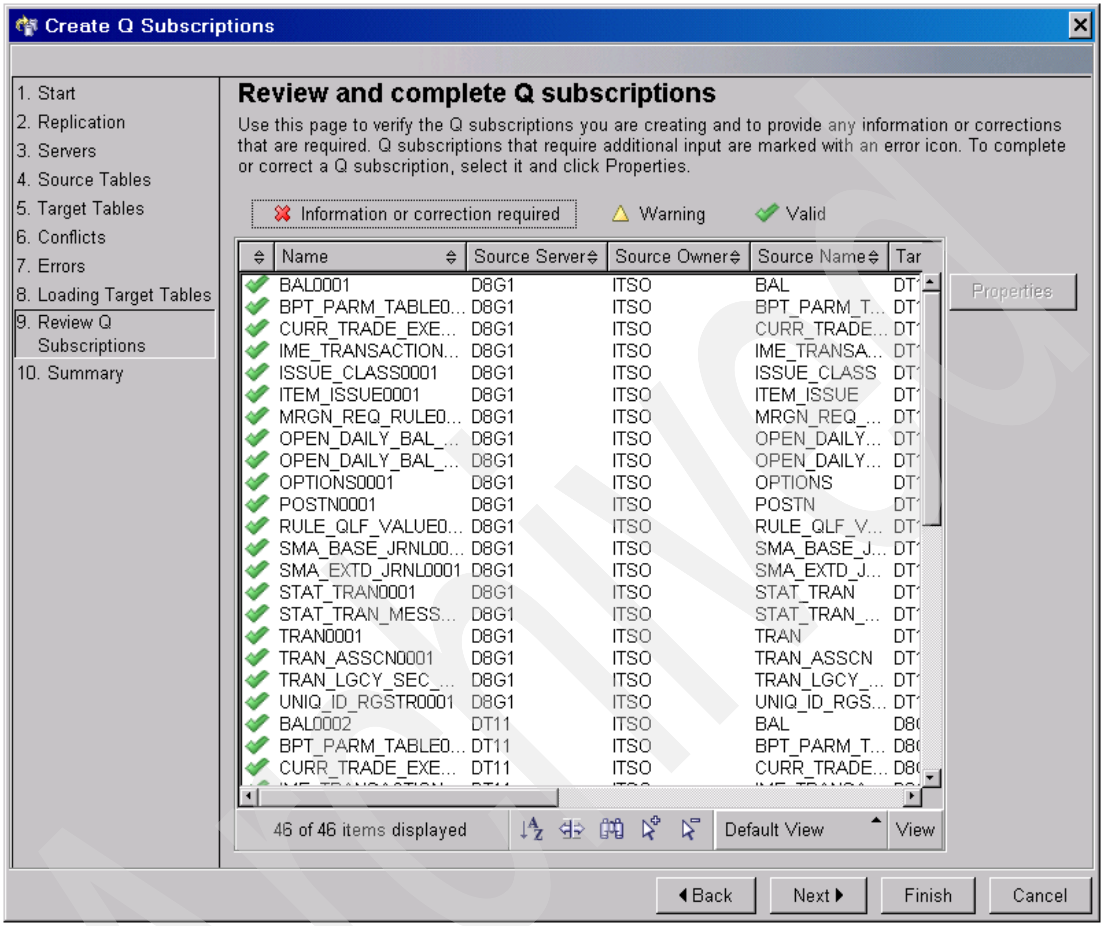


Figure 3-63 Review and complete Q subscriptions

Figure 3-63 shows 46 subscriptions in all—two for each of the twenty three tables on each server. Click **Next** to complete the definition of the subscriptions and view the summary, as shown in Figure 3-64.

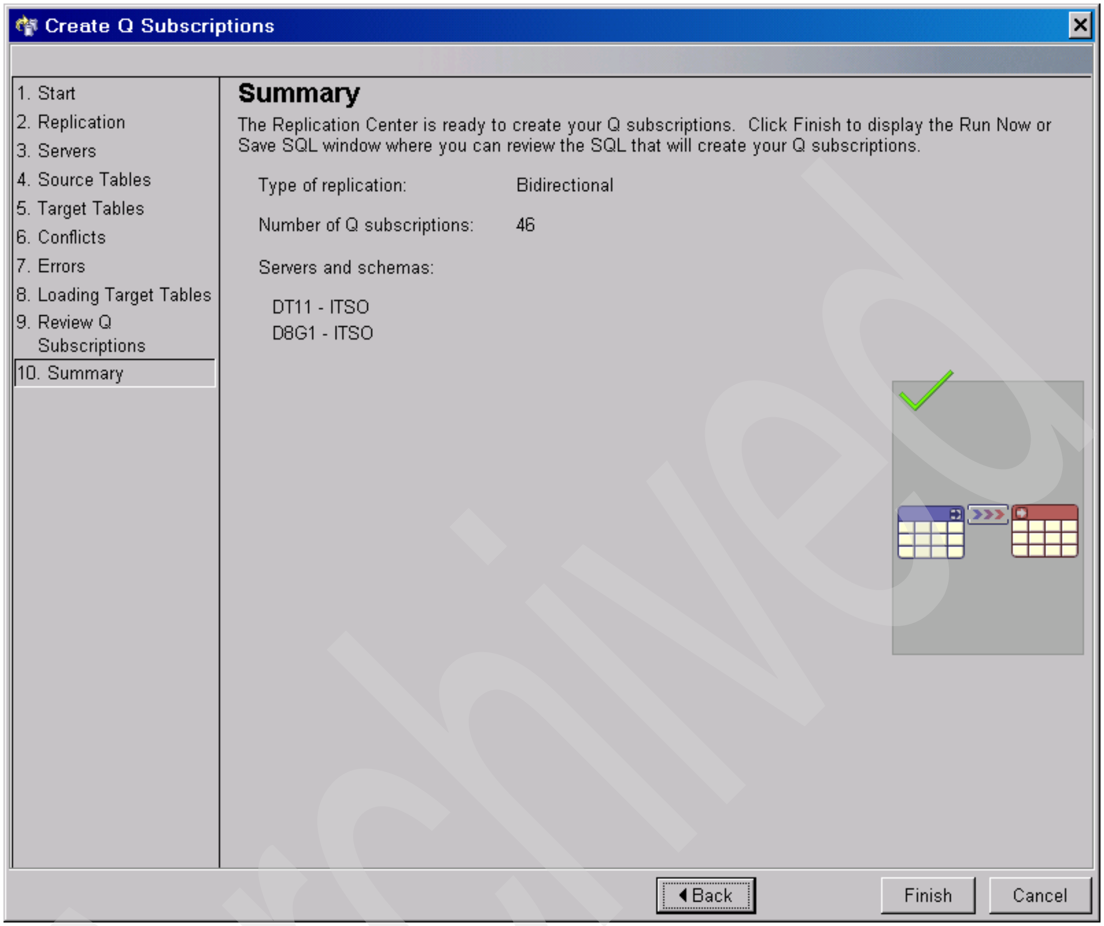


Figure 3-64 Create Q Subscriptions - Summary

Review the summary in Figure 3-64 and click **Finish** to generate the scripts for creating the various objects on the two servers, as shown in Figure 3-65.

Click the **Back** button to go back and correct any errors.

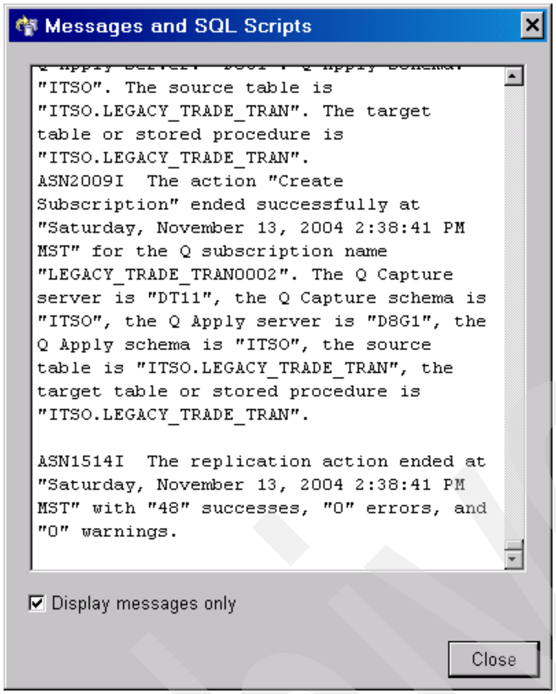


Figure 3-65 Messages and SQL Scripts

Click **Close** to decide whether to run the scripts right away or save them for later execution, as shown in Figure 3-66.

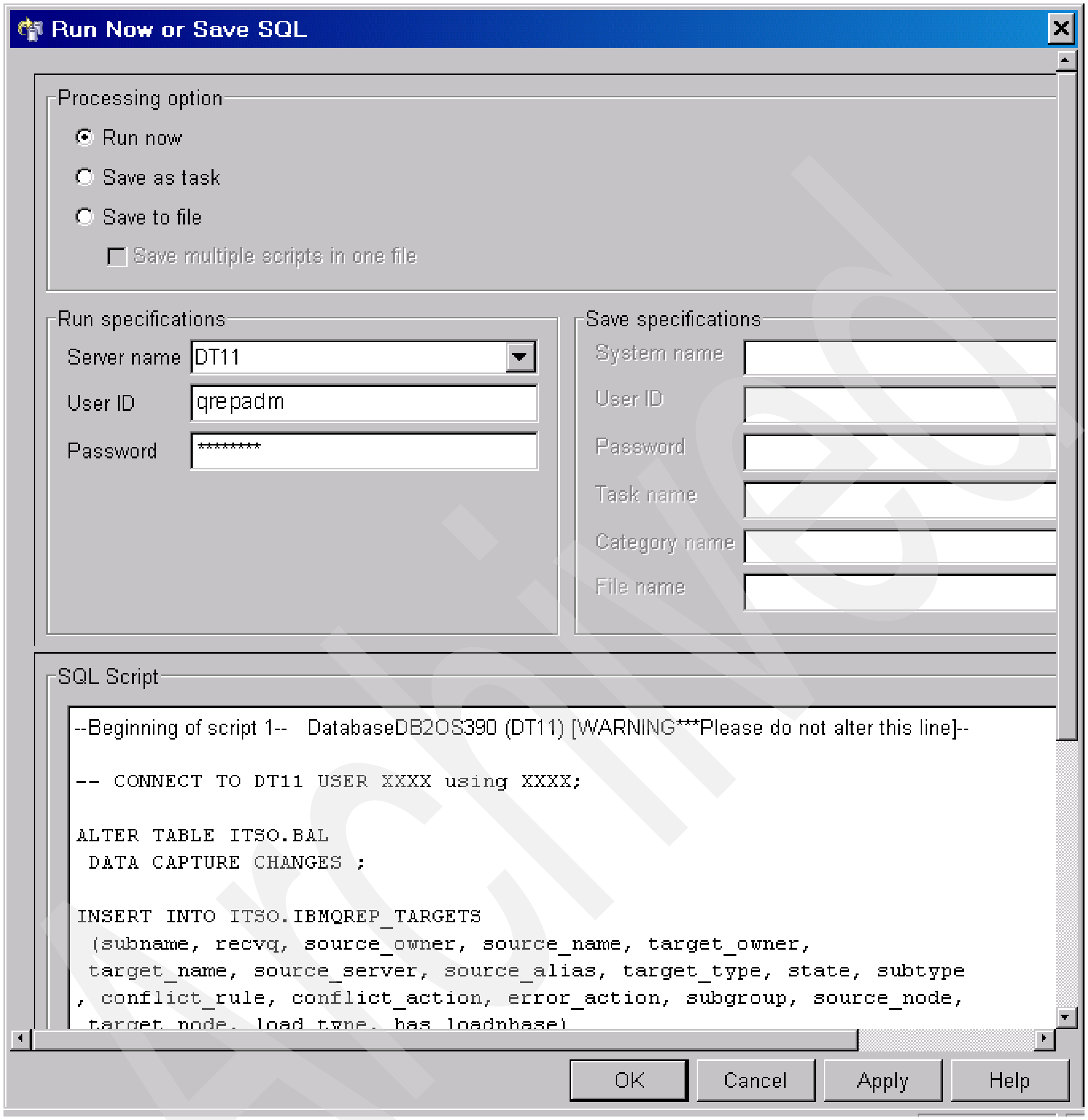


Figure 3-66 Run Now or Save SQL

We selected **Run now** and clicked OK to execute the scripts immediately. Figure 3-67 shows that the scripts ran successfully.

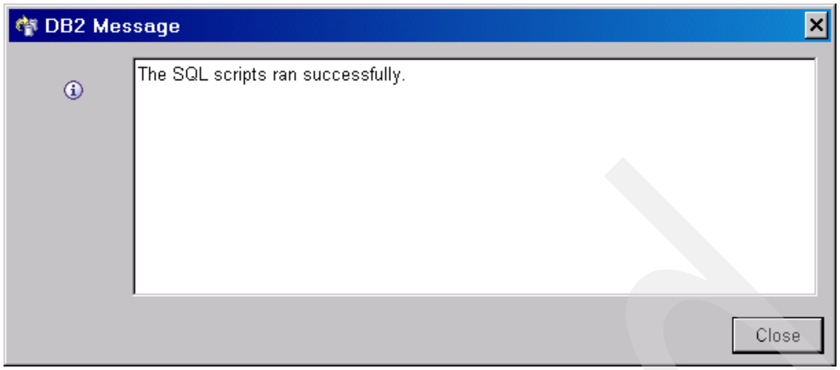


Figure 3-67 SQL scripts ran successfully message

With the completion of the subscriptions, the Q Capture program needs to be started on both servers.

> **Attention:** Q Capture should be started before Q Apply to ensure that all messages needed for activating subscriptions are processed correctly.

Step 6f: Start the Q Capture on both servers

As mentioned earlier, DAS was *not* installed in our environment. In many z/OS environments the control of tasks belongs within the security of the operating system itself. Procedures are defined and placed in the system PROCLIB for starting at IPL time, or via other startup mechanisms.

In our environment, we did not include them in the IPL startup processes, but chose to start them on an on demand basis.

Example 3-20 shows the procedure (ITSOQCAP) for starting the Q Capture program on STLABD1.

Example 3-20 Q Capture startup procedure ITSOQCAP on STLABD1

```
//ITSOQCAP PROC SERVE=DT11,        CAPTURE_SERVER
//          SCHEM=ITSO,            CAPTURE_SCHEMA
//          SMODE=WARMSA,          STARTMODE
//*         DPARM='DEBUG=Y',       DEBUG
//          DPARM='',              DEBUG
//          APARM=                 ADDITIONAL PARAMETERS
//*
//***********************************************************
//* RUN ON STLABD1
//***********************************************************
//*
```

```
//*EXECUTE THE CAPTURE PROGRAM WITH THE SUPPLIED PARAMETERS
//*
//QCAP      EXEC PGM=ASNQCAP,REGION=64M,
//  PARM='STORAGE(FF,FF,FF)/CAPTURE_SERVER=&SERVE CAPTURE_SCHEMA=&SCHEM
//              STARTMODE=&SMODE &DPARM &APARM'
//STEPLIB DD DSN=DPROPR.V820.BASE.SASNLOAD,DISP=SHR,
//           UNIT=SYSDA,VOL=SER=DRRSH1
//        DD DISP=SHR,DSN=DSNT1.DT11.SDSNEXIT
//        DD DISP=SHR,DSN=DSNT1.DT11.SDSNLOAD
//        DD DISP=SHR,DSN=DPROPR.HO.XML.LOAD,
//           UNIT=SYSDA,VOL=SER=DRRSH1
//MSGS     DD PATH='/usr/lpp//db2repl_08_02/msg/En_US/db2asn.cat'
//CEEDUMP  DD  DUMMY
//SYSPRINT DD  SYSOUT=*
//CAPSPILL DD  DSN=&&CAPSPL,DISP=(NEW,DELETE,DELETE),
//             UNIT=SYSDA,SPACE=(CYL,(50,100)),
//             DCB=(RECFM=VB,BLKSIZE=6404)
//SYSUDUMP DD DUMMY
//SYSTERM  DD DUMMY
//*
```

When a Q Capture program is started from the Replication Center, additional parameters can be specified, such as CAPTURE_PATH and COMMIT_INTERVAL. These same parameters may also be specified in the z/OS procedure in the APARM parameter in Example 3-20. To change these parameter values for other than a single session, the IBMQREP_CAPPARMS table can be modified. Example 3-21 shows the update of the CAPTURE_PATH column with the name of an MVS sequential dataset for the log file. This will produce a log file of the name <userid>.DB2PROD.DT11.ITSO.QCAP.LOG, where <userid> is the user ID associated with the Q Capture program. All informational and error messages will be placed in either this log file or on the JOB output.

Example 3-21 Update IBMQREP_CAPPARMS table parameters

```
UPDATE ITSO.IBMQREP_CAPPARMS SET CAPTURE_PATH = '//DB2PROD';
```

Example 3-22 and Example 3-23 show how this Q Capture startup procedure described in Example 3-20 may be started and shut down on demand.

Example 3-22 Q Capture startup command on an MVS console on STLABD1

```
/S ITSOQCAP
```

Example 3-23 Q Capture shutdown command on an MVS console on STLABD1

```
/F ITSOQCAP,STOP
```

Note: Several options for communicating with Q replication programs running on the z/OS platform are discussed in Appendix G, "Communicating with Q replication programs in z/OS" on page 873.

Example 3-24, Example 3-25, and Example 3-26 are the corresponding examples for the Q Capture program procedure (QCAPITSO) on WTSC53.

Example 3-24 Q Capture startup procedure QCAPITSO on WTSC53

```
//QCAPITSO PROC SERVE=D8G1,       CAPTURE_SERVER
//          SCHEM=ITSO,           CAPTURE_SCHEMA
//          SMODE=WARMSA,         STARTMODE
//*         DPARM='DEBUG=Y',      DEBUG
//          DPARM='',             DEBUG
//          APARM=                ADDITIONAL PARAMETERS
//*
//***********************************************************
//* RUN ON SC53
//***********************************************************
//*
//*
//*EXECUTE THE CAPTURE PROGRAM WITH THE SUPPLIED PARAMETERS
//*
//QCAP      EXEC PGM=ASNQCAP,REGION=64M,
//  PARM='STORAGE(FF,FF,FF)/CAPTURE_SERVER=&SERVE CAPTURE_SCHEMA=&SCHEM
//              STARTMODE=&SMODE &DPARM &APARM'
//STEPLIB  DD DSN=ASN.V8R2M0.SASNLOAD,DISP=SHR
//         DD DSN=DB8G8.SDSNLOAD,DISP=SHR
//*        DD DSN=IXM.SIXMMOD1,DISP=SHR
//MSGS     DD PATH='/usr/lpp//db2repl_08_02/msg/En_US/db2asn.cat'
//CEEDUMP  DD  DUMMY
//SYSPRINT DD  SYSOUT=*
//CAPSPILL DD  DSN=&&CAPSPL,DISP=(NEW,DELETE,DELETE),
//             UNIT=VIO,SPACE=(CYL,(50,100)),
//             DCB=(RECFM=VB,BLKSIZE=6404)
//SYSUDUMP DD DUMMY
//SYSTERM  DD DUMMY
//*
```

Example 3-25 Q Capture startup command on an MVS console on WTSC53

```
/S QCAPITSO
```

Example 3-26 Q Capture shutdown command on an MVS console on WTSC53

```
/F QCAPITSO,STOP
```

Step 6g: Start the Q Apply on both servers

As described in “Step 6f: Start the Q Capture on both servers” on page 173, startup procedures were also defined for the Q Apply program, which are to be started on demand.

Example 3-27 shows the procedure (ITSOQAPP) for starting the Q Apply program on STLABD1.

Example 3-27 Q Apply startup procedure ITSOQAPP on STLABD1

```
//ITSOQAPP PROC SERVE=DT11,        APPLY_SERVER
//            SCHEM=ITSO,         APPLY_SCHEMA
//*           DPARM='DEBUG=Y',    DEBUG
//            DPARM='',           DEBUG
//            APARM=              ADDITIONAL PARAMETERS
//*
//*******************************************************************
//*  PROC MUST RUN ON DT11, TO APPLY ITSO CAPTURE SCHEMA DATA FROM POK
//*******************************************************************
//*
//*
//* EXECUTE THE APPLY PROGRAM WITH THE SUPPLIED PARAMETERS
//*
//QAPP     EXEC PGM=ASNQAPP,REGION=64M,
//      PARM='STORAGE(FF,FF,FF)/APPLY_SERVER=&SERVE APPLY_SCHEMA=&SCHEM
//             &DPARM &APARM'
//STEPLIB DD DSN=DPROPR.V820.BASE.SASNLOAD,DISP=SHR,
//           UNIT=SYSDA,VOL=SER=DRRSH1
//        DD DISP=SHR,DSN=DSNT1.DT11.SDSNEXIT
//        DD DISP=SHR,DSN=DSNT1.DT11.SDSNLOAD
//        DD DISP=SHR,DSN=DPROPR.HO.XML.LOAD,
//           UNIT=SYSDA,VOL=SER=DRRSH1
//MSGS     DD PATH='/usr/lpp//db2repl_08_02/msg/En_US/db2asn.cat'
//CEEDUMP  DD  DUMMY
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD  DUMMY
//*
```

When a Q Apply program is started from the Replication Center, additional parameters can be specified, such as APPLY_PATH and DEADLOCK_RETRIES. These same parameters may also be specified in the z/OS procedure in the APARM parameter in Example 3-27. To change these parameter values for other than a single session, the IBMQREP_APPLYPARMS table can be modified. Example 3-28 shows the update of the APPLY_PATH column with the name of an MVS sequential dataset for the log file. This will produce a log file of the name <userid>.DB2PROD.DT11.ITSO.QAPP.LOG, where <userid> is the user ID associated with the Q Apply program. All informational and error messages will be placed in either this log file or on the JOB output.

Example 3-28 Update IBMQREP_APPLYPARMS table parameters

```
UPDATE ITSO.IBMQREP_APPLYPARMS SET APPLY_PATH = '//DB2PROD';
```

Example 3-29 and Example 3-30 show how this Q Apply startup procedure described in Example 3-27 may be started and shut down on demand.

Example 3-29 Q Apply startup command on an MVS console on STLABD1

```
/S ITSOQAPP
```

Example 3-30 Q Apply shutdown command on an MVS console on STLABD1

```
/F ITSOQAPP,STOP
```

> **Note:** Several options for communicating with Q replication programs running on the z/OS platform are discussed in Appendix G, “Communicating with Q replication programs in z/OS” on page 873.

Example 3-31, Example 3-32, and Example 3-33 are the corresponding examples for the Q Apply program procedure (QAPPITSO) on WTSC53.

Example 3-31 Q Apply startup procedure QAPPITSO on WTSC53

```
//QAPPITSO PROC SERVE=D8G1,        APPLY_SERVER
//              SCHEM=ITSO,        APPLY_SCHEMA
//*             DPARM='DEBUG=Y',   DEBUG
//              DPARM='',          DEBUG
//              APARM=             ADDITIONAL PARAMETERS
//*
//*******************************************************************
//* RUN ON SC53, TO APPLY THE ITSO CAPTURE SCHEMA DATA from STL
//*******************************************************************
//*
//*
//QAPP     EXEC PGM=ASNQAPP,REGION=64M,
```

```
//      PARM='STORAGE(FF,FF,FF)/APPLY_SERVER=&SERVE APPLY_SCHEMA=&SCHEM
//              &DPARM &APARM'
//STEPLIB  DD DSN=ASN.V8R2M0.SASNLOAD,DISP=SHR
//         DD DSN=DB8G8.SDSNLOAD,DISP=SHR
//         DD DSN=IXM.SIXMMOD1,DISP=SHR,
//         VOL=SER=Z04RE1,UNIT=3390
//MSGS     DD PATH='/usr/lpp//db2repl_08_02/msg/En_US/db2asn.cat'
//CEEDUMP  DD  DUMMY
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD  DUMMY
//*
```

Example 3-32 Q Apply startup command on an MVS console on WTSC53

```
/S QAPPITSO
```

Example 3-33 Q Apply shutdown command on an MVS console on WTSC53

```
/F QAPPITSO,STOP
```

Step 6h: Verify status of Q Capture and Q Apply processes

We have not yet started the Q Capture or Q Apply programs.

Our bidirectional replication configuration requested an automatic load of the target tables from the DT11 server, and automatic starting of the subscriptions (see Figure 3-61 on page 167).

When the Q Capture task is started on DT11, it recognizes the existence of new subscriptions and the need to activate them. It then sends Q subscription activation information to the Q Apply task running on the target server D8G1. The Q Apply task running on D8G1 then initiates the load process via DRDA from the DT11 server. The following screens monitor this process.

Launch the Replication Center from the DB2 Control Center by clicking the **Tools** tab and selecting **Replication Center**, and expand the **Q Capture Servers** folder in the navigation pane under the Definitions folder, as shown in Figure 3-68.

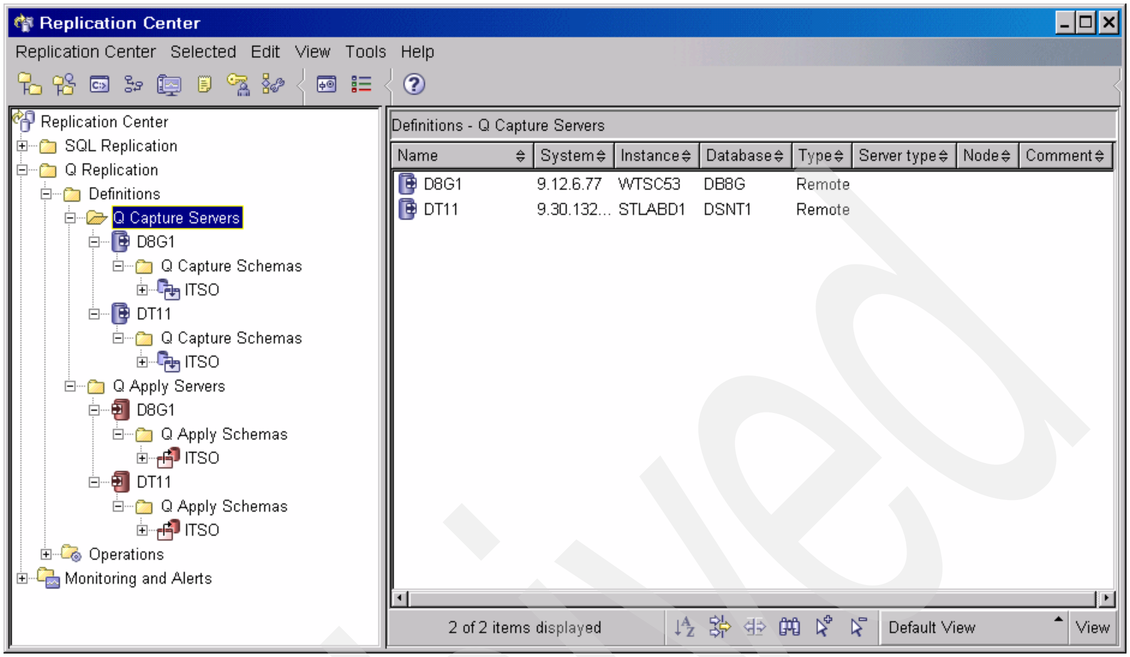


Figure 3-68 Replication Center - Q Capture servers definitions

Expand the **Operations** folder in the navigation pane in Figure 3-68, and highlight the **Q Capture Servers** folder to display the replication servers, as shown in Figure 3-69.

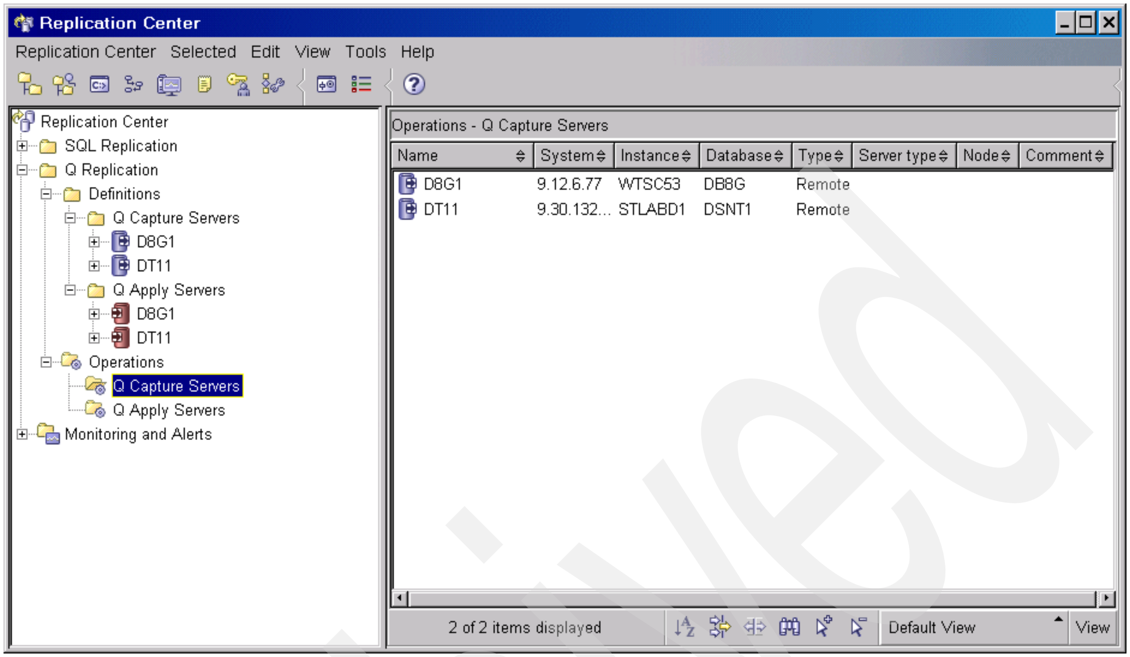


Figure 3-69 Operations - Q Capture Servers

The right-hand pane shows the two Q Capture servers D8G1 and DT11.

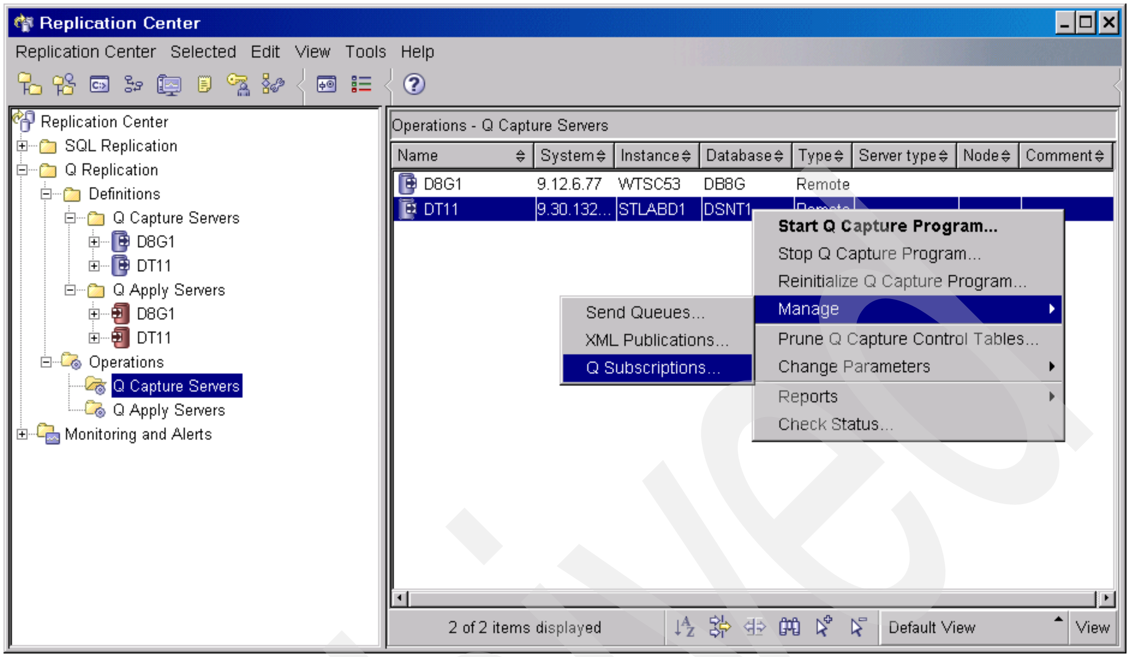


Figure 3-70 Manage Q Subscriptions - DT11

Right-click the **DT11** server name, and select **Manage** and then **Q Subscriptions...**, as shown in Figure 3-70. This displays Figure 3-71, which lists all the Q subscriptions and their status.

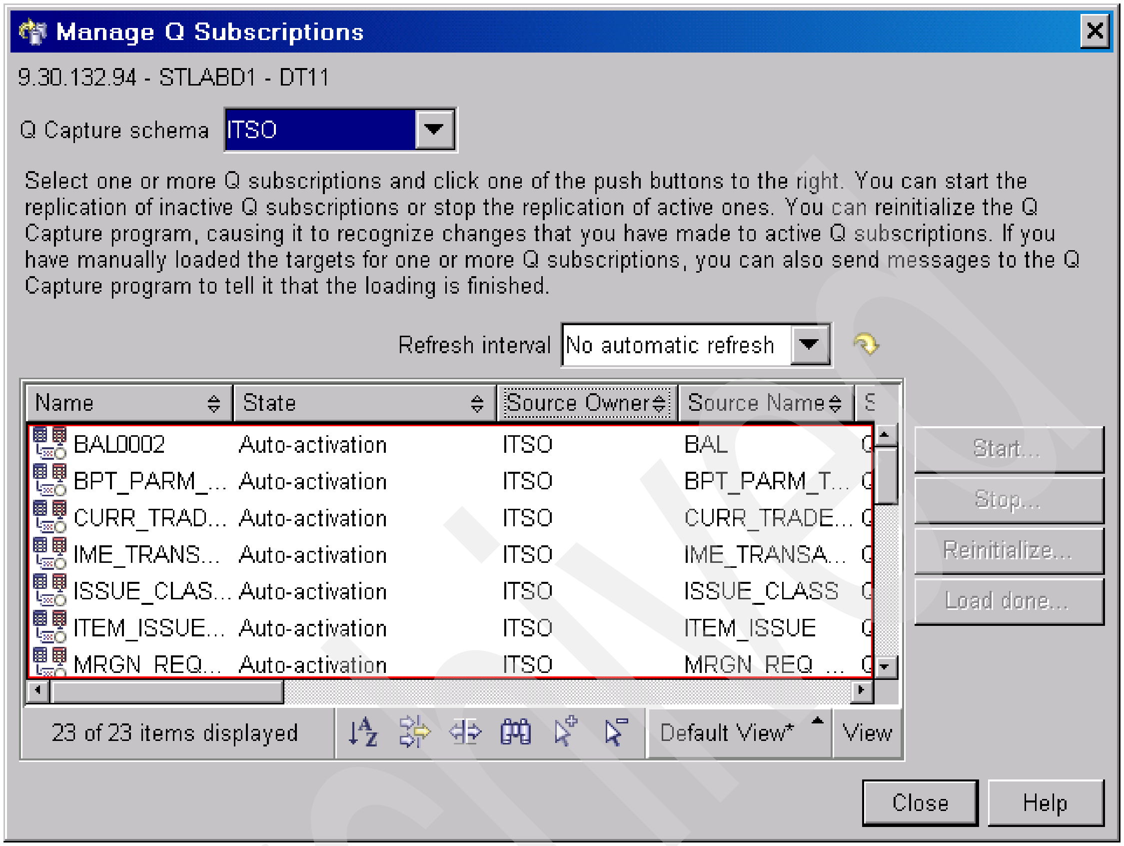


Figure 3-71 Q Subscriptions (DT11) and their states

The Refresh interval field and the yellow arrow next to it can be used to monitor the changing state of the subscriptions listed. The icons next to the Name field also indicate the state of the Q subscription.

Click **Close** to return to the previous screen (Figure 3-69 on page 180).

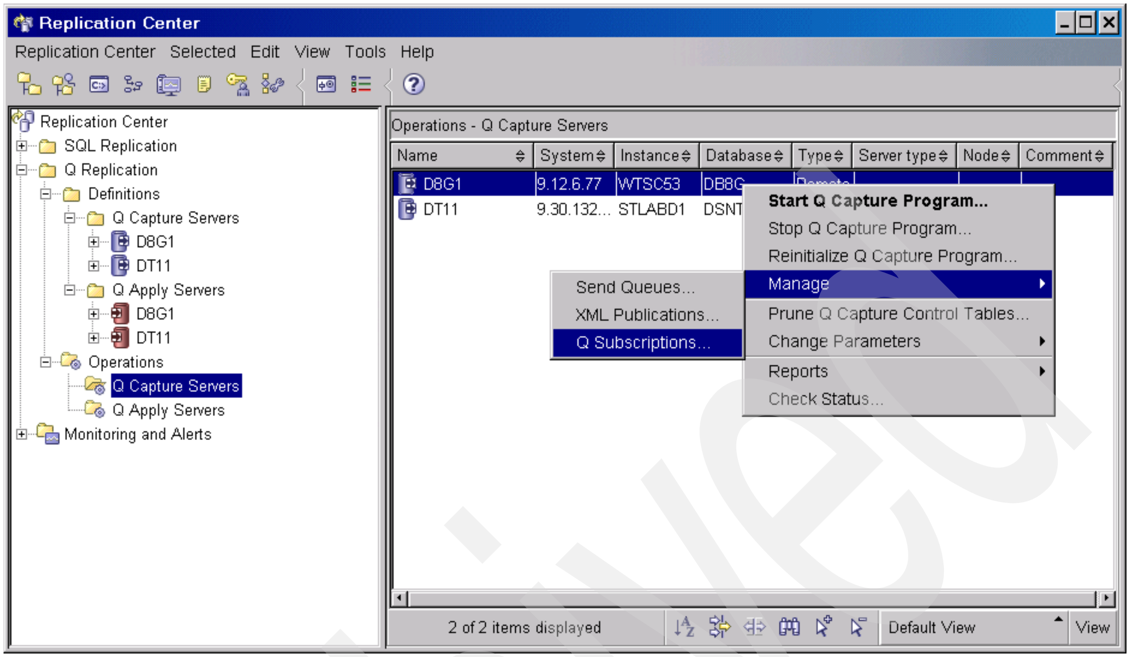


Figure 3-72 Manage Q Subscriptions - D8G1

Right-click the **D8G1** server name, which is the target, and select **Manage** and then **Q Subscriptions...**, as shown in Figure 3-72. This displays Figure 3-73, which lists all the Q subscriptions and their status.

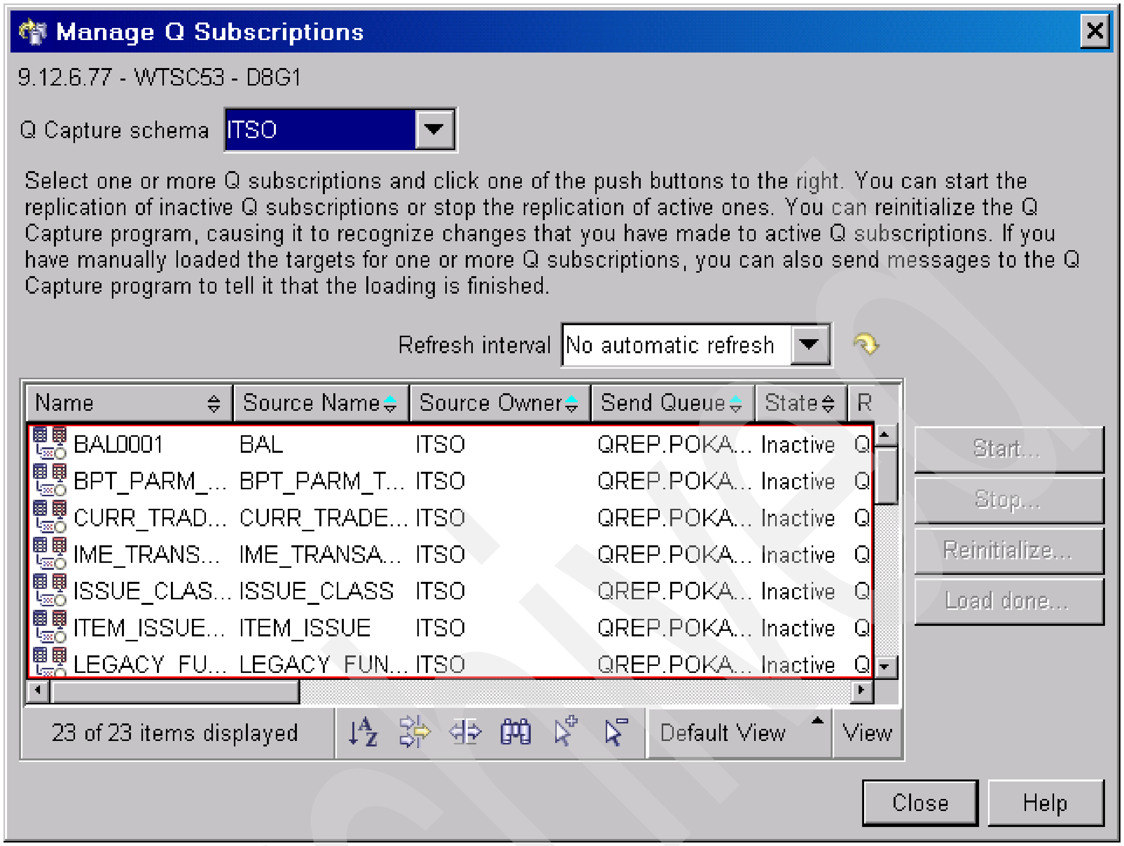


Figure 3-73 Q Subscriptions (D8G1) and their states

Here again, the Refresh interval field and the yellow arrow next to it can be used to monitor the changing state of the subscriptions listed. The icons next to the Name field also indicate the state of the Q subscription. The Inactive condition under the State column will remain until the table is loaded and the spill queue is emptied.

> **Attention:** Now start the Q Capture tasks on DT11 and D8G1 using the commands shown in Example 3-22 on page 174 and Example 3-25 on page 176, respectively, followed by a start of the Q Apply tasks on DT11 and D8G1 using the commands shown in Example 3-29 on page 177 and Example 3-32 on page 178, respectively.

Click **Close** in Figure 3-73 to return to the previous screen (Figure 3-69 on page 180), and then bring up the Q subscriptions screen for DT11, as shown in Figure 3-74.

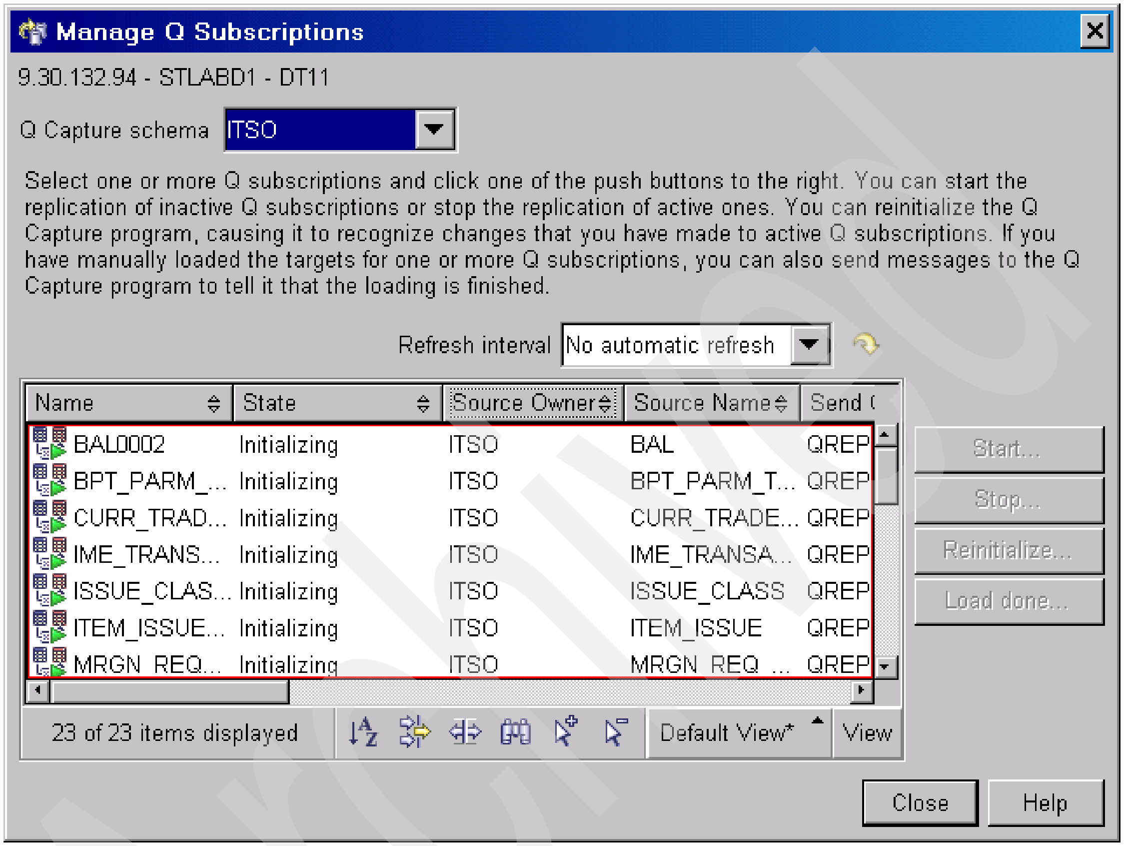


Figure 3-74 Q Subscriptions (DT11) and their states

Figure 3-74 shows the subscriptions to be in an Initializing state with a change in the icon as well. Q subscription messages are being processed in DT11 (source of the load) and sent to initialize the load phase of Q Apply on D8G1 (target of the load). After the initialization message is processed for a subscription, the Q Capture program inserts a message in its output log (MVS job output), as well as in the Capture log dataset, as shown in Example 3-34.

Example 3-34 Q Capture log contents - Ready to load

```
19.13.30 STC01107  ASN7017I  "Q Capture" : "ITSO" : "WorkerThread" : The target
                   table "ITSO.BAL" is ready to be loaded from source table
                   "ITSO.BAL" for XML publication or Q subscription "BAL0002".
```

Example 3-35 shows the message put out by the Q Apply program in the Q Apply log (*not* the Q Apply program job output) when it receives the message from Q Capture to begin the load.

Example 3-35 Q Apply log contents - Load method

```
2004-11-16-22.13.36.978150 <invokeLoad> ASN7528I  "Q Apply" : "ITSO" :
"BR00000SP001" : The Q Apply program for the Q subscription "BAL0001" (receive
queue
   "QREP.STLA.TO.POKA.RECVQ", replication queue map "QMAP_STL_TO_POK")
   Will use the "LOAD from CURSOR" utility to load table "ITSO.BAL".
```

Figure 3-75 shows the Loading state on some of the subscriptions in the DT11 server.

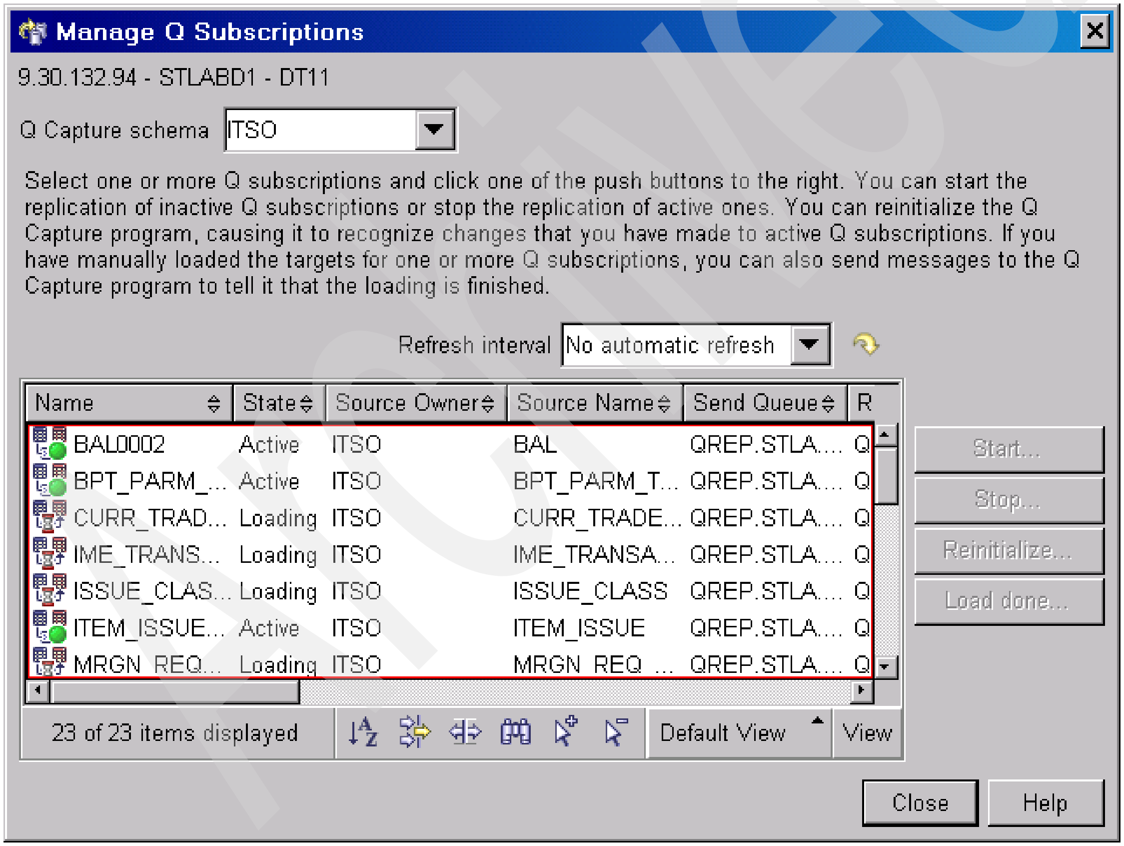


Figure 3-75 Q Subscriptions (DT11) and their states

Figure 3-75 shows some subscriptions as being Active, indicating that the loading has completed and the messages in the spill queue on the D8G1 server have been processed. An activation message is received on the Q Capture job output, as shown in Example 3-36.

Example 3-36 Activation message in Q Capture job output

```
19.13.56 STC01107  ASN7010I  "Q Capture" : "ITSO" : "WorkerThread" : The progra
                   successfully activated  XML publication or Q subscription
                   "BAL0001"  (send queue "QREP.STLA.TO.POKA.SENDQ", publishing
                   replication queue map "QMAP_STL_TO_POK") for source table
                   "ITSO.BAL".
```

The Loading state (in Figure 3-75) on some subscriptions indicates that the process is ongoing.

The Inactive state (in Figure 3-76) indicates that the load has failed for some reason.

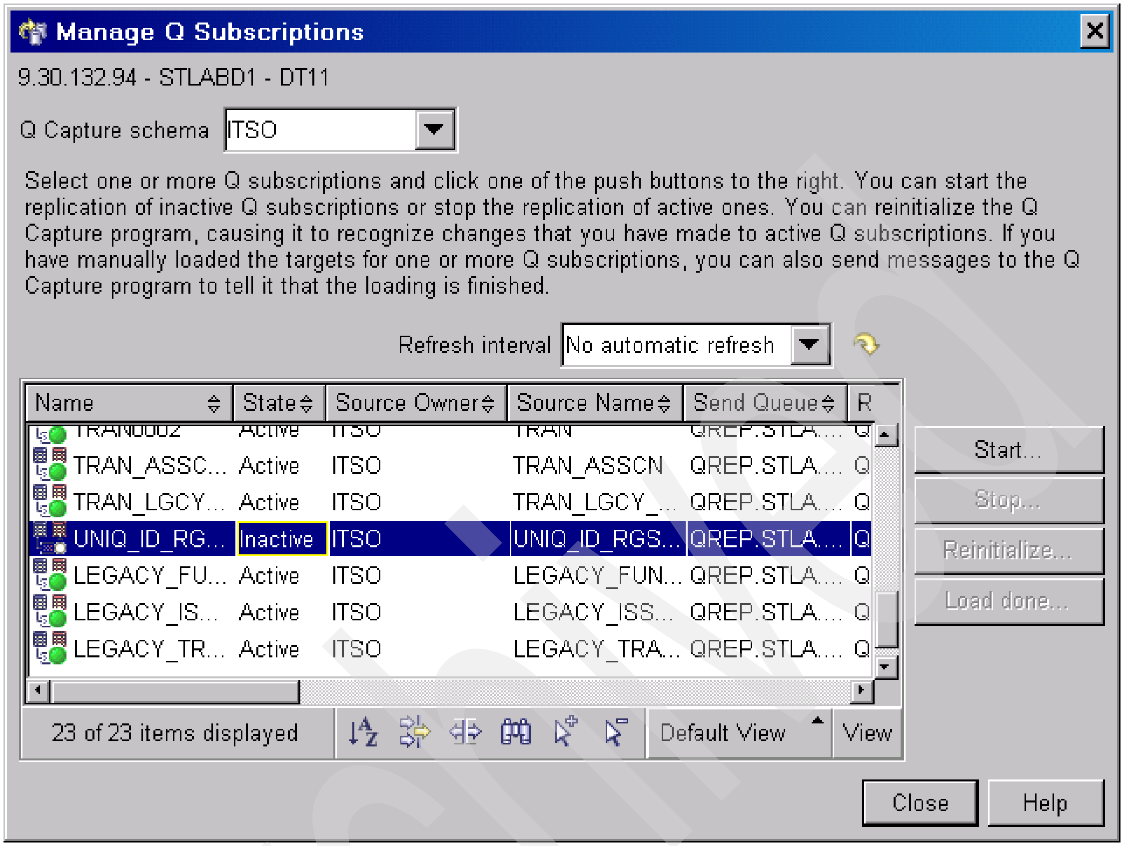


Figure 3-76 Q Subscriptions (DT11) and their states

Locate the error from the appropriate LOADTRC datasets (DB2PROD.DB2PROD.<D8G1.ITSO.S00000nn.LOADTRC) on the target server WTSC53, and correct the cause of the failure. In this example the error was due to contention with the other load processes.

Note: Q replication documentation cautions the reader that the internal loader (automatic load) on MVS uses parallel loads. In our scenarios, 23 tables are being loaded at the same time. The documentation suggests serializing these loads by deactivating them before Q Capture startup (by selecting **Stop** from Manage Q Subscriptions—not shown here), and starting them one at a time to avoid contention between parallel loads. Our tables only contained a few hundred rows in each table being loaded.

After the error has been corrected, highlight the inactive subscription, and click **Start** to request activation of the subscription, as shown in Figure 3-77.

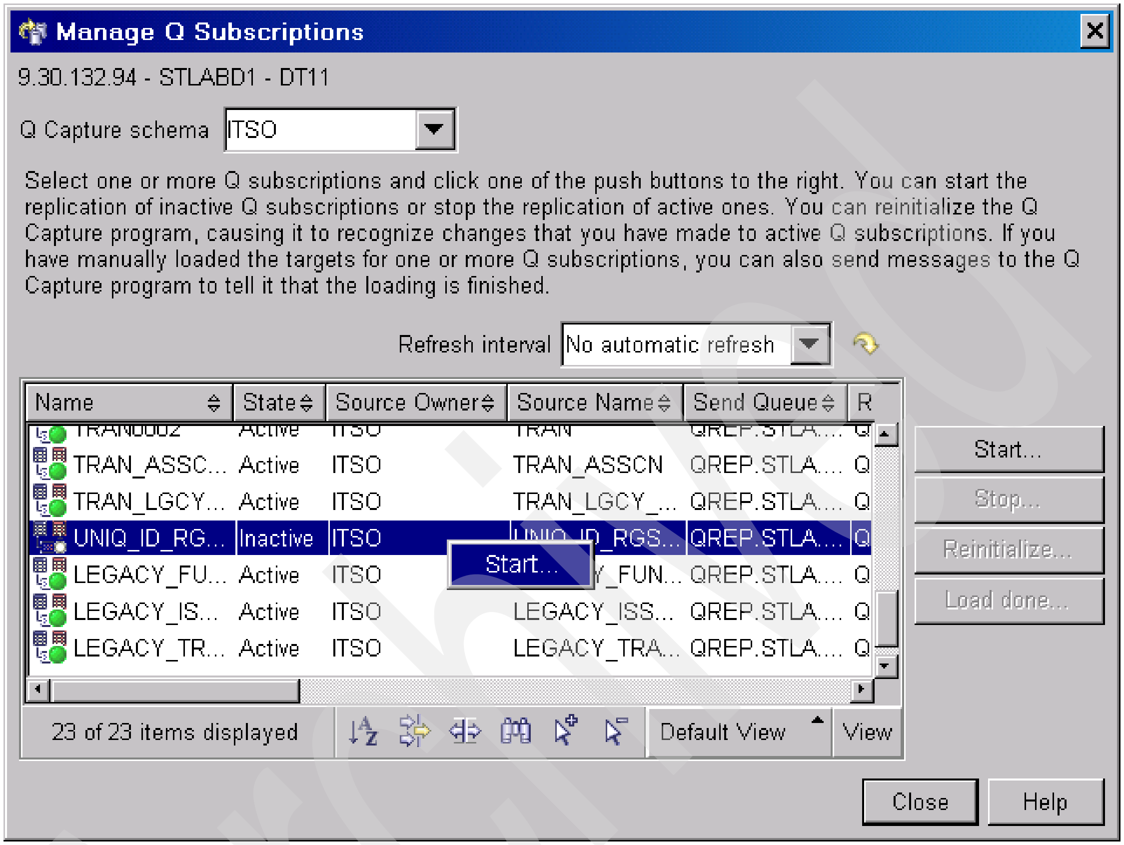


Figure 3-77 Start an inactive subscription

The start action causes a CAPSTART signal to be generated via an SQL INSERT statement, as shown in Figure 3-78.

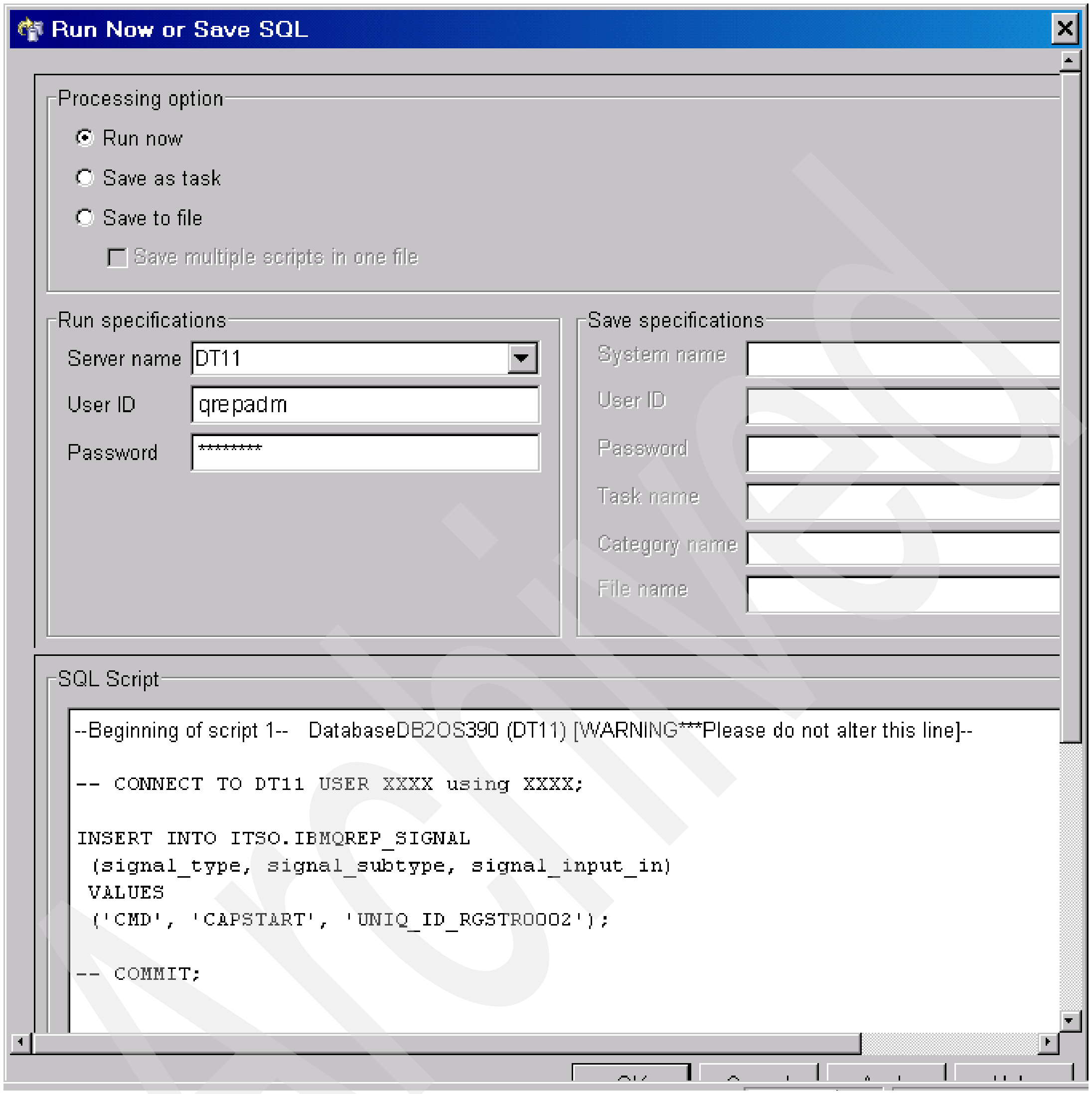


Figure 3-78 Run Now or Save SQL

We selected **Run now** and clicked **OK** for the script to be executed immediately. Figure 3-79 displays the message indicating that the SQL scripts ran successfully.

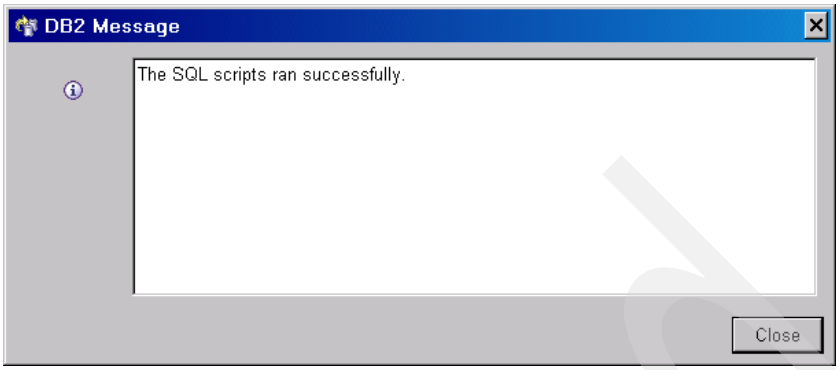


Figure 3-79 SQL scripts ran successfully message

Click **Close** to return to the screen (shown in Figure 3-80) that indicates that the subscription is now in Active state.

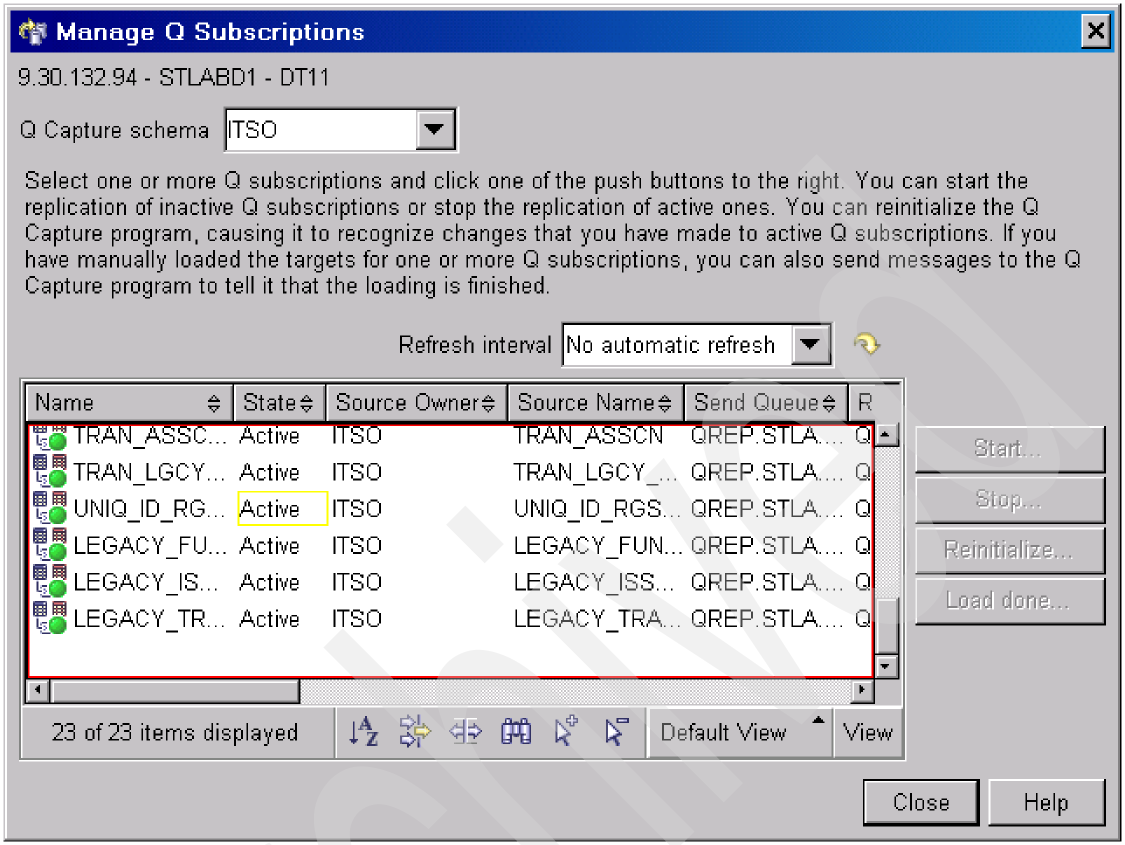


Figure 3-80 Q Subscriptions (DT11) and their states

Repeat the above process until all subscriptions are active, at which point initialization is complete and normal bidirectional replication activity commences.

> **Attention:** The DBA should ensure that all the subscriptions are active following a successful load process; otherwise, the spill queues will continue to fill and possibly result in error conditions.

Verify successful implementation

Since all subscriptions were created to activate automatically and load automatically, the start of Q Capture and Q Apply programs will load the target tables and mark the subscriptions as active (see Figure 3-68 on page 179 through Figure 3-78 on page 190). One can additionally verify that the target tables contain the same number of records on each side prior to releasing applications to run against them by doing 'SELECT COUNT(*) from xxxx.yyyy;' queries in SPUFI on each server for each table being replicated.

At any time after implementation, tables can be compared to identify any inconsistencies by running the compare utility ASNTDIFF to identify any differences in table content.

Step 6i: Perform manual load if appropriate

Since we chose automatic loading of the tables involved in the subscriptions, this step does not apply in our scenario.

> **Attention:** This completes the setup of the bidirectional Q replication implementation for Ithaca.

3.5.7 Step 6: Configure and activate Q replication using commands

WebSphere II Q replication configuration may be performed using GUI as described in 3.5.6, “Step 6: Configure and activate Q replication using GUI” on page 111; or by using ASNCLP commands to generate SQL scripts.

As mentioned earlier, ASNCLP does not run directly in a z/OS environment; therefore, the SQL scripts cannot be executed from ASNCLP as they are generated. ASNCLP may be executed on UNIX or Windows to generate SQL that can then be ported to z/OS and executed under SPUFI or DSNTEP2.

> **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 on the workstation, the user environment must be configured to provide support for Java, as shown in Example 3-37.

Example 3-37 Set up Java environment on the workstation

```
set CP=%CP%;c:\progra~1\ibm\sqllib\java\Common.jar;
set CP=%CP%;c:\progra~1\ibm\sqllib\tools\db2cmn.jar;
set CP=%CP%;c:\progra~1\ibm\sqllib\tools\db2replapis.jar;
set CP=%CP%;c:\progra~1\ibm\sqllib\tools\db2qreplapis.jar;
set CP=%CP%;c:\progra~1\ibm\sqllib\tools\jt400.jar;
set CLASSPATH=%CLASSPATH%;%CP%;
```

To ensure that the environment has been set up properly, run the **asnclp** command. If the `>Repl` command line appears, the setup was successful. Enter **quit** to exit the Repl> command line, as shown in Example 3-38.

Example 3-38 asnclp setup configuration test from a workstation C: prompt

```
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 3-39.

Example 3-39 Execute commands from a file

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

The ASNCLP commands in the following sections are categorized according to the following tasks:

- Create the Q Capture and Q Apply control tables.
- Create the replication queue maps.
- Create the subscriptions.

Create the Q Capture and Q Apply control tables

Example 3-40 lists the ASNCLP commands for creating the Q Capture and Q Apply control tables on logical database QREPITSO in subsystem DT11 on STLABD1, while Example 3-42 does the same on logical database QREPITSO in subsystem D8G1 on WTSC53. These commands should be saved in a file that is then executed using the command described in Example 3-39 on page 194.

Example 3-40 Create the Q Capture and Q Apply control tables on DT11

```
####################################################
# The following commands create the control tables
####################################################
ASNCLP SESSION SET TO Q REPLICATION;
SET OUTPUT MULTIDIR;
#############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 3-1 on page 85, Table 3-2 on page 86 and Table 3-3 on page 88
############################################################
```

```
##DT11 from reference A.7a in Table 3-1 on page 85
##QREPADM from reference A.9 in Table 3-1 on page 85
##QREPITSO from reference A.10 in Table 3-1 on page 85
##MQS1 from reference A.11 in Table 3-2 on page 86
##Capture schema ITSO from reference A.22 in Table 3-3 on page 88
##QREP.STLA.RESTARTQ from reference A.16 in Table 3-2 on page 86
##//DB2PROD from reference A.24 in Table 3-3 on page 88
##QREP.STLA.ADMINQ from reference A.17 in Table 3-2 on page 86
#############################################################
SET SERVER CAPTURE TO DB DT11 ID QREPADM PASSWORD "xxxxxx";
SET QMANAGER "MQS1" FOR CAPTURE SCHEMA;
SET CAPTURE SCHEMA SOURCE ITSO;
SET RUN SCRIPT LATER;
CREATE CONTROL TABLES FOR CAPTURE SERVER USING RESTARTQ "QREP.STLA.RESTARTQ"
ADMINQ "QREP.STLA.ADMINQ" STARTMODE WARMSI CAPTURE PATH "//DB2PROD" IN ZOS PAGE
LOCK DB "QREPITSO" NAMING PREFIX QC CREATE ROW LOCK DB "QREPITSO" NAMING PREFIX
QC CREATE;
############################################################
##DT11 from reference A.7a in Table 3-1 on page 85
##QREPADM from reference A.9 in Table 3-1 on page 85
##QREPITSO from reference A.10 in Table 3-1 on page 85
##MQS1 from reference A.11 in Table 3-2 on page 86
##//DB2PROD from reference A.25 in Table 3-3 on page 88
##Apply schema ITSO from reference A.23 in Table 3-3 on page 88
#############################################################
SET SERVER TARGET TO DB DT11 ID QREPADM PASSWORD "xxxxxx";
SET QMANAGER "MQS1" FOR APPLY SCHEMA;
SET APPLY SCHEMA ITSO;
SET RUN SCRIPT LATER;
CREATE CONTROL TABLES FOR APPLY SERVER USING APPLY PATH "//DB2PROD"  IN ZOS
PAGE LOCK DB "QREPITSO" NAMING PREFIX QA CREATE ROW LOCK DB "QREPITSO" NAMING
PREFIX QA CREATE;
```

The generated SQL[2] shown in Example 3-41 can be ported to z/OS for execution under SPUFI or DSNTEP2.

Example 3-41 Generated SQL for DT11 control tables

```
-- DatabaseDB2OS390 (DT11)

-- CONNECT TO DT11 USER XXXX using XXXX;
```

[2] When ASNCLP specifies SET RUN SCRIPT LATER, the SQL that is generated by ASNCLP will be stored in a file on the workstation, in the same directory that ASNCLP was executed from. It will have the name of the DB from the SET SERVER Statement with a .sql extension, for example, DT11.sql. If the ASNCLP generates sql for two servers there will be two files generated.

```
CREATE TABLESPACE QCITSOCP IN QREPITSO
  SEGSIZE 4
  LOCKSIZE PAGE
  CLOSE NO;



CREATE TABLESPACE QCITSOCR IN QREPITSO
  SEGSIZE 4
  LOCKSIZE ROW
  CLOSE NO;


CREATE TABLE ITSO.IBMQREP_CAPPARMS
(
 QMGR VARCHAR(48) NOT NULL,
 REMOTE_SRC_SERVER VARCHAR(18),
 RESTARTQ VARCHAR(48) NOT NULL,
 ADMINQ VARCHAR(48) NOT NULL,
 STARTMODE VARCHAR(6) NOT NULL WITH DEFAULT 'WARMSI',
 MEMORY_LIMIT INTEGER NOT NULL WITH DEFAULT 32,
 COMMIT_INTERVAL INTEGER NOT NULL WITH DEFAULT 500,
 AUTOSTOP CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 MONITOR_INTERVAL INTEGER NOT NULL WITH DEFAULT 300,
 MONITOR_LIMIT INTEGER NOT NULL WITH DEFAULT 10080,
 TRACE_LIMIT INTEGER NOT NULL WITH DEFAULT 10080,
 SIGNAL_LIMIT INTEGER NOT NULL WITH DEFAULT 10080,
 PRUNE_INTERVAL INTEGER NOT NULL WITH DEFAULT 300,
 SLEEP_INTERVAL INTEGER NOT NULL WITH DEFAULT 5000,
 LOGREUSE CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 LOGSTDOUT CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 TERM CHARACTER(1) NOT NULL WITH DEFAULT 'Y',
 CAPTURE_PATH VARCHAR(1040) WITH DEFAULT NULL,
 ARCH_LEVEL CHARACTER(4) NOT NULL WITH DEFAULT '0802',
 CONSTRAINT CC_STARTMODE CHECK(STARTMODE IN ('COLD','WARMSI','WARMSA'
,'WARMNS')),
 CONSTRAINT CC_MEMORY_LIMIT CHECK(MEMORY_LIMIT >= 1 AND MEMORY_LIMIT
 <= 1000),
 CONSTRAINT CC_COMMIT_INTERVAL CHECK(COMMIT_INTERVAL  >= 100 AND
 COMMIT_INTERVAL <= 600000),
 CONSTRAINT CC_AUTOSTOP CHECK(AUTOSTOP IN ('Y','N')),
 CONSTRAINT CC_MON_INTERVAL CHECK(MONITOR_INTERVAL >= 1 AND
 MONITOR_INTERVAL <= 2147483647),
 CONSTRAINT CC_MON_LIMIT CHECK(MONITOR_LIMIT >= 1 AND MONITOR_LIMIT
 <= 2147483647),
 CONSTRAINT CC_TRACE_LIMT CHECK(TRACE_LIMIT >= 1 AND TRACE_LIMIT <=
 2147483647),
 CONSTRAINT CC_SIGNAL_LIMIT CHECK(SIGNAL_LIMIT >= 1 AND SIGNAL_LIMIT
 <= 2147483647),
```

```
 CONSTRAINT CC_PRUNE_INTERVAL CHECK(PRUNE_INTERVAL >= 1 AND
 PRUNE_INTERVAL <= 2147483647),
 CONSTRAINT CC_LOGREUSE CHECK(LOGREUSE IN ('Y','N')),
 CONSTRAINT CC_LOGSTDOUT CHECK(LOGSTDOUT IN ('Y','N')),
 CONSTRAINT CC_TERM CHECK(TERM IN ('Y','N')),
 CONSTRAINT CC_SLEEP_INTERVAL CHECK(SLEEP_INTERVAL >= 1 AND
 SLEEP_INTERVAL <= 2147483647)
)
 IN QREPITSO.QCITSOCP;


CREATE UNIQUE INDEX ITSO.IX1CQMGRCOL ON ITSO.IBMQREP_CAPPARMS
(
 QMGR ASC
);


CREATE TABLE ITSO.IBMQREP_SENDQUEUES
(
 PUBQMAPNAME VARCHAR(128) NOT NULL,
 SENDQ VARCHAR(48) NOT NULL,
 RECVQ VARCHAR(48),
 MESSAGE_FORMAT CHARACTER(1) NOT NULL WITH DEFAULT 'C',
 MSG_CONTENT_TYPE CHARACTER(1) NOT NULL WITH DEFAULT 'T',
 STATE CHARACTER(1) NOT NULL WITH DEFAULT 'A',
 STATE_TIME TIMESTAMP NOT NULL WITH DEFAULT,
 STATE_INFO CHARACTER(8),
 ERROR_ACTION CHARACTER(1) NOT NULL WITH DEFAULT 'S',
 HEARTBEAT_INTERVAL INTEGER NOT NULL WITH DEFAULT 60,
 MAX_MESSAGE_SIZE INTEGER NOT NULL WITH DEFAULT 64,
 APPLY_SERVER VARCHAR(18),
 APPLY_ALIAS VARCHAR(8),
 APPLY_SCHEMA VARCHAR(128),
 DESCRIPTION VARCHAR(254),
 PRIMARY KEY(SENDQ),
 CONSTRAINT CC_MSG_FORMAT CHECK(MESSAGE_FORMAT IN ('X','C', 'J')),
 CONSTRAINT CC_MSG_CONT_TYPE CHECK(MSG_CONTENT_TYPE IN ('T','R')),
 CONSTRAINT CC_SENDQ_STATE CHECK(STATE IN ('A','I')),
 CONSTRAINT CC_QERRORACTION CHECK(ERROR_ACTION IN ('I','S')),
 CONSTRAINT CC_HTBEAT_INTERVAL CHECK(HEARTBEAT_INTERVAL >= 0 AND
 HEARTBEAT_INTERVAL <= 32767)
)
 IN QREPITSO.QCITSOCR;


CREATE UNIQUE INDEX ITSO.PKIBMQREP_SENDQUEUES ON
 ITSO.IBMQREP_SENDQUEUES
(
 SENDQ
```

```
);


CREATE UNIQUE INDEX ITSO.IX1PUBMAPCOL ON ITSO.IBMQREP_SENDQUEUES
(
 PUBQMAPNAME ASC
);


CREATE TABLE ITSO.IBMQREP_SUBS
(
 SUBNAME VARCHAR(132) NOT NULL,
 SOURCE_OWNER VARCHAR(128) NOT NULL,
 SOURCE_NAME VARCHAR(128) NOT NULL,
 TARGET_SERVER VARCHAR(18),
 TARGET_ALIAS VARCHAR(8),
 TARGET_OWNER VARCHAR(128),
 TARGET_NAME VARCHAR(128),
 TARGET_TYPE INTEGER,
 APPLY_SCHEMA VARCHAR(128),
 SENDQ VARCHAR(48) NOT NULL,
 SEARCH_CONDITION VARCHAR(2048) WITH DEFAULT NULL,
 SUB_ID INTEGER WITH DEFAULT NULL,
 SUBTYPE CHARACTER(1) NOT NULL WITH DEFAULT 'U',
 ALL_CHANGED_ROWS CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 BEFORE_VALUES CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 CHANGED_COLS_ONLY CHARACTER(1) NOT NULL WITH DEFAULT 'Y',
 HAS_LOADPHASE CHARACTER(1) NOT NULL WITH DEFAULT 'I',
 STATE CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 STATE_TIME TIMESTAMP NOT NULL WITH DEFAULT,
 STATE_INFO CHARACTER(8),
 STATE_TRANSITION VARCHAR(256) FOR BIT DATA,
 SUBGROUP VARCHAR(30) WITH DEFAULT NULL,
 SOURCE_NODE SMALLINT NOT NULL WITH DEFAULT 0,
 TARGET_NODE SMALLINT NOT NULL WITH DEFAULT 0,
 GROUP_MEMBERS CHARACTER(254) FOR BIT DATA WITH DEFAULT NULL,
 OPTIONS_FLAG CHARACTER(4) NOT NULL WITH DEFAULT 'NNNN',
 SUPPRESS_DELETES CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 DESCRIPTION VARCHAR(200),
 TOPIC VARCHAR(256),
 PRIMARY KEY(SUBNAME),
 CONSTRAINT FKSENDQ FOREIGN KEY(SENDQ) REFERENCES
 ITSO.IBMQREP_SENDQUEUES(SENDQ),
 CONSTRAINT CC_SUBTYPE CHECK(SUBTYPE IN ('U','B','P')),
 CONSTRAINT CC_ALL_CHGD_ROWS CHECK(ALL_CHANGED_ROWS IN ('Y','N')),
 CONSTRAINT CC_BEFORE_VALUES CHECK(BEFORE_VALUES IN ('Y','N')),
 CONSTRAINT CC_CHGD_COLS_ONLY CHECK(CHANGED_COLS_ONLY IN ('Y','N')),
 CONSTRAINT CC_HAS_LOADPHASE CHECK(HAS_LOADPHASE IN ('N','I','E')),
 CONSTRAINT CC_SUBS_STATE CHECK(STATE IN ('L','A','I','T','G','N')),
```

```
 CONSTRAINT CC_SUPPRESS_DELS CHECK(SUPPRESS_DELETES IN ('Y','N'))
)
 IN QREPITSO.QCITSOCR;


CREATE UNIQUE INDEX ITSO.PKIBMQREP_SUBS ON ITSO.IBMQREP_SUBS
(
 SUBNAME
);


CREATE TABLE ITSO.IBMQREP_SRC_COLS
(
 SUBNAME VARCHAR(132) NOT NULL,
 SRC_COLNAME VARCHAR(128) NOT NULL,
 IS_KEY SMALLINT NOT NULL WITH DEFAULT 0,
 PRIMARY KEY(SUBNAME, SRC_COLNAME),
 CONSTRAINT FKSUBS FOREIGN KEY(SUBNAME) REFERENCES ITSO.IBMQREP_SUBS
(SUBNAME)
)
 IN QREPITSO.QCITSOCP;


CREATE UNIQUE INDEX ITSO.PKIBMQREP_SRC_COLS ON ITSO.IBMQREP_SRC_COLS
(
 SUBNAME,
 SRC_COLNAME
);


CREATE TABLE ITSO.IBMQREP_SRCH_COND
(
 ASNQREQD INTEGER
)
 IN QREPITSO.QCITSOCP;


CREATE TABLE ITSO.IBMQREP_SIGNAL
(
 SIGNAL_TIME TIMESTAMP NOT NULL WITH DEFAULT,
 SIGNAL_TYPE VARCHAR(30) NOT NULL,
 SIGNAL_SUBTYPE VARCHAR(30),
 SIGNAL_INPUT_IN VARCHAR(500),
 SIGNAL_STATE CHARACTER(1) NOT NULL WITH DEFAULT 'P',
 SIGNAL_LSN CHARACTER(10) FOR BIT DATA,
 PRIMARY KEY(SIGNAL_TIME),
 CONSTRAINT CC_SIGNAL_TYPE CHECK(SIGNAL_TYPE IN ('CMD','USER')),
 CONSTRAINT CC_SIGNAL_STATE CHECK(SIGNAL_STATE IN ('P','R','C','F'))
)
```

```
 IN QREPITSO.QCITSOCR
 DATA CAPTURE CHANGES;


CREATE UNIQUE INDEX ITSO.PKIBMQREP_SIGNAL ON ITSO.IBMQREP_SIGNAL
(
 SIGNAL_TIME
);


CREATE TABLE ITSO.IBMQREP_CAPTRACE
(
 OPERATION CHARACTER(8) NOT NULL,
 TRACE_TIME TIMESTAMP NOT NULL,
 DESCRIPTION VARCHAR(1024) NOT NULL
)
 IN QREPITSO.QCITSOCR;


CREATE TABLE ITSO.IBMQREP_CAPMON
(
 MONITOR_TIME TIMESTAMP NOT NULL,
 CURRENT_LOG_TIME TIMESTAMP NOT NULL,
 CAPTURE_IDLE INTEGER NOT NULL,
 CURRENT_MEMORY INTEGER NOT NULL,
 ROWS_PROCESSED INTEGER NOT NULL,
 TRANS_SKIPPED INTEGER NOT NULL,
 TRANS_PROCESSED INTEGER NOT NULL,
 TRANS_SPILLED INTEGER NOT NULL,
 MAX_TRANS_SIZE INTEGER NOT NULL,
 QUEUES_IN_ERROR INTEGER NOT NULL,
 RESTART_SEQ CHARACTER(10) FOR BIT DATA NOT NULL,
 CURRENT_SEQ CHARACTER(10) FOR BIT DATA NOT NULL,
 PRIMARY KEY(MONITOR_TIME)
)
 IN QREPITSO.QCITSOCP;


CREATE UNIQUE INDEX ITSO.PKIBMQREP_CAPMON ON ITSO.IBMQREP_CAPMON
(
 MONITOR_TIME
);


CREATE TABLE ITSO.IBMQREP_CAPQMON
(
 MONITOR_TIME TIMESTAMP NOT NULL,
 SENDQ VARCHAR(48) NOT NULL,
 ROWS_PUBLISHED INTEGER NOT NULL,
```

```
 TRANS_PUBLISHED INTEGER NOT NULL,
 CHG_ROWS_SKIPPED INTEGER NOT NULL,
 DELROWS_SUPPRESSED INTEGER NOT NULL,
 ROWS_SKIPPED INTEGER NOT NULL,
 PRIMARY KEY(MONITOR_TIME, SENDQ)
)
 IN QREPITSO.QCITSOCP;


CREATE UNIQUE INDEX ITSO.PKIBMQREP_CAPQMON ON ITSO.IBMQREP_CAPQMON
(
 MONITOR_TIME,
 SENDQ
);


CREATE TABLE ITSO.IBMQREP_CAPENQ
(
 LOCKNAME INTEGER
)
 IN QREPITSO.QCITSOCR;


CREATE TABLE ITSO.IBMQREP_ADMINMSG
(
 MQMSGID CHARACTER(24) FOR BIT DATA NOT NULL,
 MSG_TIME TIMESTAMP NOT NULL WITH DEFAULT,
 PRIMARY KEY(MQMSGID)
)
 IN QREPITSO.QCITSOCP;


CREATE UNIQUE INDEX ITSO.PKIBMQREP_ADMINMSG ON ITSO.IBMQREP_ADMINMSG
(
 MQMSGID
);


INSERT INTO ITSO.IBMQREP_CAPPARMS
 (qmgr, restartq, adminq, startmode, memory_limit, commit_interval,
 autostop, monitor_interval, monitor_limit, trace_limit, signal_limit,
 prune_interval, sleep_interval, logreuse, logstdout, term,
 capture_path, arch_level)
 VALUES
 ('MQS1', 'QREP.STLA.RESTARTQ', 'QREP.STLA.ADMINQ', 'WARMSI', 32, 500
, 'N', 300, 10080, 10080, 10080, 300, 5000, 'N', 'N', 'Y', '//DB2PROD'
, '0802');
```

```
-- COMMIT;


-- DatabaseDB2OS390 (DT11)

-- CONNECT TO DT11 USER XXXX using XXXX;



CREATE TABLESPACE QAASNAP IN QREPITSO
  SEGSIZE 4
  LOCKSIZE PAGE
  CLOSE NO;



CREATE TABLESPACE QAASNAR IN QREPITSO
  SEGSIZE 4
  LOCKSIZE ROW
  CLOSE NO;


CREATE TABLE ITSO.IBMQREP_APPLYPARMS
(
 QMGR VARCHAR(48) NOT NULL,
 MONITOR_LIMIT INTEGER NOT NULL WITH DEFAULT 10080,
 TRACE_LIMIT INTEGER NOT NULL WITH DEFAULT 10080,
 MONITOR_INTERVAL INTEGER NOT NULL WITH DEFAULT 300,
 PRUNE_INTERVAL INTEGER NOT NULL WITH DEFAULT 300,
 AUTOSTOP CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 LOGREUSE CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 LOGSTDOUT CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 APPLY_PATH VARCHAR(1040) WITH DEFAULT NULL,
 ARCH_LEVEL CHARACTER(4) NOT NULL WITH DEFAULT '0802',
 TERM CHARACTER(1) NOT NULL WITH DEFAULT 'Y',
 PWDFILE VARCHAR(48) WITH DEFAULT NULL,
 DEADLOCK_RETRIES INTEGER NOT NULL WITH DEFAULT 3,
 CONSTRAINT CA_MON_LIMIT CHECK(MONITOR_LIMIT >= 1 AND MONITOR_LIMIT
 <= 2147483647),
 CONSTRAINT CA_TRACE_LIMT CHECK(TRACE_LIMIT >= 1 AND TRACE_LIMIT <=
 2147483647),
 CONSTRAINT CA_MON_INTERVAL CHECK(MONITOR_INTERVAL >= 1 AND
 MONITOR_INTERVAL <= 2147483647),
 CONSTRAINT CA_PRUNE_INTERVAL CHECK(PRUNE_INTERVAL >= 1 AND
 PRUNE_INTERVAL <= 2147483647),
 CONSTRAINT CA_AUTOSTOP CHECK(AUTOSTOP IN ('Y','N')),
 CONSTRAINT CA_LOGREUSE CHECK(LOGREUSE IN ('Y','N')),
 CONSTRAINT CA_LOGSTDOUT CHECK(LOGSTDOUT IN ('Y','N')),
 CONSTRAINT CA_TERM CHECK(TERM IN ('Y','N')),
```

```
 CONSTRAINT CA_RETRIES CHECK(DEADLOCK_RETRIES  >= 3 AND
 DEADLOCK_RETRIES <= 2147483647)
)
 IN QREPITSO.QAASNAP;


CREATE UNIQUE INDEX ITSO.IX1AQMGRCOL ON ITSO.IBMQREP_APPLYPARMS
(
 QMGR ASC
);


CREATE TABLE ITSO.IBMQREP_RECVQUEUES
(
 REPQMAPNAME VARCHAR(128) NOT NULL,
 RECVQ VARCHAR(48) NOT NULL,
 SENDQ VARCHAR(48) WITH DEFAULT NULL,
 ADMINQ VARCHAR(48) NOT NULL,
 NUM_APPLY_AGENTS INTEGER NOT NULL WITH DEFAULT 16,
 MEMORY_LIMIT INTEGER NOT NULL WITH DEFAULT 32,
 CAPTURE_SERVER VARCHAR(18) NOT NULL,
 CAPTURE_ALIAS VARCHAR(8) NOT NULL,
 CAPTURE_SCHEMA VARCHAR(30) NOT NULL WITH DEFAULT 'ASN',
 STATE CHARACTER(1) NOT NULL WITH DEFAULT 'A',
 STATE_TIME TIMESTAMP NOT NULL WITH DEFAULT,
 STATE_INFO CHARACTER(8),
 DESCRIPTION VARCHAR(254),
 PRIMARY KEY(RECVQ),
 CONSTRAINT CC_SENDQ_STATE CHECK(STATE IN ('A','I'))
)
 IN QREPITSO.QAASNAR;


CREATE UNIQUE INDEX ITSO.PKIBMQREP_RECVQUEUES ON
 ITSO.IBMQREP_RECVQUEUES
(
 RECVQ
);


CREATE UNIQUE INDEX ITSO.IX1REPMAPCOL ON ITSO.IBMQREP_RECVQUEUES
(
 REPQMAPNAME ASC
);


CREATE TABLE ITSO.IBMQREP_TARGETS
(
 SUBNAME VARCHAR(132) NOT NULL,
```

```
 RECVQ VARCHAR(48) NOT NULL,
 SUB_ID INTEGER WITH DEFAULT NULL,
 SOURCE_SERVER VARCHAR(18) NOT NULL,
 SOURCE_ALIAS VARCHAR(8) NOT NULL,
 SOURCE_OWNER VARCHAR(128) NOT NULL,
 SOURCE_NAME VARCHAR(128) NOT NULL,
 SRC_NICKNAME_OWNER VARCHAR(128),
 SRC_NICKNAME VARCHAR(128),
 TARGET_OWNER VARCHAR(128) NOT NULL,
 TARGET_NAME VARCHAR(128) NOT NULL,
 TARGET_TYPE INTEGER NOT NULL WITH DEFAULT 1,
 FEDERATED_TGT_SRVR VARCHAR(18) WITH DEFAULT NULL,
 STATE CHARACTER(1) NOT NULL WITH DEFAULT 'I',
 STATE_TIME TIMESTAMP NOT NULL WITH DEFAULT,
 STATE_INFO CHARACTER(8),
 SUBTYPE CHARACTER(1) NOT NULL WITH DEFAULT 'U',
 CONFLICT_RULE CHARACTER(1) NOT NULL WITH DEFAULT 'K',
 CONFLICT_ACTION CHARACTER(1) NOT NULL WITH DEFAULT 'I',
 ERROR_ACTION CHARACTER(1) NOT NULL WITH DEFAULT 'Q',
 SPILLQ VARCHAR(48) WITH DEFAULT NULL,
 OKSQLSTATES VARCHAR(128) WITH DEFAULT NULL,
 SUBGROUP VARCHAR(30) WITH DEFAULT NULL,
 SOURCE_NODE SMALLINT NOT NULL WITH DEFAULT 0,
 TARGET_NODE SMALLINT NOT NULL WITH DEFAULT 0,
 GROUP_INIT_ROLE CHARACTER(1) WITH DEFAULT NULL,
 HAS_LOADPHASE CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 LOAD_TYPE SMALLINT NOT NULL WITH DEFAULT 0,
 DESCRIPTION VARCHAR(254),
 SEARCH_CONDITION VARCHAR(2048) WITH DEFAULT NULL,
 CONSTRAINT CA_TARGTBL_STATE CHECK(STATE IN ('L','A','I','E','D','F'
,'T')),
 CONSTRAINT CA_UPDATEANY CHECK(SUBTYPE IN ('U','B','P')),
 CONSTRAINT CA_CONFLICTACTION CHECK(CONFLICT_ACTION IN ('F','I','D'
,'S','Q')),
 CONSTRAINT CA_ERRORACTION CHECK(ERROR_ACTION IN ('D','S','Q')),
 CONSTRAINT CA_UPANY_SOURCE CHECK(SOURCE_NODE <= 32767 AND
 SOURCE_NODE >= 0 ),
 CONSTRAINT CA_UPANY_TARGET CHECK(TARGET_NODE <= 32767 AND
 TARGET_NODE >= 0 ),
 CONSTRAINT CA_TARGET_TYPE CHECK(TARGET_TYPE >= 1 AND TARGET_TYPE <=
 5 ),
 CONSTRAINT CA_GROUP_INIT_ROLE CHECK(GROUP_INIT_ROLE IN ('I','M','N')
),
 CONSTRAINT CA_LOAD_TYPE CHECK(LOAD_TYPE >= 0 AND LOAD_TYPE <= 3 )
)
 IN QREPITSO.QAASNAR;


CREATE UNIQUE INDEX ITSO.IX1TARGETS ON ITSO.IBMQREP_TARGETS
```

```
(
 SUBNAME ASC,
 RECVQ ASC
);


CREATE UNIQUE INDEX ITSO.IX2TARGETS ON ITSO.IBMQREP_TARGETS
(
 TARGET_OWNER ASC,
 TARGET_NAME ASC,
 RECVQ ASC,
 SOURCE_OWNER ASC,
 SOURCE_NAME ASC
);


CREATE INDEX ITSO.IX3TARGETS ON ITSO.IBMQREP_TARGETS
(
 RECVQ ASC,
 SUB_ID ASC
);


CREATE TABLE ITSO.IBMQREP_TRG_COLS
(
 RECVQ VARCHAR(48) NOT NULL,
 SUBNAME VARCHAR(132) NOT NULL,
 SOURCE_COLNAME VARCHAR(128) NOT NULL,
 TARGET_COLNAME VARCHAR(128) NOT NULL,
 TARGET_COLNO INTEGER WITH DEFAULT NULL,
 MSG_COL_CODEPAGE INTEGER WITH DEFAULT NULL,
 MSG_COL_NUMBER SMALLINT WITH DEFAULT NULL,
 MSG_COL_TYPE SMALLINT WITH DEFAULT NULL,
 MSG_COL_LENGTH INTEGER WITH DEFAULT NULL,
 IS_KEY CHARACTER(1) NOT NULL,
 CONSTRAINT CA_IS_KEY CHECK(IS_KEY IN ('Y','N'))
)
 IN QREPITSO.QAASNAP;


CREATE UNIQUE INDEX ITSO.IX1TRGCOL ON ITSO.IBMQREP_TRG_COLS
(
 RECVQ ASC,
 SUBNAME ASC,
 TARGET_COLNAME ASC
);


CREATE TABLE ITSO.IBMQREP_SPILLQS
```

```
(
 SPILLQ VARCHAR(48) NOT NULL,
 SUBNAME VARCHAR(132) NOT NULL,
 RECVQ VARCHAR(48) NOT NULL,
 PRIMARY KEY(SPILLQ)
)
 IN QREPITSO.QAASNAP;


CREATE UNIQUE INDEX ITSO.PKIBMQREP_SPILLQS ON ITSO.IBMQREP_SPILLQS
(
 SPILLQ
);



CREATE LOB TABLESPACE LXTIBMQR IN QREPITSO
  LOG NO;


CREATE TABLE ITSO.IBMQREP_EXCEPTIONS
(
 EXCEPTION_TIME TIMESTAMP NOT NULL WITH DEFAULT,
 RECVQ VARCHAR(48) NOT NULL,
 SRC_COMMIT_LSN CHARACTER(10) FOR BIT DATA NOT NULL,
 SRC_TRANS_TIME TIMESTAMP NOT NULL,
 SUBNAME VARCHAR(132) NOT NULL,
 REASON CHARACTER(12) NOT NULL,
 SQLCODE INTEGER,
 SQLSTATE CHARACTER(5),
 SQLERRMC VARCHAR(70) FOR BIT DATA,
 OPERATION VARCHAR(18) NOT NULL,
 TEXT CLOB(32768) NOT NULL,
 IS_APPLIED CHARACTER(1) NOT NULL,
 CONFLICT_RULE CHARACTER(1),
 REPLROWID ROWID NOT NULL GENERATED BY DEFAULT,
 CONSTRAINT CA_IS_APPLIED CHECK(IS_APPLIED IN ('Y','N'))
)
 IN QREPITSO.QAASNAP;



CREATE AUXILIARY TABLE ITSO.XTIBMQREP_EXCEPTIONS0
 IN QREPITSO.LXTIBMQR
 STORES ITSO.IBMQREP_EXCEPTIONS COLUMN TEXT;


CREATE INDEX ITSO.XIXTIBMQREP_EXCEPTIONS0 ON
 ITSO.XTIBMQREP_EXCEPTIONS0;
```

```
CREATE UNIQUE INDEX ITSO.RIIBMQREP_EXCEPTIONS ON
 ITSO.IBMQREP_EXCEPTIONS
(
 REPLROWID
);


CREATE TABLE ITSO.IBMQREP_APPLYTRACE
(
 OPERATION CHARACTER(8) NOT NULL,
 TRACE_TIME TIMESTAMP NOT NULL,
 DESCRIPTION VARCHAR(1024) NOT NULL
)
 IN QREPITSO.QAASNAR;


CREATE INDEX ITSO.IX1TRCTMCOL ON ITSO.IBMQREP_APPLYTRACE
(
 TRACE_TIME ASC
);


CREATE TABLE ITSO.IBMQREP_APPLYMON
(
 MONITOR_TIME TIMESTAMP NOT NULL,
 RECVQ VARCHAR(48) NOT NULL,
 QSTART_TIME TIMESTAMP NOT NULL,
 CURRENT_MEMORY INTEGER NOT NULL,
 QDEPTH INTEGER NOT NULL,
 END2END_LATENCY INTEGER NOT NULL,
 QLATENCY INTEGER NOT NULL,
 APPLY_LATENCY INTEGER NOT NULL,
 TRANS_APPLIED INTEGER NOT NULL,
 ROWS_APPLIED INTEGER NOT NULL,
 TRANS_SERIALIZED INTEGER NOT NULL,
 RI_DEPENDENCIES INTEGER NOT NULL,
 RI_RETRIES INTEGER NOT NULL,
 DEADLOCK_RETRIES INTEGER NOT NULL,
 ROWS_NOT_APPLIED INTEGER NOT NULL,
 MONSTER_TRANS INTEGER NOT NULL,
 MEM_FULL_TIME INTEGER NOT NULL,
 APPLY_SLEEP_TIME INTEGER NOT NULL,
 SPILLED_ROWS INTEGER NOT NULL,
 SPILLEDROWSAPPLIED INTEGER NOT NULL,
 OLDEST_TRANS TIMESTAMP NOT NULL,
 OKSQLSTATE_ERRORS INTEGER NOT NULL,
 HEARTBEAT_LATENCY INTEGER NOT NULL,
```

```
 KEY_DEPENDENCIES INTEGER NOT NULL,
 UNIQ_DEPENDENCIES INTEGER NOT NULL,
 UNIQ_RETRIES INTEGER NOT NULL,
 PRIMARY KEY(MONITOR_TIME, RECVQ)
)
 IN QREPITSO.QAASNAP;


CREATE UNIQUE INDEX ITSO.PKIBMQREP_APPLYMON ON ITSO.IBMQREP_APPLYMON
(
 MONITOR_TIME,
 RECVQ
);


CREATE TABLE ITSO.IBMQREP_DONEMSG
(
 RECVQ VARCHAR(48) NOT NULL,
 MQMSGID CHARACTER(24) FOR BIT DATA NOT NULL,
 PRIMARY KEY(RECVQ, MQMSGID)
)
 IN QREPITSO.QAASNAP;


CREATE UNIQUE INDEX ITSO.PKIBMQREP_DONEMSG ON ITSO.IBMQREP_DONEMSG
(
 RECVQ,
 MQMSGID
);


CREATE TABLE ITSO.IBMQREP_SPILLEDROW
(
 SPILLQ VARCHAR(48) NOT NULL,
 MQMSGID CHARACTER(24) FOR BIT DATA NOT NULL,
 PRIMARY KEY(SPILLQ, MQMSGID)
)
 IN QREPITSO.QAASNAP;


CREATE UNIQUE INDEX ITSO.PKIBMQREP_SPILLEDROW ON
 ITSO.IBMQREP_SPILLEDROW
(
 SPILLQ,
 MQMSGID
);


CREATE TABLE ITSO.IBMQREP_SAVERI
```

```
(
 SUBNAME VARCHAR(132) NOT NULL,
 RECVQ VARCHAR(48) NOT NULL,
 CONSTNAME VARCHAR(18) NOT NULL,
 TABSCHEMA VARCHAR(128) NOT NULL,
 TABNAME VARCHAR(128) NOT NULL,
 REFTABSCHEMA VARCHAR(128) NOT NULL,
 REFTABNAME VARCHAR(128) NOT NULL,
 ALTER_RI_DDL VARCHAR(1680) NOT NULL,
 TYPE_OF_LOAD CHARACTER(1) NOT NULL,
 CONSTRAINT CA_TYPE_OF_LOAD CHECK(TYPE_OF_LOAD IN ('I','E'))
)
 IN QREPITSO.QAASNAP;


CREATE UNIQUE INDEX ITSO.IX1SAVERI ON ITSO.IBMQREP_SAVERI
(
 SUBNAME ASC,
 RECVQ ASC,
 CONSTNAME ASC
);


CREATE TABLE ITSO.IBMQREP_APPLYENQ
(
 LOCKNAME INTEGER
)
 IN QREPITSO.QAASNAR;


INSERT INTO ITSO.IBMQREP_APPLYPARMS
 (qmgr, monitor_limit, trace_limit, monitor_interval, prune_interval,
 autostop, logreuse, logstdout, apply_path, arch_level, term,
 deadlock_retries)
 VALUES
 ('MQS1', 10080, 10080, 300, 300, 'N', 'N', 'N', '//DB2PROD', '0802',
 'Y', 3);


-- COMMIT;
```

Example 3-42   Create the Q Capture and Q Apply control tables on D8G1

```
#######################################################
# The following commands create the control tables
######################################################
ASNCLP SESSION SET TO Q REPLICATION;
```

```
SET OUTPUT MULTIDIR;
#############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 3-1 on page 85, Table 3-2 on page 86 and Table 3-3 on page 88
############################################################
##D8G1 from reference B.7a in Table 3-1 on page 85
##QREPADM from reference B.9 in Table 3-1 on page 85
##QREPITSO from reference B.10 in Table 3-1 on page 85
##MQZ1 from reference B.11 in Table 3-2 on page 86
##Capture schema ITSO from reference B.22 in Table 3-3 on page 88
##QREP.POKA.RESTARTQ from reference B.16 in Table 3-2 on page 86
##//DB2PROD from reference B.24 in Table 3-3 on page 88
##QREP.POKA.ADMINQ from reference B.17 in Table 3-2 on page 86
#############################################################
SET SERVER CAPTURE TO DB D8G1 ID QREPADM PASSWORD "xxxxxx";
SET QMANAGER "MQZ1" FOR CAPTURE SCHEMA;
SET CAPTURE SCHEMA SOURCE ITSO;
SET RUN SCRIPT LATER;
CREATE CONTROL TABLES FOR CAPTURE SERVER USING RESTARTQ "QREP.POKA.RESTARTQ"
ADMINQ "QREP.POKA.ADMINQ" STARTMODE WARMSI CAPTURE PATH "//DB2PROD" IN ZOS PAGE
LOCK DB "QREPITSO" NAMING PREFIX QC CREATE ROW LOCK DB "QREPITSO" NAMING PREFIX
QC CREATE;
############################################################
##D8G1 from reference B.7a in Table 3-1 on page 85
##QREPADM from reference B.9 in Table 3-1 on page 85
##QREPITSO from reference B.10 in Table 3-1 on page 85
##MQZ1 from reference B.11 in Table 3-2 on page 86
##//DB2PROD from reference B.25 in Table 3-3 on page 88
##Apply schema ITSO from reference B.23 in Table 3-3 on page 88
#############################################################
SET SERVER TARGET TO DB D8G1 ID QREPADM PASSWORD "xxxxxx";
SET QMANAGER "MQZ1" FOR APPLY SCHEMA;
SET APPLY SCHEMA ITSO;
SET RUN SCRIPT LATER;
CREATE CONTROL TABLES FOR APPLY SERVER USING APPLY PATH "//DB2PROD"  IN ZOS PAGE
LOCK DB "QREPITSO" NAMING PREFIX QA CREATE ROW LOCK DB "QREPITSO" NAMING PREFIX
QA CREATE;
```

The generated SQL shown in Example 3-43 can be ported to z/OS for execution under SPUFI or DSNTEP2.

Example 3-43 Generated SQL for D8G1 control tables

```
-- DatabaseDB2OS390 (D8G1)

-- CONNECT TO D8G1 USER XXXX using XXXX;
```

```
CREATE TABLESPACE QCITSOCP IN QREPITSO
  SEGSIZE 4
  LOCKSIZE PAGE
  CLOSE NO;



CREATE TABLESPACE QCITSOCR IN QREPITSO
  SEGSIZE 4
  LOCKSIZE ROW
  CLOSE NO;


CREATE TABLE ITSO.IBMQREP_CAPPARMS
(
 QMGR VARCHAR(48) NOT NULL,
 REMOTE_SRC_SERVER VARCHAR(18),
 RESTARTQ VARCHAR(48) NOT NULL,
 ADMINQ VARCHAR(48) NOT NULL,
 STARTMODE VARCHAR(6) NOT NULL WITH DEFAULT 'WARMSI',
 MEMORY_LIMIT INTEGER NOT NULL WITH DEFAULT 32,
 COMMIT_INTERVAL INTEGER NOT NULL WITH DEFAULT 500,
 AUTOSTOP CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 MONITOR_INTERVAL INTEGER NOT NULL WITH DEFAULT 300,
 MONITOR_LIMIT INTEGER NOT NULL WITH DEFAULT 10080,
 TRACE_LIMIT INTEGER NOT NULL WITH DEFAULT 10080,
 SIGNAL_LIMIT INTEGER NOT NULL WITH DEFAULT 10080,
 PRUNE_INTERVAL INTEGER NOT NULL WITH DEFAULT 300,
 SLEEP_INTERVAL INTEGER NOT NULL WITH DEFAULT 5000,
 LOGREUSE CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 LOGSTDOUT CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 TERM CHARACTER(1) NOT NULL WITH DEFAULT 'Y',
 CAPTURE_PATH VARCHAR(1040) WITH DEFAULT NULL,
 ARCH_LEVEL CHARACTER(4) NOT NULL WITH DEFAULT '0802',
 CONSTRAINT CC_STARTMODE CHECK(STARTMODE IN ('COLD','WARMSI','WARMSA'
,'WARMNS')),
 CONSTRAINT CC_MEMORY_LIMIT CHECK(MEMORY_LIMIT >= 1 AND MEMORY_LIMIT
 <= 1000),
 CONSTRAINT CC_COMMIT_INTERVAL CHECK(COMMIT_INTERVAL  >= 100 AND
 COMMIT_INTERVAL <= 600000),
 CONSTRAINT CC_AUTOSTOP CHECK(AUTOSTOP IN ('Y','N')),
 CONSTRAINT CC_MON_INTERVAL CHECK(MONITOR_INTERVAL >= 1 AND
 MONITOR_INTERVAL <= 2147483647),
 CONSTRAINT CC_MON_LIMIT CHECK(MONITOR_LIMIT >= 1 AND MONITOR_LIMIT
 <= 2147483647),
 CONSTRAINT CC_TRACE_LIMT CHECK(TRACE_LIMIT >= 1 AND TRACE_LIMIT <=
 2147483647),
 CONSTRAINT CC_SIGNAL_LIMIT CHECK(SIGNAL_LIMIT >= 1 AND SIGNAL_LIMIT
```

```
 <= 2147483647),
 CONSTRAINT CC_PRUNE_INTERVAL CHECK(PRUNE_INTERVAL >= 1 AND
 PRUNE_INTERVAL <= 2147483647),
 CONSTRAINT CC_LOGREUSE CHECK(LOGREUSE IN ('Y','N')),
 CONSTRAINT CC_LOGSTDOUT CHECK(LOGSTDOUT IN ('Y','N')),
 CONSTRAINT CC_TERM CHECK(TERM IN ('Y','N')),
 CONSTRAINT CC_SLEEP_INTERVAL CHECK(SLEEP_INTERVAL >= 1 AND
 SLEEP_INTERVAL <= 2147483647)
)
 IN QREPITSO.QCITSOCP;


CREATE UNIQUE INDEX ITSO.IX1CQMGRCOL ON ITSO.IBMQREP_CAPPARMS
(
 QMGR ASC
);


CREATE TABLE ITSO.IBMQREP_SENDQUEUES
(
 PUBQMAPNAME VARCHAR(128) NOT NULL,
 SENDQ VARCHAR(48) NOT NULL,
 RECVQ VARCHAR(48),
 MESSAGE_FORMAT CHARACTER(1) NOT NULL WITH DEFAULT 'C',
 MSG_CONTENT_TYPE CHARACTER(1) NOT NULL WITH DEFAULT 'T',
 STATE CHARACTER(1) NOT NULL WITH DEFAULT 'A',
 STATE_TIME TIMESTAMP NOT NULL WITH DEFAULT,
 STATE_INFO CHARACTER(8),
 ERROR_ACTION CHARACTER(1) NOT NULL WITH DEFAULT 'S',
 HEARTBEAT_INTERVAL INTEGER NOT NULL WITH DEFAULT 60,
 MAX_MESSAGE_SIZE INTEGER NOT NULL WITH DEFAULT 64,
 APPLY_SERVER VARCHAR(18),
 APPLY_ALIAS VARCHAR(8),
 APPLY_SCHEMA VARCHAR(128),
 DESCRIPTION VARCHAR(254),
 PRIMARY KEY(SENDQ),
 CONSTRAINT CC_MSG_FORMAT CHECK(MESSAGE_FORMAT IN ('X','C', 'J')),
 CONSTRAINT CC_MSG_CONT_TYPE CHECK(MSG_CONTENT_TYPE IN ('T','R')),
 CONSTRAINT CC_SENDQ_STATE CHECK(STATE IN ('A','I')),
 CONSTRAINT CC_QERRORACTION CHECK(ERROR_ACTION IN ('I','S')),
 CONSTRAINT CC_HTBEAT_INTERVAL CHECK(HEARTBEAT_INTERVAL >= 0 AND
 HEARTBEAT_INTERVAL <= 32767)
)
 IN QREPITSO.QCITSOCR;


CREATE UNIQUE INDEX ITSO.PKIBMQREP_SENDQUEUES ON
 ITSO.IBMQREP_SENDQUEUES
(
```

```
 SENDQ
);


CREATE UNIQUE INDEX ITSO.IX1PUBMAPCOL ON ITSO.IBMQREP_SENDQUEUES
(
 PUBQMAPNAME ASC
);


CREATE TABLE ITSO.IBMQREP_SUBS
(
 SUBNAME VARCHAR(132) NOT NULL,
 SOURCE_OWNER VARCHAR(128) NOT NULL,
 SOURCE_NAME VARCHAR(128) NOT NULL,
 TARGET_SERVER VARCHAR(18),
 TARGET_ALIAS VARCHAR(8),
 TARGET_OWNER VARCHAR(128),
 TARGET_NAME VARCHAR(128),
 TARGET_TYPE INTEGER,
 APPLY_SCHEMA VARCHAR(128),
 SENDQ VARCHAR(48) NOT NULL,
 SEARCH_CONDITION VARCHAR(2048) WITH DEFAULT NULL,
 SUB_ID INTEGER WITH DEFAULT NULL,
 SUBTYPE CHARACTER(1) NOT NULL WITH DEFAULT 'U',
 ALL_CHANGED_ROWS CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 BEFORE_VALUES CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 CHANGED_COLS_ONLY CHARACTER(1) NOT NULL WITH DEFAULT 'Y',
 HAS_LOADPHASE CHARACTER(1) NOT NULL WITH DEFAULT 'I',
 STATE CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 STATE_TIME TIMESTAMP NOT NULL WITH DEFAULT,
 STATE_INFO CHARACTER(8),
 STATE_TRANSITION VARCHAR(256) FOR BIT DATA,
 SUBGROUP VARCHAR(30) WITH DEFAULT NULL,
 SOURCE_NODE SMALLINT NOT NULL WITH DEFAULT 0,
 TARGET_NODE SMALLINT NOT NULL WITH DEFAULT 0,
 GROUP_MEMBERS CHARACTER(254) FOR BIT DATA WITH DEFAULT NULL,
 OPTIONS_FLAG CHARACTER(4) NOT NULL WITH DEFAULT 'NNNN',
 SUPPRESS_DELETES CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 DESCRIPTION VARCHAR(200),
 TOPIC VARCHAR(256),
 PRIMARY KEY(SUBNAME),
 CONSTRAINT FKSENDQ FOREIGN KEY(SENDQ) REFERENCES
 ITSO.IBMQREP_SENDQUEUES(SENDQ),
 CONSTRAINT CC_SUBTYPE CHECK(SUBTYPE IN ('U','B','P')),
 CONSTRAINT CC_ALL_CHGD_ROWS CHECK(ALL_CHANGED_ROWS IN ('Y','N')),
 CONSTRAINT CC_BEFORE_VALUES CHECK(BEFORE_VALUES IN ('Y','N')),
 CONSTRAINT CC_CHGD_COLS_ONLY CHECK(CHANGED_COLS_ONLY IN ('Y','N')),
 CONSTRAINT CC_HAS_LOADPHASE CHECK(HAS_LOADPHASE IN ('N','I','E')),
```

```
 CONSTRAINT CC_SUBS_STATE CHECK(STATE IN ('L','A','I','T','G','N')),
 CONSTRAINT CC_SUPPRESS_DELS CHECK(SUPPRESS_DELETES IN ('Y','N'))
)
 IN QREPITSO.QCITSOCR;


CREATE UNIQUE INDEX ITSO.PKIBMQREP_SUBS ON ITSO.IBMQREP_SUBS
(
 SUBNAME
);


CREATE TABLE ITSO.IBMQREP_SRC_COLS
(
 SUBNAME VARCHAR(132) NOT NULL,
 SRC_COLNAME VARCHAR(128) NOT NULL,
 IS_KEY SMALLINT NOT NULL WITH DEFAULT 0,
 PRIMARY KEY(SUBNAME, SRC_COLNAME),
 CONSTRAINT FKSUBS FOREIGN KEY(SUBNAME) REFERENCES ITSO.IBMQREP_SUBS
(SUBNAME)
)
 IN QREPITSO.QCITSOCP;


CREATE UNIQUE INDEX ITSO.PKIBMQREP_SRC_COLS ON ITSO.IBMQREP_SRC_COLS
(
 SUBNAME,
 SRC_COLNAME
);


CREATE TABLE ITSO.IBMQREP_SRCH_COND
(
 ASNQREQD INTEGER
)
 IN QREPITSO.QCITSOCP;


CREATE TABLE ITSO.IBMQREP_SIGNAL
(
 SIGNAL_TIME TIMESTAMP NOT NULL WITH DEFAULT,
 SIGNAL_TYPE VARCHAR(30) NOT NULL,
 SIGNAL_SUBTYPE VARCHAR(30),
 SIGNAL_INPUT_IN VARCHAR(500),
 SIGNAL_STATE CHARACTER(1) NOT NULL WITH DEFAULT 'P',
 SIGNAL_LSN CHARACTER(10) FOR BIT DATA,
 PRIMARY KEY(SIGNAL_TIME),
 CONSTRAINT CC_SIGNAL_TYPE CHECK(SIGNAL_TYPE IN ('CMD','USER')),
 CONSTRAINT CC_SIGNAL_STATE CHECK(SIGNAL_STATE IN ('P','R','C','F'))
```

```
)
 IN QREPITSO.QCITSOCR
 DATA CAPTURE CHANGES;


CREATE UNIQUE INDEX ITSO.PKIBMQREP_SIGNAL ON ITSO.IBMQREP_SIGNAL
(
 SIGNAL_TIME
);


CREATE TABLE ITSO.IBMQREP_CAPTRACE
(
 OPERATION CHARACTER(8) NOT NULL,
 TRACE_TIME TIMESTAMP NOT NULL,
 DESCRIPTION VARCHAR(1024) NOT NULL
)
 IN QREPITSO.QCITSOCR;


CREATE TABLE ITSO.IBMQREP_CAPMON
(
 MONITOR_TIME TIMESTAMP NOT NULL,
 CURRENT_LOG_TIME TIMESTAMP NOT NULL,
 CAPTURE_IDLE INTEGER NOT NULL,
 CURRENT_MEMORY INTEGER NOT NULL,
 ROWS_PROCESSED INTEGER NOT NULL,
 TRANS_SKIPPED INTEGER NOT NULL,
 TRANS_PROCESSED INTEGER NOT NULL,
 TRANS_SPILLED INTEGER NOT NULL,
 MAX_TRANS_SIZE INTEGER NOT NULL,
 QUEUES_IN_ERROR INTEGER NOT NULL,
 RESTART_SEQ CHARACTER(10) FOR BIT DATA NOT NULL,
 CURRENT_SEQ CHARACTER(10) FOR BIT DATA NOT NULL,
 PRIMARY KEY(MONITOR_TIME)
)
 IN QREPITSO.QCITSOCP;


CREATE UNIQUE INDEX ITSO.PKIBMQREP_CAPMON ON ITSO.IBMQREP_CAPMON
(
 MONITOR_TIME
);


CREATE TABLE ITSO.IBMQREP_CAPQMON
(
 MONITOR_TIME TIMESTAMP NOT NULL,
 SENDQ VARCHAR(48) NOT NULL,
```

```
 ROWS_PUBLISHED INTEGER NOT NULL,
 TRANS_PUBLISHED INTEGER NOT NULL,
 CHG_ROWS_SKIPPED INTEGER NOT NULL,
 DELROWS_SUPPRESSED INTEGER NOT NULL,
 ROWS_SKIPPED INTEGER NOT NULL,
 PRIMARY KEY(MONITOR_TIME, SENDQ)
)
 IN QREPITSO.QCITSOCP;


CREATE UNIQUE INDEX ITSO.PKIBMQREP_CAPQMON ON ITSO.IBMQREP_CAPQMON
(
 MONITOR_TIME,
 SENDQ
);


CREATE TABLE ITSO.IBMQREP_CAPENQ
(
 LOCKNAME INTEGER
)
 IN QREPITSO.QCITSOCR;


CREATE TABLE ITSO.IBMQREP_ADMINMSG
(
 MQMSGID CHARACTER(24) FOR BIT DATA NOT NULL,
 MSG_TIME TIMESTAMP NOT NULL WITH DEFAULT,
 PRIMARY KEY(MQMSGID)
)
 IN QREPITSO.QCITSOCP;


CREATE UNIQUE INDEX ITSO.PKIBMQREP_ADMINMSG ON ITSO.IBMQREP_ADMINMSG
(
 MQMSGID
);


INSERT INTO ITSO.IBMQREP_CAPPARMS
 (qmgr, restartq, adminq, startmode, memory_limit, commit_interval,
 autostop, monitor_interval, monitor_limit, trace_limit, signal_limit,
 prune_interval, sleep_interval, logreuse, logstdout, term,
 capture_path, arch_level)
 VALUES
 ('MQZ1', 'QREP.POKA.RESTARTQ', 'QREP.POKA.ADMINQ', 'WARMSI', 32, 500
, 'N', 300, 10080, 10080, 10080, 300, 5000, 'N', 'N', 'Y', '//DKELSEY'
, '0802');
```

```
-- COMMIT;


-- DatabaseDB2OS390 (D8G1)

-- CONNECT TO D8G1 USER XXXX using XXXX;



CREATE TABLESPACE QAASNAP IN QREPITSO
  SEGSIZE 4
  LOCKSIZE PAGE
  CLOSE NO;



CREATE TABLESPACE QAASNAR IN QREPITSO
  SEGSIZE 4
  LOCKSIZE ROW
  CLOSE NO;


CREATE TABLE ITSO.IBMQREP_APPLYPARMS
(
 QMGR VARCHAR(48) NOT NULL,
 MONITOR_LIMIT INTEGER NOT NULL WITH DEFAULT 10080,
 TRACE_LIMIT INTEGER NOT NULL WITH DEFAULT 10080,
 MONITOR_INTERVAL INTEGER NOT NULL WITH DEFAULT 300,
 PRUNE_INTERVAL INTEGER NOT NULL WITH DEFAULT 300,
 AUTOSTOP CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 LOGREUSE CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 LOGSTDOUT CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 APPLY_PATH VARCHAR(1040) WITH DEFAULT NULL,
 ARCH_LEVEL CHARACTER(4) NOT NULL WITH DEFAULT '0802',
 TERM CHARACTER(1) NOT NULL WITH DEFAULT 'Y',
 PWDFILE VARCHAR(48) WITH DEFAULT NULL,
 DEADLOCK_RETRIES INTEGER NOT NULL WITH DEFAULT 3,
 CONSTRAINT CA_MON_LIMIT CHECK(MONITOR_LIMIT >= 1 AND MONITOR_LIMIT
 <= 2147483647),
 CONSTRAINT CA_TRACE_LIMT CHECK(TRACE_LIMIT >= 1 AND TRACE_LIMIT <=
 2147483647),
 CONSTRAINT CA_MON_INTERVAL CHECK(MONITOR_INTERVAL >= 1 AND
 MONITOR_INTERVAL <= 2147483647),
 CONSTRAINT CA_PRUNE_INTERVAL CHECK(PRUNE_INTERVAL >= 1 AND
 PRUNE_INTERVAL <= 2147483647),
 CONSTRAINT CA_AUTOSTOP CHECK(AUTOSTOP IN ('Y','N')),
 CONSTRAINT CA_LOGREUSE CHECK(LOGREUSE IN ('Y','N')),
 CONSTRAINT CA_LOGSTDOUT CHECK(LOGSTDOUT IN ('Y','N')),
```

```
 CONSTRAINT CA_TERM CHECK(TERM IN ('Y','N')),
 CONSTRAINT CA_RETRIES CHECK(DEADLOCK_RETRIES  >= 3 AND
 DEADLOCK_RETRIES <= 2147483647)
)
 IN QREPITSO.QAASNAP;


CREATE UNIQUE INDEX ITSO.IX1AQMGRCOL ON ITSO.IBMQREP_APPLYPARMS
(
 QMGR ASC
);


CREATE TABLE ITSO.IBMQREP_RECVQUEUES
(
 REPQMAPNAME VARCHAR(128) NOT NULL,
 RECVQ VARCHAR(48) NOT NULL,
 SENDQ VARCHAR(48) WITH DEFAULT NULL,
 ADMINQ VARCHAR(48) NOT NULL,
 NUM_APPLY_AGENTS INTEGER NOT NULL WITH DEFAULT 16,
 MEMORY_LIMIT INTEGER NOT NULL WITH DEFAULT 32,
 CAPTURE_SERVER VARCHAR(18) NOT NULL,
 CAPTURE_ALIAS VARCHAR(8) NOT NULL,
 CAPTURE_SCHEMA VARCHAR(30) NOT NULL WITH DEFAULT 'ASN',
 STATE CHARACTER(1) NOT NULL WITH DEFAULT 'A',
 STATE_TIME TIMESTAMP NOT NULL WITH DEFAULT,
 STATE_INFO CHARACTER(8),
 DESCRIPTION VARCHAR(254),
 PRIMARY KEY(RECVQ),
 CONSTRAINT CC_SENDQ_STATE CHECK(STATE IN ('A','I'))
)
 IN QREPITSO.QAASNAR;


CREATE UNIQUE INDEX ITSO.PKIBMQREP_RECVQUEUES ON
 ITSO.IBMQREP_RECVQUEUES
(
 RECVQ
);


CREATE UNIQUE INDEX ITSO.IX1REPMAPCOL ON ITSO.IBMQREP_RECVQUEUES
(
 REPQMAPNAME ASC
);


CREATE TABLE ITSO.IBMQREP_TARGETS
(
```

```
 SUBNAME VARCHAR(132) NOT NULL,
 RECVQ VARCHAR(48) NOT NULL,
 SUB_ID INTEGER WITH DEFAULT NULL,
 SOURCE_SERVER VARCHAR(18) NOT NULL,
 SOURCE_ALIAS VARCHAR(8) NOT NULL,
 SOURCE_OWNER VARCHAR(128) NOT NULL,
 SOURCE_NAME VARCHAR(128) NOT NULL,
 SRC_NICKNAME_OWNER VARCHAR(128),
 SRC_NICKNAME VARCHAR(128),
 TARGET_OWNER VARCHAR(128) NOT NULL,
 TARGET_NAME VARCHAR(128) NOT NULL,
 TARGET_TYPE INTEGER NOT NULL WITH DEFAULT 1,
 FEDERATED_TGT_SRVR VARCHAR(18) WITH DEFAULT NULL,
 STATE CHARACTER(1) NOT NULL WITH DEFAULT 'I',
 STATE_TIME TIMESTAMP NOT NULL WITH DEFAULT,
 STATE_INFO CHARACTER(8),
 SUBTYPE CHARACTER(1) NOT NULL WITH DEFAULT 'U',
 CONFLICT_RULE CHARACTER(1) NOT NULL WITH DEFAULT 'K',
 CONFLICT_ACTION CHARACTER(1) NOT NULL WITH DEFAULT 'I',
 ERROR_ACTION CHARACTER(1) NOT NULL WITH DEFAULT 'Q',
 SPILLQ VARCHAR(48) WITH DEFAULT NULL,
 OKSQLSTATES VARCHAR(128) WITH DEFAULT NULL,
 SUBGROUP VARCHAR(30) WITH DEFAULT NULL,
 SOURCE_NODE SMALLINT NOT NULL WITH DEFAULT 0,
 TARGET_NODE SMALLINT NOT NULL WITH DEFAULT 0,
 GROUP_INIT_ROLE CHARACTER(1) WITH DEFAULT NULL,
 HAS_LOADPHASE CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 LOAD_TYPE SMALLINT NOT NULL WITH DEFAULT 0,
 DESCRIPTION VARCHAR(254),
 SEARCH_CONDITION VARCHAR(2048) WITH DEFAULT NULL,
 CONSTRAINT CA_TARGTBL_STATE CHECK(STATE IN ('L','A','I','E','D','F'
,'T')),
 CONSTRAINT CA_UPDATEANY CHECK(SUBTYPE IN ('U','B','P')),
 CONSTRAINT CA_CONFLICTACTION CHECK(CONFLICT_ACTION IN ('F','I','D'
,'S','Q')),
 CONSTRAINT CA_ERRORACTION CHECK(ERROR_ACTION IN ('D','S','Q')),
 CONSTRAINT CA_UPANY_SOURCE CHECK(SOURCE_NODE <= 32767 AND
 SOURCE_NODE >= 0 ),
 CONSTRAINT CA_UPANY_TARGET CHECK(TARGET_NODE <= 32767 AND
 TARGET_NODE >= 0 ),
 CONSTRAINT CA_TARGET_TYPE CHECK(TARGET_TYPE >= 1 AND TARGET_TYPE <=
 5 ),
 CONSTRAINT CA_GROUP_INIT_ROLE CHECK(GROUP_INIT_ROLE IN ('I','M','N')
),
 CONSTRAINT CA_LOAD_TYPE CHECK(LOAD_TYPE >= 0 AND LOAD_TYPE <= 3 )
)
 IN QREPITSO.QAASNAR;
```

```
CREATE UNIQUE INDEX ITSO.IX1TARGETS ON ITSO.IBMQREP_TARGETS
(
 SUBNAME ASC,
 RECVQ ASC
);


CREATE UNIQUE INDEX ITSO.IX2TARGETS ON ITSO.IBMQREP_TARGETS
(
 TARGET_OWNER ASC,
 TARGET_NAME ASC,
 RECVQ ASC,
 SOURCE_OWNER ASC,
 SOURCE_NAME ASC
);


CREATE INDEX ITSO.IX3TARGETS ON ITSO.IBMQREP_TARGETS
(
 RECVQ ASC,
 SUB_ID ASC
);


CREATE TABLE ITSO.IBMQREP_TRG_COLS
(
 RECVQ VARCHAR(48) NOT NULL,
 SUBNAME VARCHAR(132) NOT NULL,
 SOURCE_COLNAME VARCHAR(128) NOT NULL,
 TARGET_COLNAME VARCHAR(128) NOT NULL,
 TARGET_COLNO INTEGER WITH DEFAULT NULL,
 MSG_COL_CODEPAGE INTEGER WITH DEFAULT NULL,
 MSG_COL_NUMBER SMALLINT WITH DEFAULT NULL,
 MSG_COL_TYPE SMALLINT WITH DEFAULT NULL,
 MSG_COL_LENGTH INTEGER WITH DEFAULT NULL,
 IS_KEY CHARACTER(1) NOT NULL,
 CONSTRAINT CA_IS_KEY CHECK(IS_KEY IN ('Y','N'))
)
 IN QREPITSO.QAASNAP;


CREATE UNIQUE INDEX ITSO.IX1TRGCOL ON ITSO.IBMQREP_TRG_COLS
(
 RECVQ ASC,
 SUBNAME ASC,
 TARGET_COLNAME ASC
);
```

```
CREATE TABLE ITSO.IBMQREP_SPILLQS
(
 SPILLQ VARCHAR(48) NOT NULL,
 SUBNAME VARCHAR(132) NOT NULL,
 RECVQ VARCHAR(48) NOT NULL,
 PRIMARY KEY(SPILLQ)
)
 IN QREPITSO.QAASNAP;


CREATE UNIQUE INDEX ITSO.PKIBMQREP_SPILLQS ON ITSO.IBMQREP_SPILLQS
(
 SPILLQ
);



CREATE LOB TABLESPACE LXTIBMQR IN QREPITSO
  LOG NO;


CREATE TABLE ITSO.IBMQREP_EXCEPTIONS
(
 EXCEPTION_TIME TIMESTAMP NOT NULL WITH DEFAULT,
 RECVQ VARCHAR(48) NOT NULL,
 SRC_COMMIT_LSN CHARACTER(10) FOR BIT DATA NOT NULL,
 SRC_TRANS_TIME TIMESTAMP NOT NULL,
 SUBNAME VARCHAR(132) NOT NULL,
 REASON CHARACTER(12) NOT NULL,
 SQLCODE INTEGER,
 SQLSTATE CHARACTER(5),
 SQLERRMC VARCHAR(70) FOR BIT DATA,
 OPERATION VARCHAR(18) NOT NULL,
 TEXT CLOB(32768) NOT NULL,
 IS_APPLIED CHARACTER(1) NOT NULL,
 CONFLICT_RULE CHARACTER(1),
 REPLROWID ROWID NOT NULL GENERATED BY DEFAULT,
 CONSTRAINT CA_IS_APPLIED CHECK(IS_APPLIED IN ('Y','N'))
)
 IN QREPITSO.QAASNAP;



CREATE AUXILIARY TABLE ITSO.XTIBMQREP_EXCEPTIONS0
 IN QREPITSO.LXTIBMQR
 STORES ITSO.IBMQREP_EXCEPTIONS COLUMN TEXT;


CREATE INDEX ITSO.XIXTIBMQREP_EXCEPTIONS0 ON
```

```
 ITSO.XTIBMQREP_EXCEPTIONS0;


CREATE UNIQUE INDEX ITSO.RIIBMQREP_EXCEPTIONS ON
 ITSO.IBMQREP_EXCEPTIONS
(
 REPLROWID
);


CREATE TABLE ITSO.IBMQREP_APPLYTRACE
(
 OPERATION CHARACTER(8) NOT NULL,
 TRACE_TIME TIMESTAMP NOT NULL,
 DESCRIPTION VARCHAR(1024) NOT NULL
)
 IN QREPITSO.QAASNAR;


CREATE INDEX ITSO.IX1TRCTMCOL ON ITSO.IBMQREP_APPLYTRACE
(
 TRACE_TIME ASC
);


CREATE TABLE ITSO.IBMQREP_APPLYMON
(
 MONITOR_TIME TIMESTAMP NOT NULL,
 RECVQ VARCHAR(48) NOT NULL,
 QSTART_TIME TIMESTAMP NOT NULL,
 CURRENT_MEMORY INTEGER NOT NULL,
 QDEPTH INTEGER NOT NULL,
 END2END_LATENCY INTEGER NOT NULL,
 QLATENCY INTEGER NOT NULL,
 APPLY_LATENCY INTEGER NOT NULL,
 TRANS_APPLIED INTEGER NOT NULL,
 ROWS_APPLIED INTEGER NOT NULL,
 TRANS_SERIALIZED INTEGER NOT NULL,
 RI_DEPENDENCIES INTEGER NOT NULL,
 RI_RETRIES INTEGER NOT NULL,
 DEADLOCK_RETRIES INTEGER NOT NULL,
 ROWS_NOT_APPLIED INTEGER NOT NULL,
 MONSTER_TRANS INTEGER NOT NULL,
 MEM_FULL_TIME INTEGER NOT NULL,
 APPLY_SLEEP_TIME INTEGER NOT NULL,
 SPILLED_ROWS INTEGER NOT NULL,
 SPILLEDROWSAPPLIED INTEGER NOT NULL,
 OLDEST_TRANS TIMESTAMP NOT NULL,
 OKSQLSTATE_ERRORS INTEGER NOT NULL,
```

```
 HEARTBEAT_LATENCY INTEGER NOT NULL,
 KEY_DEPENDENCIES INTEGER NOT NULL,
 UNIQ_DEPENDENCIES INTEGER NOT NULL,
 UNIQ_RETRIES INTEGER NOT NULL,
 PRIMARY KEY(MONITOR_TIME, RECVQ)
)
 IN QREPITSO.QAASNAP;


CREATE UNIQUE INDEX ITSO.PKIBMQREP_APPLYMON ON ITSO.IBMQREP_APPLYMON
(
 MONITOR_TIME,
 RECVQ
);


CREATE TABLE ITSO.IBMQREP_DONEMSG
(
 RECVQ VARCHAR(48) NOT NULL,
 MQMSGID CHARACTER(24) FOR BIT DATA NOT NULL,
 PRIMARY KEY(RECVQ, MQMSGID)
)
 IN QREPITSO.QAASNAP;


CREATE UNIQUE INDEX ITSO.PKIBMQREP_DONEMSG ON ITSO.IBMQREP_DONEMSG
(
 RECVQ,
 MQMSGID
);


CREATE TABLE ITSO.IBMQREP_SPILLEDROW
(
 SPILLQ VARCHAR(48) NOT NULL,
 MQMSGID CHARACTER(24) FOR BIT DATA NOT NULL,
 PRIMARY KEY(SPILLQ, MQMSGID)
)
 IN QREPITSO.QAASNAP;


CREATE UNIQUE INDEX ITSO.PKIBMQREP_SPILLEDROW ON
 ITSO.IBMQREP_SPILLEDROW
(
 SPILLQ,
 MQMSGID
);
```

```
CREATE TABLE ITSO.IBMQREP_SAVERI
(
 SUBNAME VARCHAR(132) NOT NULL,
 RECVQ VARCHAR(48) NOT NULL,
 CONSTNAME VARCHAR(18) NOT NULL,
 TABSCHEMA VARCHAR(128) NOT NULL,
 TABNAME VARCHAR(128) NOT NULL,
 REFTABSCHEMA VARCHAR(128) NOT NULL,
 REFTABNAME VARCHAR(128) NOT NULL,
 ALTER_RI_DDL VARCHAR(1680) NOT NULL,
 TYPE_OF_LOAD CHARACTER(1) NOT NULL,
 CONSTRAINT CA_TYPE_OF_LOAD CHECK(TYPE_OF_LOAD IN ('I','E'))
)
 IN QREPITSO.QAASNAP;


CREATE UNIQUE INDEX ITSO.IX1SAVERI ON ITSO.IBMQREP_SAVERI
(
 SUBNAME ASC,
 RECVQ ASC,
 CONSTNAME ASC
);


CREATE TABLE ITSO.IBMQREP_APPLYENQ
(
 LOCKNAME INTEGER
)
 IN QREPITSO.QAASNAR;


INSERT INTO ITSO.IBMQREP_APPLYPARMS
 (qmgr, monitor_limit, trace_limit, monitor_interval, prune_interval,
 autostop, logreuse, logstdout, apply_path, arch_level, term,
 deadlock_retries)
 VALUES
 ('MQZ1', 10080, 10080, 300, 300, 'N', 'N', 'N', '//DKELSEY', '0802',
 'Y', 3);


-- COMMIT;
```

Attention: Both the generated scripts *must be executed before* the next step of creating the replication queue maps.

Create the replication queue maps

Example 3-44 lists the ASNCLP commands for creating the replication queue maps on DT11 (STLABD1), while Example 3-47 does the same on D8G1 (WTSC53). These commands should be saved in a file that is then executed using the command described in Example 3-39 on page 194.

Example 3-44 Create the replication queue maps for replication from DT11 to D8G1

```
##############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 3-1 on page 85, Table 3-2 on page 86 and Table 3-3 on page 88
#############################################################
##DT11 from reference A.7a in Table 3-1 on page 85
##QREPADM from reference A.9 in Table 3-1 on page 85
##D8G1 from reference B.7a in Table 3-1 on page 85
##QREPADM from reference B.9 in Table 3-1 on page 85
##Capture schema ITSO from reference A.22 in Table 3-3 on page 88
##Apply schema ITSO from reference B.23 in Table 3-3 on page 88
##QMAP_STL_TO_POK from reference A.27 in Table 3-3 on page 88
##QREP.STLA.ADMINQ from reference A.17 in Table 3-2 on page 86
##RECVQ QREP.STLA.TO.POKA.SENDQ from reference B.19 in Table 3-2 on page 86
##SENDQ QREP.STLA.TO.POKA.SENDQ from reference A.18 in Table 3-2 on page 86
##Number of apply agents 16 from reference A.27.c in Table 3-3 on page 96
##Memory Limit 2 from reference A.27.d in Table 3-3 on page 96
##Error Action 'S' from reference A.27.b in Table 3-3 on page 96
##MSGSIZE 64 from reference A.27.a in Table 3-3 on page 96
##############################################################

ASNCLP SESSION SET TO Q REPLICATION;
SET OUTPUT MULTIDIR;
SET SERVER CAPTURE TO DB DT11 ID "QREPADM" PASSWORD "xxxxxx";
SET SERVER TARGET  TO DB D8G1 ID "QREPADM" PASSWORD "xxxxxx";

SET CAPTURE SCHEMA SOURCE ITSO;
SET APPLY   SCHEMA ITSO;
SET RUN SCRIPT LATER;
CREATE REPLQMAP QMAP_STL_TO_POK USING ADMINQ "QREP.STLA.ADMINQ" RECVQ
"QREP.STLA.TO.POKA.RECVQ" SENDQ "QREP.STLA.TO.POKA.SENDQ" NUM APPLY AGENTS 16
MEMORY LIMIT 2 ERROR ACTION S MAX MESSAGE SIZE 64;
```

Two SQL members are generated (one each for the SEND and RECV replication queue map), as shown in Example 3-45 and Example 3-46, which can be ported to z/OS for execution under SPUFI or DSNTEP2.

Example 3-45 Generated SQL for replication queue map SENDQ DT11 to D8G1

```
-- DatabaseDB2OS390 (DT11)
```

```
-- CONNECT TO DT11 USER XXXX using XXXX;


INSERT INTO ITSO.IBMQREP_SENDQUEUES
 (pubqmapname, sendq, message_format, msg_content_type, state,
 error_action, heartbeat_interval, max_message_size, apply_alias,
 apply_schema, recvq, apply_server)
 VALUES
 ('QMAP_STL_TO_POK', 'QREP.STLA.TO.POKA.SENDQ', 'C', 'T', 'A', 'I',
 60, 2048, 'D8G1', 'ITSO', 'QREP.STLA.TO.POKA.RECVQ', 'DB8G');

-- COMMIT;
```

Example 3-46 Generated SQL for replication queue map RECVQ DT11 to D8G1

```
-- DatabaseDB2OS390 (D8G1)

-- CONNECT TO D8G1 USER XXXX using XXXX;


INSERT INTO ITSO.IBMQREP_RECVQUEUES
 (repqmapname, recvq, sendq, adminq, capture_alias, capture_schema,
 num_apply_agents, memory_limit, state, capture_server)
 VALUES
 ('QMAP_STL_TO_POK', 'QREP.STLA.TO.POKA.RECVQ',
 'QREP.STLA.TO.POKA.SENDQ', 'QREP.STLA.ADMINQ', 'DT11', 'ITSO', 16, 30
, 'A', 'DSNT1');


-- COMMIT;
```

Example 3-47 Create the replication queue maps for replication from D8G1 to DT11

```
############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 3-1 on page 85, Table 3-2 on page 86 and Table 3-3 on page 88
############################################################
##DT11 from reference A.7a in Table 3-1 on page 85
##QREPADM from reference A.9 in Table 3-1 on page 85
##D8G1 from reference B.7a in Table 3-1 on page 85
##QREPADM from reference B.9 in Table 3-1 on page 85
##Capture schema ITSO from reference B.22 in Table 3-3 on page 88
##Apply schema ITSO from reference A.23 in Table 3-3 on page 88
##QMAP_POK_TO_STL from reference B.27 in Table 3-3 on page 88
##QREP.POKA.ADMINQ from reference B.17 in Table 3-2 on page 86
##RECVQ QREP.POKA.TO.STLA.RECVQ from reference A.19 in Table 3-2 on page 86
```

```
##SENDQ QREP.POKA.TO.STLA.RECVQ from reference B.18 in Table 3-2 on page 86
##Number of apply agents 16 from reference B.27.c in Table 3-3 on page 96
##Memory Limit 2 from reference B.27.d in Table 3-3 on page 96
##Error Action 'S' from reference B.27.b in Table 3-3 on page 96
##MSGSIZE 64 from reference B.27.a in Table 3-3 on page 96
#############################################################
ASNCLP SESSION SET TO Q REPLICATION;
SET OUTPUT MULTIDIR;

SET SERVER CAPTURE TO DB D8G1 ID "QREPADM" PASSWORD "xxxxxx";
SET SERVER TARGET  TO DB DT11 ID "QREPADM" PASSWORD "xxxxxx";

SET CAPTURE SCHEMA SOURCE ITSO;
SET APPLY   SCHEMA ITSO;
SET RUN SCRIPT LATER;
CREATE REPLQMAP QMAP_POK_TO_STL USING ADMINQ "QREP.POKA.ADMINQ" RECVQ
"QREP.POKA.TO.STLA.RECVQ" SENDQ "QREP.POKA.TO.STLA.RECVQ" NUM APPLY AGENTS 16
MEMORY LIMIT 2 ERROR ACTION S MAX MESSAGE SIZE 64;
```

Two SQL members are generated (one each for the SEND and RECV replication queue map), as shown in Example 3-48 and Example 3-49, which can be ported to z/OS for execution under SPUFI or DSNTEP2.

Example 3-48 Generated SQL for replication queue map SENDQ D8G1 to DT11

```
-- DatabaseDB2OS390 (D8G1)

-- CONNECT TO D8G1 USER XXXX using XXXX;


INSERT INTO ITSO.IBMQREP_SENDQUEUES
 (pubqmapname, sendq, message_format, msg_content_type, state,
 error_action, heartbeat_interval, max_message_size, apply_alias,
 apply_schema, recvq, apply_server)
 VALUES
 ('QMAP_POK_TO_STL', 'QREP.POKA.TO.STLA.SENDQ', 'C', 'T', 'A', 'I',
 60, 2048, 'DT11', 'ITSO', 'QREP.POKA.TO.STLA.RECVQ', 'DSNT1');

-- COMMIT;
```

Example 3-49 Generated SQL for replication queue map RECVQ D8G1 to DT11

```
-- DatabaseDB2OS390 (DT11)

-- CONNECT TO DT11 USER XXXX using XXXX;


INSERT INTO ITSO.IBMQREP_RECVQUEUES
```

```
(repqmapname, recvq, sendq, adminq, capture_alias, capture_schema,
num_apply_agents, memory_limit, state, capture_server)
VALUES
('QMAP_POK_TO_STL', 'QREP.POKA.TO.STLA.RECVQ',
'QREP.POKA.TO.STLA.SENDQ', 'QREP.POKA.ADMINQ', 'D8G1', 'ITSO', 16, 30
, 'A', 'DB8G');


-- COMMIT;
```

> **Attention:** These scripts *must be executed before* the next step of creating the subscriptions.

Create the subscriptions

Example 3-50 lists the ASNCLP commands for creating the subscriptions on DT11 (STLABD1) and D8G1 (WTSC53). These commands should be saved in a file (ASNCLP_STL04_subs1.in).

Example 3-50 Create the subscriptions

```
#############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 3-1 on page 85, Table 3-2 on page 86 and Table 3-3 on page 88
############################################################
##DT11 from reference A.7a in Table 3-1 on page 85
##QREPADM from reference A.9 in Table 3-1 on page 85
##D8G1 from reference B.7a in Table 3-1 on page 85
##QREPADM from reference B.9 in Table 3-1 on page 85
##Capture schema ITSO from reference A.22 in Table 3-3 on page 88
##Apply schema ITSO from reference B.23 in Table 3-3 on page 88
##QMAP_STL_TO_POK from reference A.27 in Table 3-3 on page 88
##QMAP_POK_TO_STL from reference B.27 in Table 3-3 on page 88
#############################################################
SET SUBGROUP "000001";
SET SERVER MULTIDIR TO DB D8G1 ID "QREPADM" PASSWORD "xxxxxx";
SET SERVER MULTIDIR TO DB DT11 ID "QREPADM" PASSWORD "xxxxxx";

SET MULTIDIR SCHEMA D8G1.ITSO;
SET MULTIDIR SCHEMA DT11.ITSO;


SET CONNECTION SOURCE DT11.ITSO TARGET D8G1.ITSO REPLQMAP QMAP_STL_TO_POK;
SET CONNECTION SOURCE D8G1.ITSO TARGET DT11.ITSO REPLQMAP QMAP_POK_TO_STL;
##################################################################
## The following set of statements need to be defined for each table to
```

```
## be replication (from reference A.29 on Table 3-3 page 96
## SET TABLES = (A.7).(A.22).CREATORNAME.TABLENAME,
## (B.7).(B.22).CREATORNAME.TABLENAME
## where A.7 and B.7 are from reference Table 3-1 page 93
## and A.22 and B.22 are from reference Table 3-3 page 96
##
## The CREATE QSUB statements all have the following format:
##
## CREATE QSUB SUBTYPE A.26        <-- A.26 from reference table 3-3 page 96
## FROM NODE (A.7).(A.22           <-- A.7 from reference table 3-1 page 93
##                                 <-- A.22 from reference table 3-3 page 96
## SOURCE HAS LOADPHASE A.28.g     <-- A.28.g from reference table 3-3 page 96
## TARGET CONFLICT RULE B.28.d     <-- B.28.d from reference table 3-3 page 96
##        CONFLICT ACTION B.28.e   <-- B.28.e from reference table 3-3 page 96
##        ERROR ACTION B.28.f      <-- B.28.f from reference table 3-3 page 96
## FROM NODE (B.7).(B.22)          <-- B.7 from reference table 3-1 page 93
##                                 <-- B.22 from reference table 3-3 page 96
## SOURCE
## TARGET CONFLICT RULE A.28.d     <-- A.28.d from reference table 3-3 page 96
##        CONFLICT ACTION A.28.e   <-- A.28.e from reference table 3-3 page 96
##        ERROR ACTION A.28.f      <-- A.28.f from reference table 3-3 page 96
##
###############################################
set tables (DT11.ITSO.ITSO.BAL,  D8G1.ITSO.ITSO.BAL);
   create qsub subtype b
   from node DT11.ITSO
   source has load phase i
   target conflict rule a
          conflict action i
          error action s
   from node D8G1.ITSO
   source
   target conflict rule a
          conflict action f
          error action s;

   set tables (DT11.ITSO.ITSO.BPT_PARM_TABLE,  D8G1.ITSO.ITSO.BPT_PARM_TABLE);
   create qsub subtype b
   from node DT11.ITSO
   source has load phase i
   target conflict rule a
          conflict action i
          error action s
   from node D8G1.ITSO
   source
   target conflict rule a
          conflict action f
          error action s;
```

The ASNCLP script to execute this multiple subscription script is shown in Example 3-51. When this script is executed from a command prompt, it forces the execution of the CREATE statements in the first script above, and SQL is generated that can be ported to the z/OS systems for execution under SPUFI or DSNTEP2.

Example 3-51 Execute multiple subscription script

```
ASNCLP SESSION SET TO Q REPLICATION;
SET OUTPUT MULTIDIR;
SET RUN SCRIPT LATER;
LOAD MULTIDIR REPL SCRIPT "ASNCLP_STL04_subs1.in";
```

Start the Q Capture and Q Apply programs

"Step 6f: Start the Q Capture on both servers" on page 173, "Step 6g: Start the Q Apply on both servers" on page 176, and "Step 6h: Verify status of Q Capture and Q Apply processes" on page 178 describe the steps involved in starting Q Capture and Q Apply, and verifying their status.

3.5.8 Step 7: Set up monitoring environment using GUI

We recommend setting up the monitoring environment with appropriate alert conditions for the bidirectional replication environment to ensure a stable operating environment,

Figure 3-81 lists the steps involved in setting up the monitoring environment.

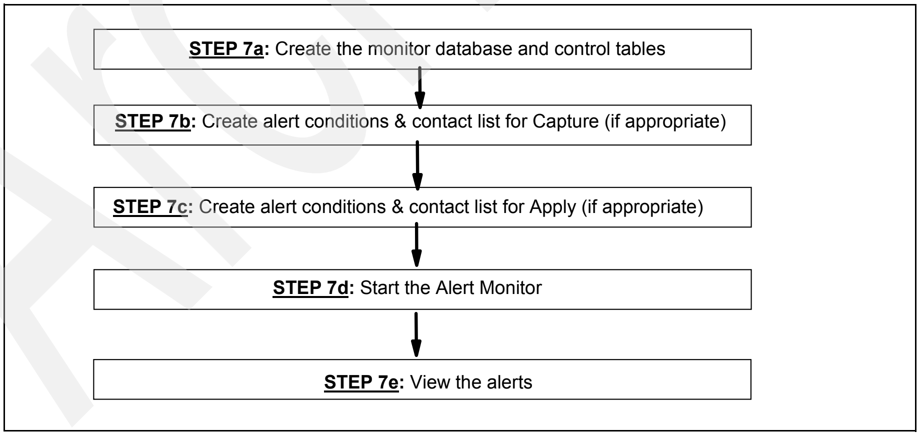


Figure 3-81 Steps to set up monitoring environment using GUI

To ensure a smooth operational Q replication environment, it is necessary to set up the Replication Alert Monitor to monitor the Q Capture and Q Apply processes as well as the efficacy of the replication. An overview of the Replication Alert Monitor is given in 2.1.4, “Replication Alert Monitor” on page 21.

The Replication Alert Monitor provides basic monitoring of the following:

- Q Capture program
  - Status
  - Errors
  - Warnings
  - Latency
  - Memory
  - Transaction size
  - Subscriptions inactivated
- Q Apply program
  - Status
  - Errors
  - Warnings
  - Latency
  - End-to-end latency
  - Memory
  - Exceptions
  - Spill queue depth
  - Receive queue depth

In this section we document the step-by-step setup of the Replication Alert Monitor using the GUI interface, to manage the Q Capture and Q Apply processes in the DT11 and D8G1 database servers.

Figure 3-82 lists the main steps involved in setting up this environment. Each of these steps is described in detail in the following sections.

> **Important:** It is assumed that the user ID QREPADM (reference M.5c in Table 3-4 on page 89) has been created with the appropriate database privileges on the server (reference M.1 in Table 3-4 on page 89) to be configured. If not, then this task must first be performed.

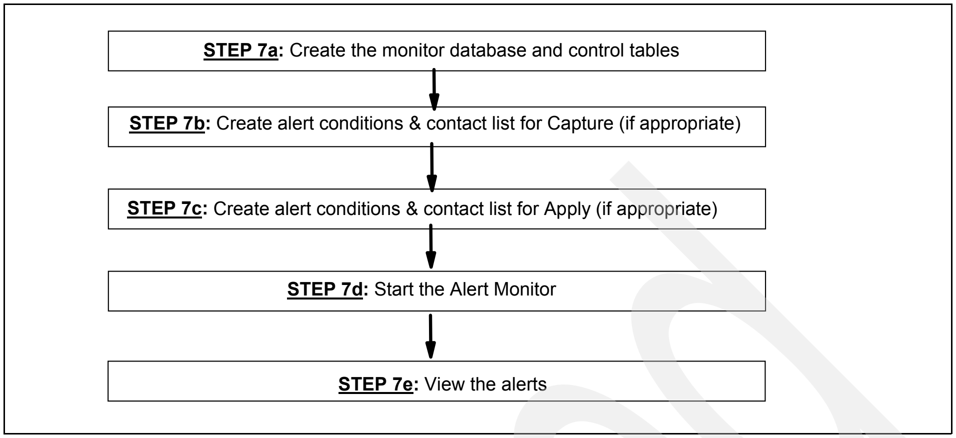


Figure 3-82 Overview of Replication Alert Monitor setup steps

Step 7a: Create the monitor database and control tables

We chose to create the monitor control tables in the D8G1 database server (reference M.5a) on the WTSC53 (reference M.1), and monitor both the DT11 and D8G1 database servers from this server.

> **Note:** A single monitor can report on two or more servers in the replication environment as long as the appropriate DRDA connections are in place. Refer to Example 3-12 on page 98 for details on a DRDA setup.

Invoke the Replication Center and right-click the **Monitor Control Servers** folder in the navigation pane. Then select **Create Monitor Control Tables...** (as shown in Figure 3-83) to bring up a list of potential servers wherein the Monitor Control Tables may be created (as shown in Figure 3-84).

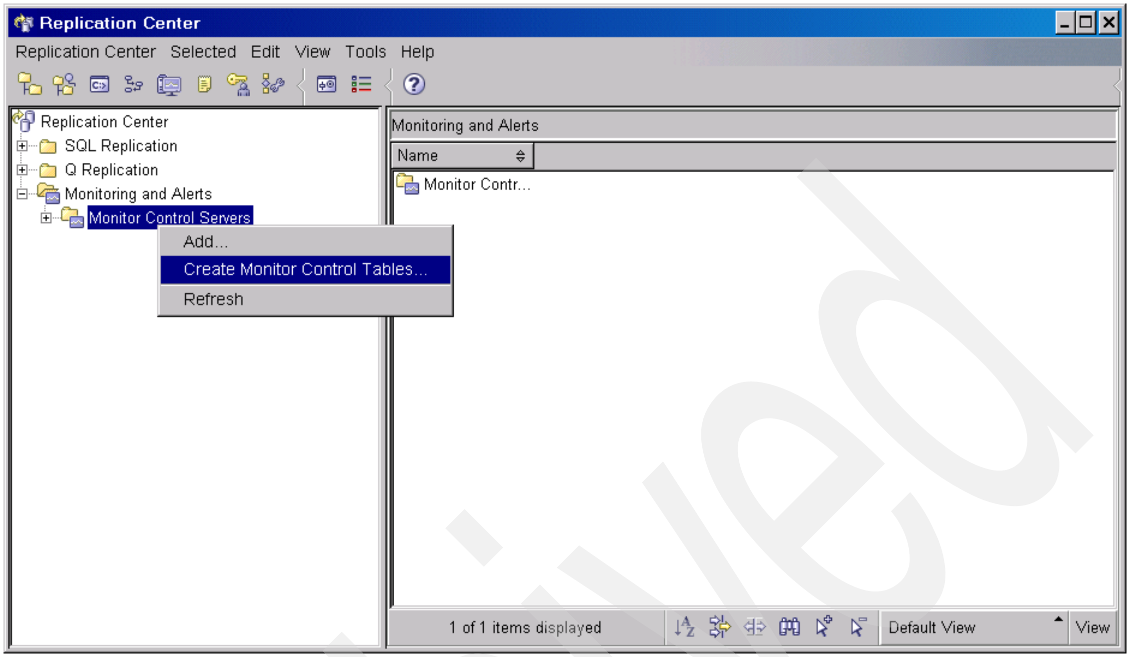


Figure 3-83 Create the Monitor Control Tables

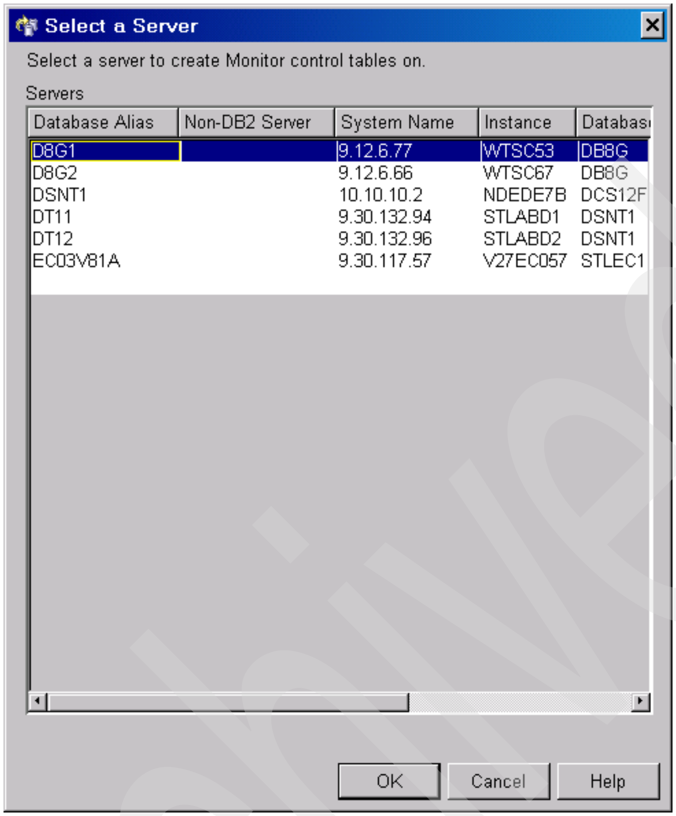


Figure 3-84 Select a server to create the monitor control tables

Select **D8G1** (reference B.7 in Table 3-1 on page 85) on WTSC53 as the target for the Monitor Control Tables, and click **OK** to proceed to Figure 3-85 in order to provide appropriate information for these tables.

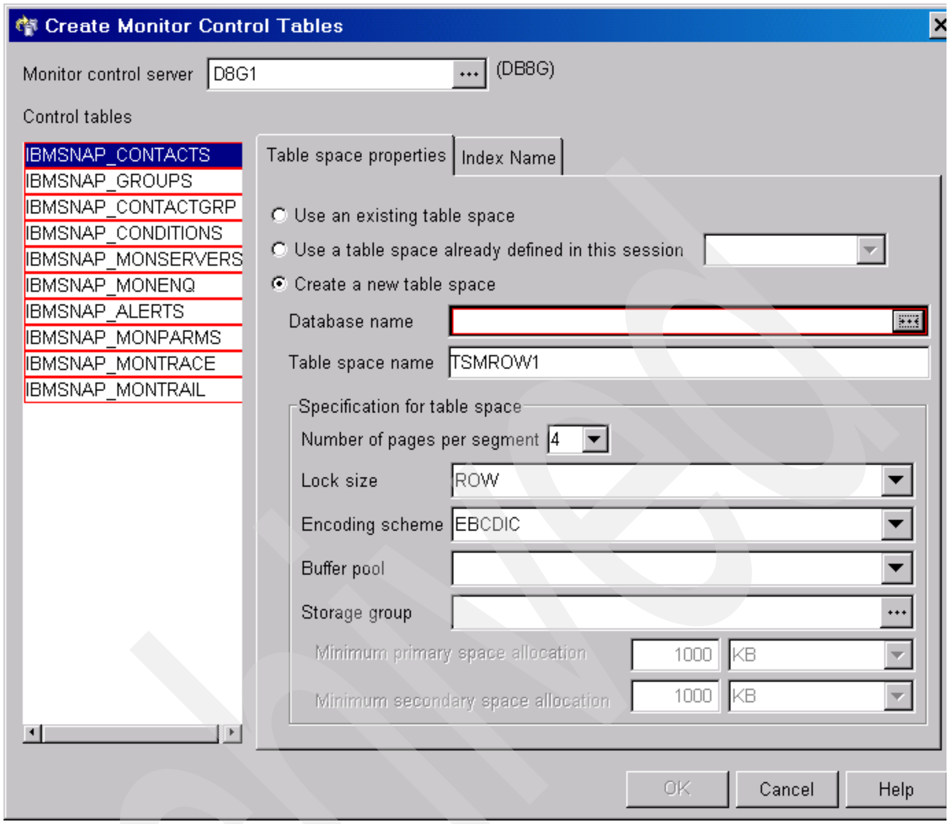


Figure 3-85 Tablespace properties

The tablespace properties must be provided for each of the ten monitor tables listed on the left of Figure 3-85. Click the **...** drop-down box to display Figure 3-86 in order select the logical database (reference B.10 in Table 3-1 on page 85) where the monitor tables will be placed.

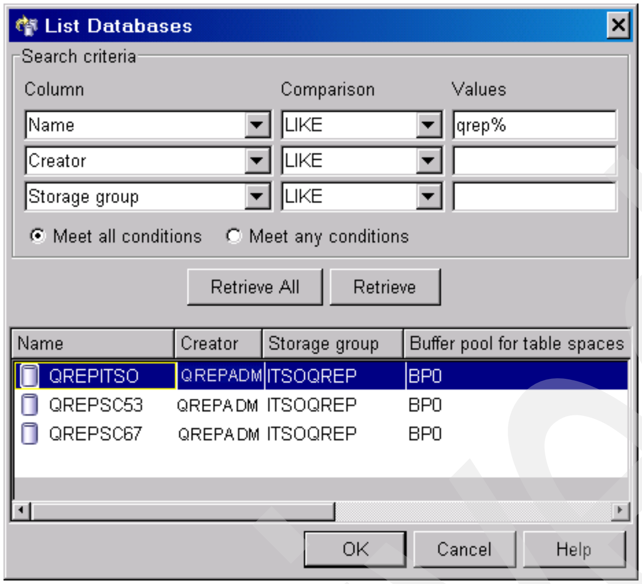


Figure 3-86 List Databases

Note: The logical database on which to build the monitor control tables must exist (Example 3-5 on page 91) prior to using the Replication Center to configure a monitor.

In Figure 3-86, specify the selection criteria of all databases beginning with qrep% and click the **Retrieve** button to view the list of databases that qualify as shown.

Select the **QREPITSO** database and click **OK** to proceed to Figure 3-87.

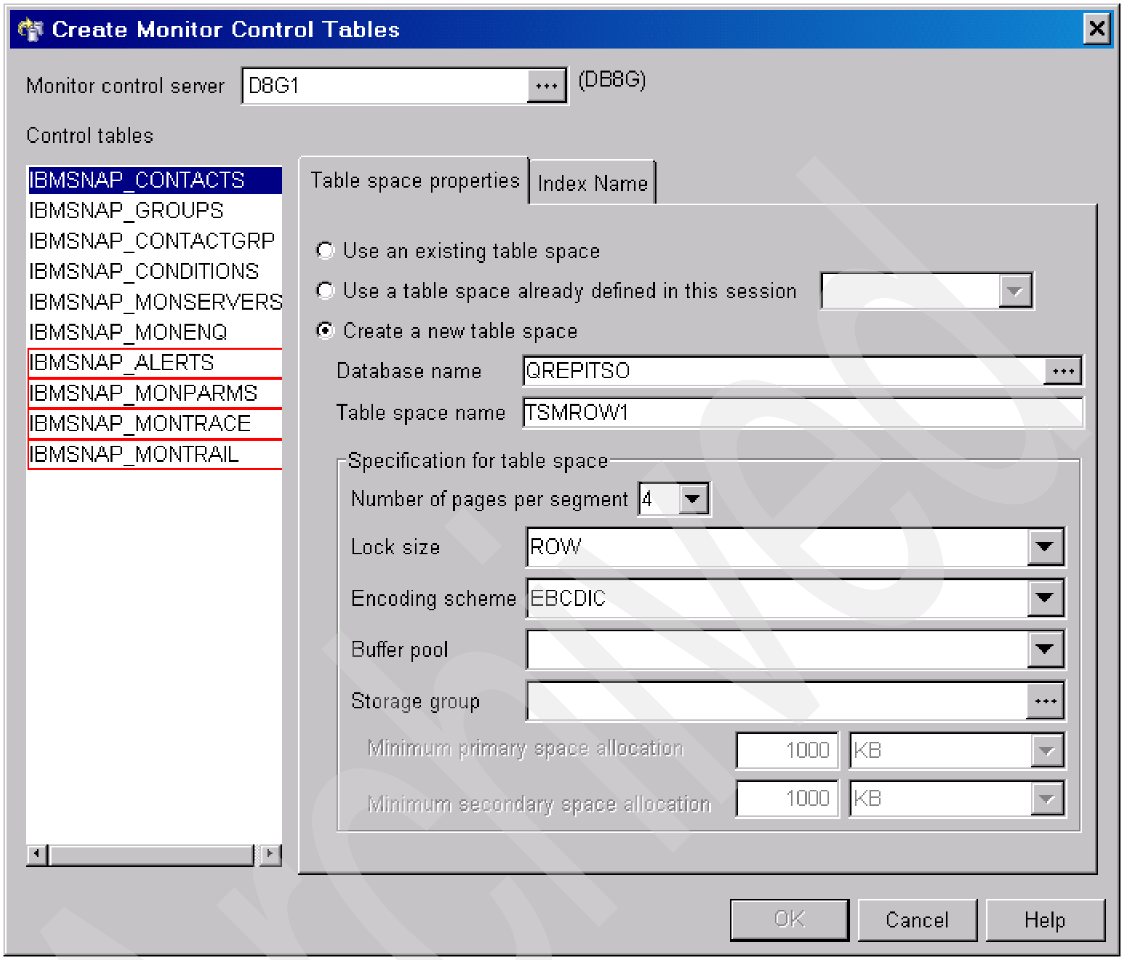


Figure 3-87 Tablespace properties

The first six tables in Figure 3-87 can all be placed in this tablespace TSMROW1 by default. Highlight the seventh table **IBMSNAP_ALERTS**, which (by default) will be placed in another tablespace, as shown in Figure 3-88.

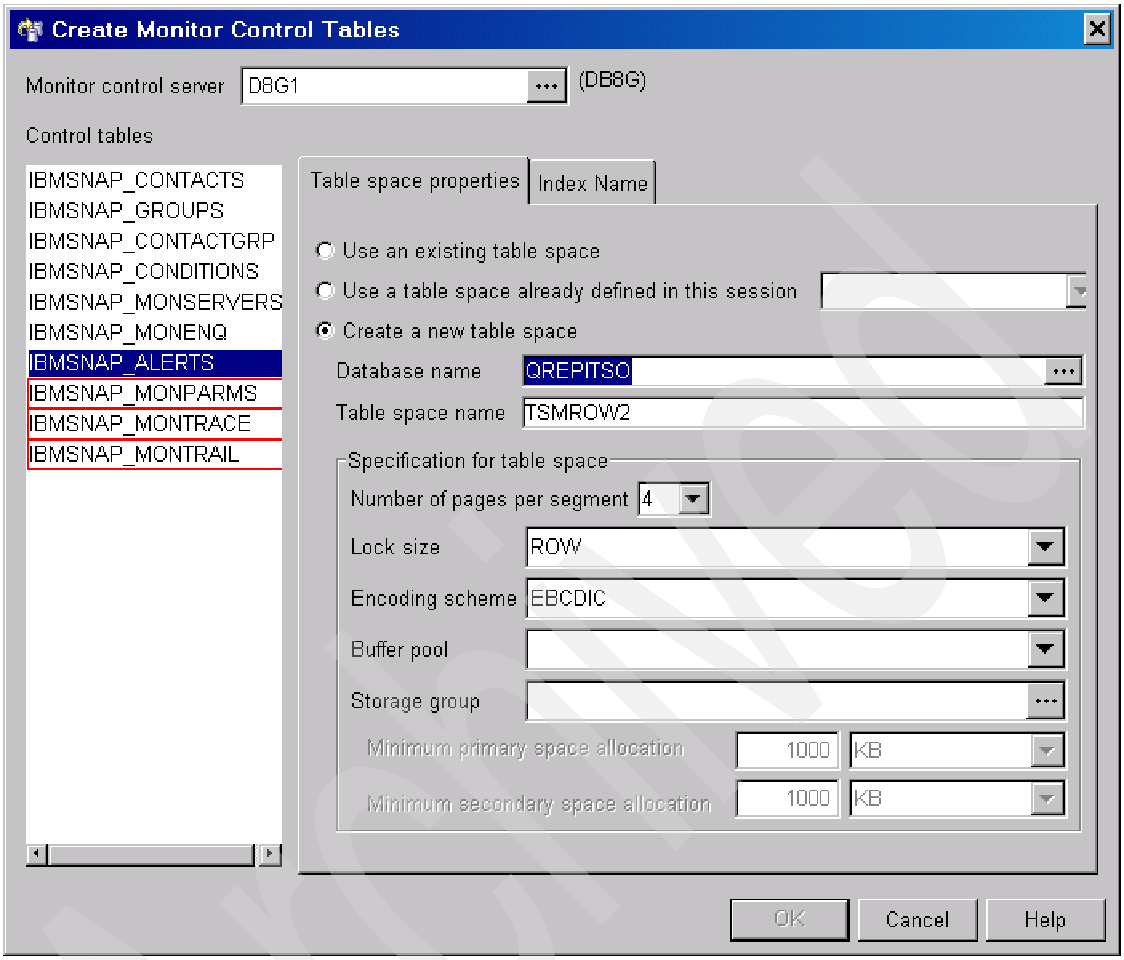


Figure 3-88 Tablespace properties

In Figure 3-88, the IBMSNAP_ALERTS table is placed in the TSMROW2 tablespace by default. Highlight the eighth table **IBMSNAP_MONPARMS**, which (by default) will be placed in another tablespace, as shown in Figure 3-89.

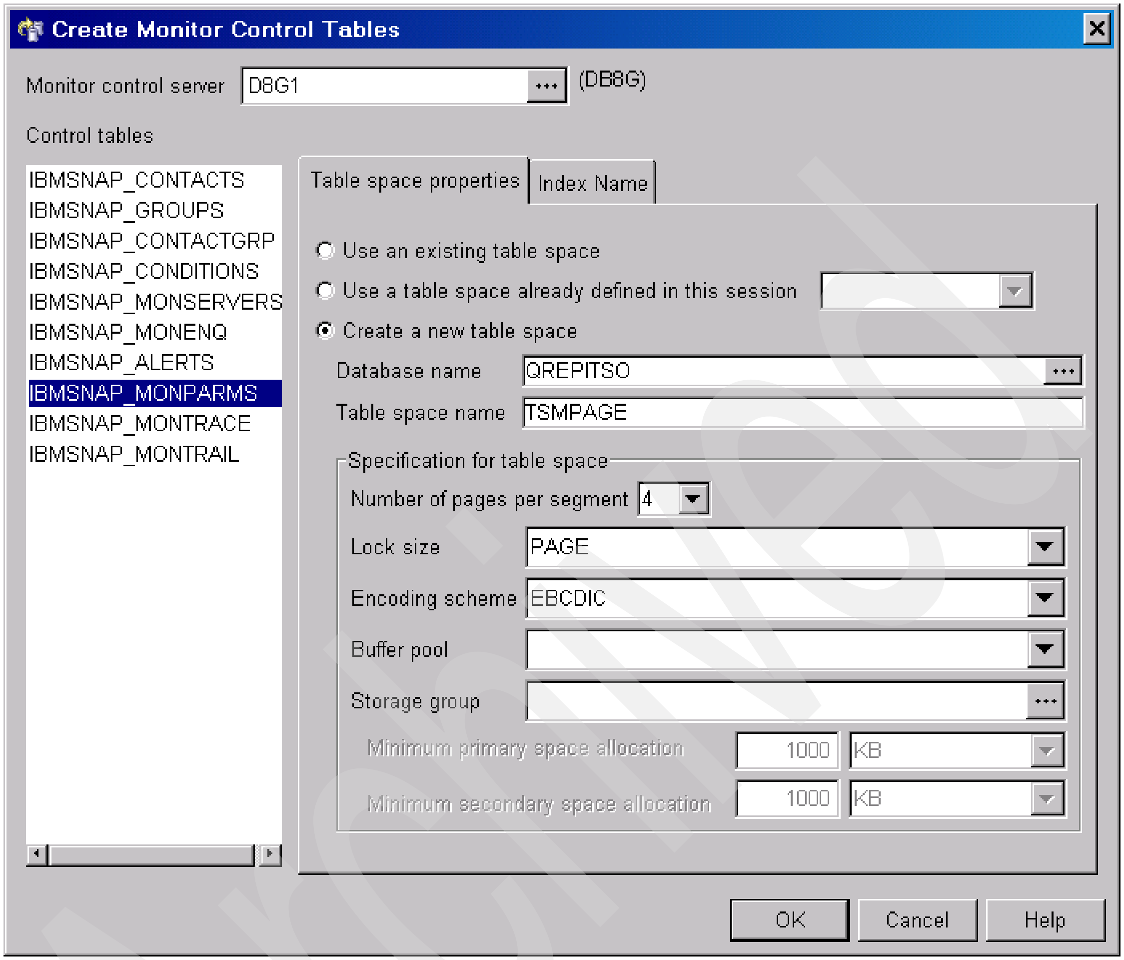


Figure 3-89 Tablespace properties

In Figure 3-89, select the drop-down box and select the database for this table as before. The last three tables will be placed in this tablespace, TSMPAGE, by default. Click the **Index Name** tab to proceed to Figure 3-90.

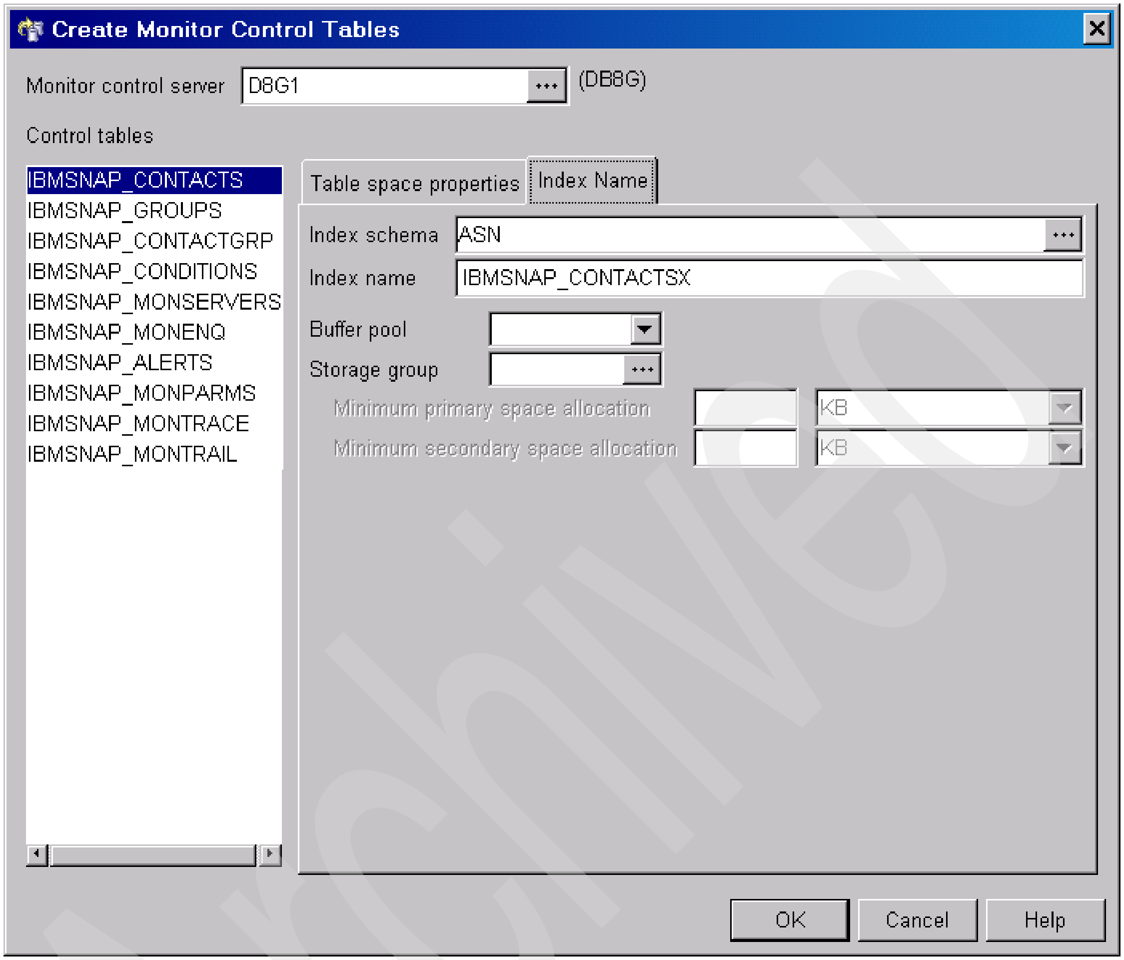


Figure 3-90 Index details

Indicate the index names to use for these tables, or allow the default names. The last table on the list, IBMSNAP_MONTRAIL, has no indexes; therefore, the screen automatically returns to the Tablespace properties tab when it is highlighted. When all tablespace and index name information is complete, click **OK** to generate the SQL script to create these control tables.

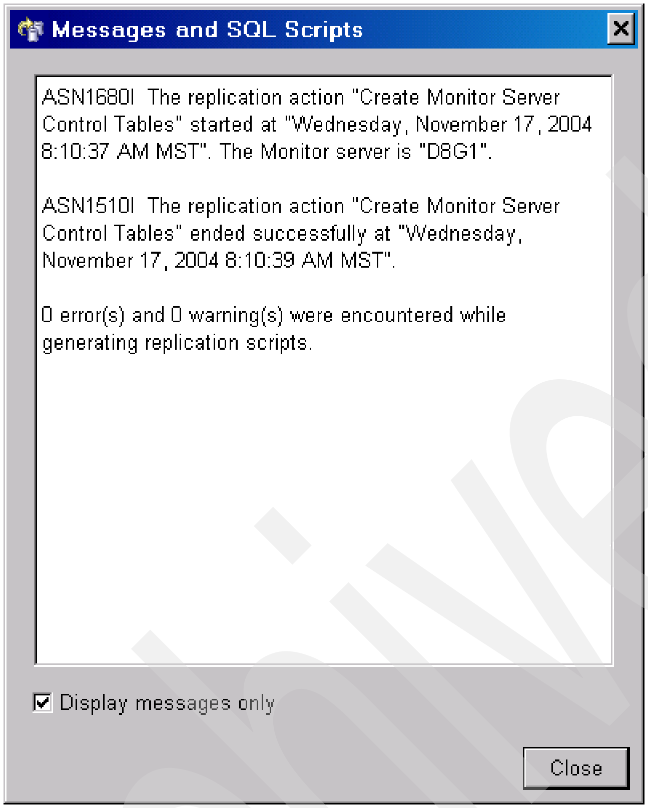


Figure 3-91 Messages and SQL Scripts

Figure 3-91 shows the successful generation of the script required to create the monitor control tables. Click **Close** to be provided with the choice of executing the script right away or saving it for later execution, as shown in Figure 3-92.

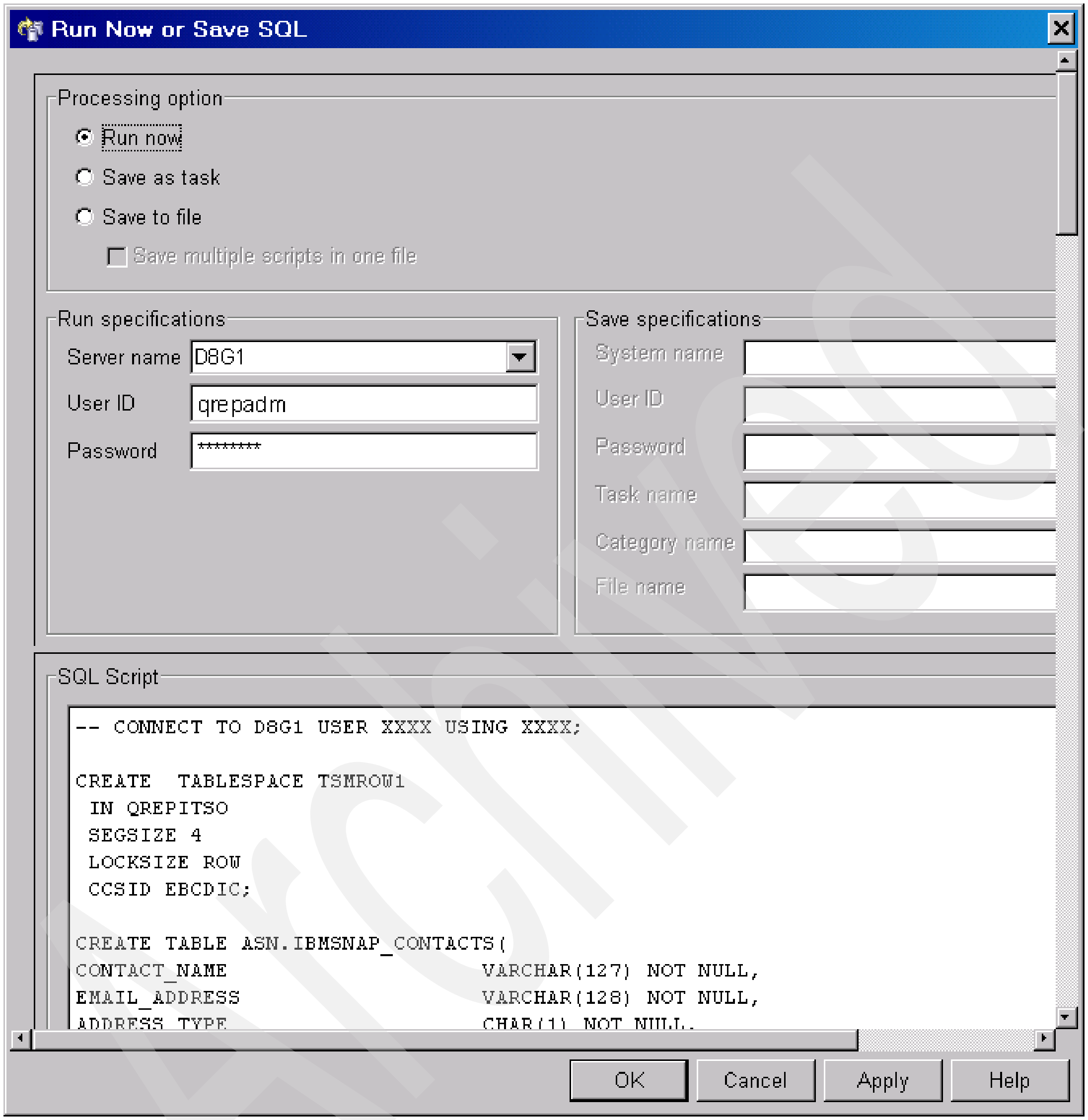


Figure 3-92 Run Now or Save SQL

Select **Run Now** and click **OK** to execute the generated script immediately. Figure 3-93 displays the successful execution of the generated script on the D8G1 server. Alternatively, the script may be saved to a file, and ported to z/OS for execution under SPUFI or DSNTEP2.

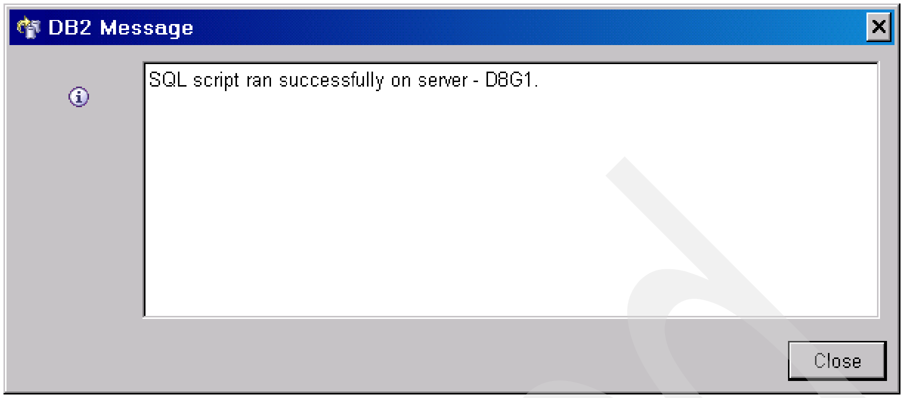


Figure 3-93 SQL script ran successfully message

The control tables are now created. Click **Close**.

Step 7b: Create the alert conditions & contact list for capture

The alert conditions to be monitored for and the contact list to be used in the event of alerts related to Q Capture are defined in this step. Figure 3-94 shows the Replication Center with the Monitor Control Servers folder displayed.

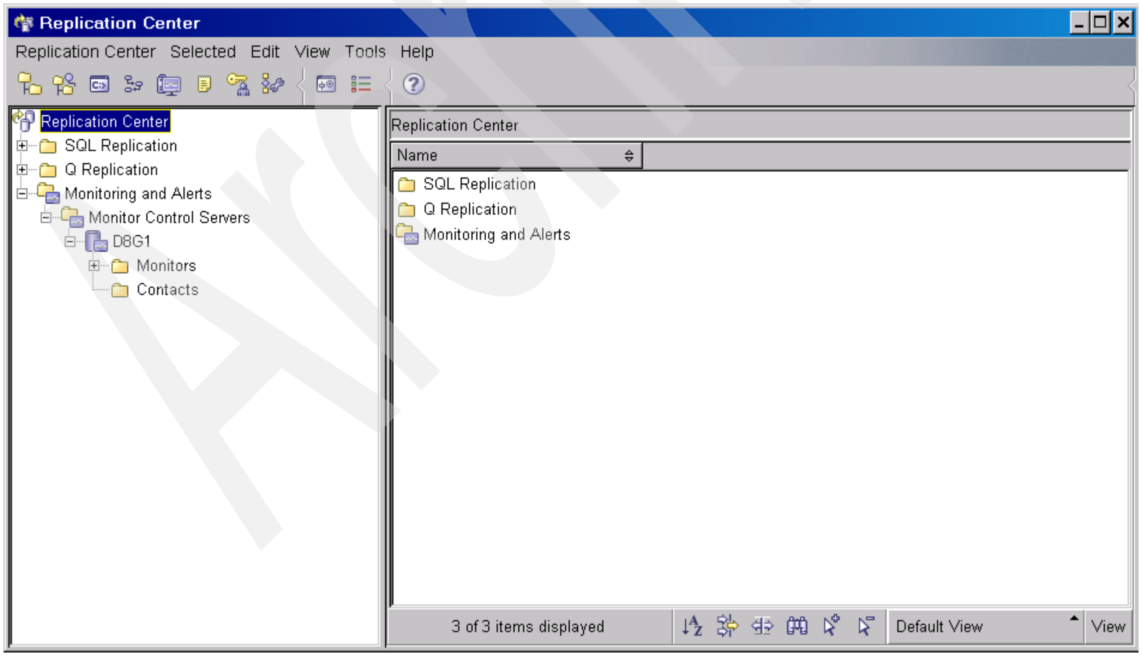


Figure 3-94 Monitor Control Servers

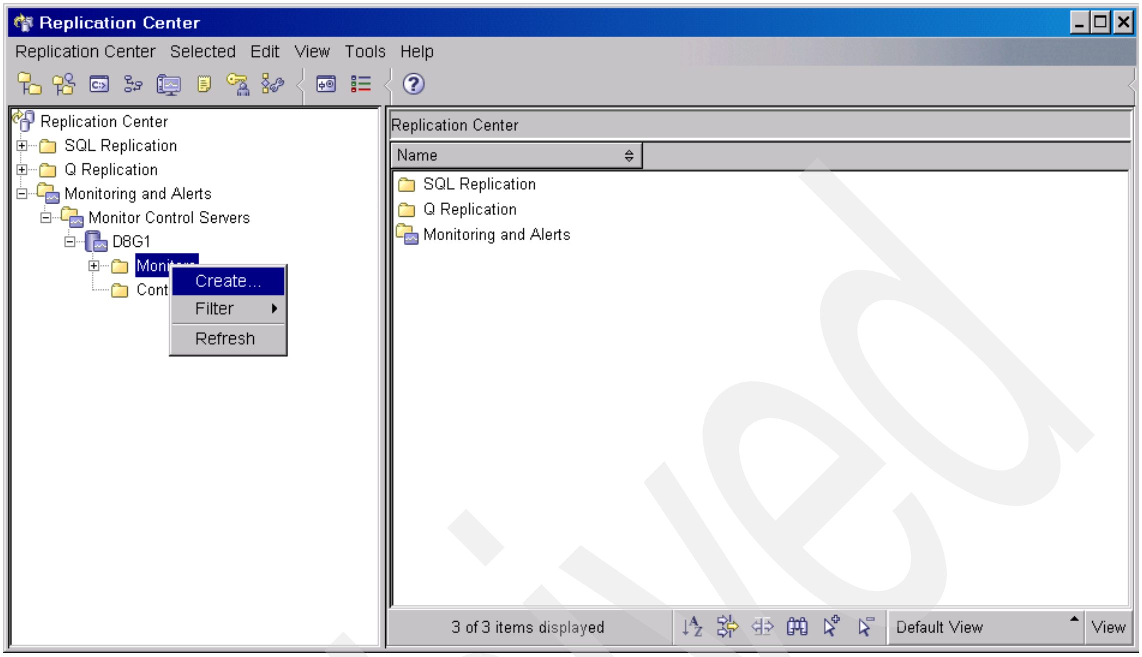


Figure 3-95 Create Monitors

Right-click the **Monitors** folder and select **Create...**, as shown in Figure 3-95, to proceed to the Create Monitor Wizard shown in Figure 3-96.

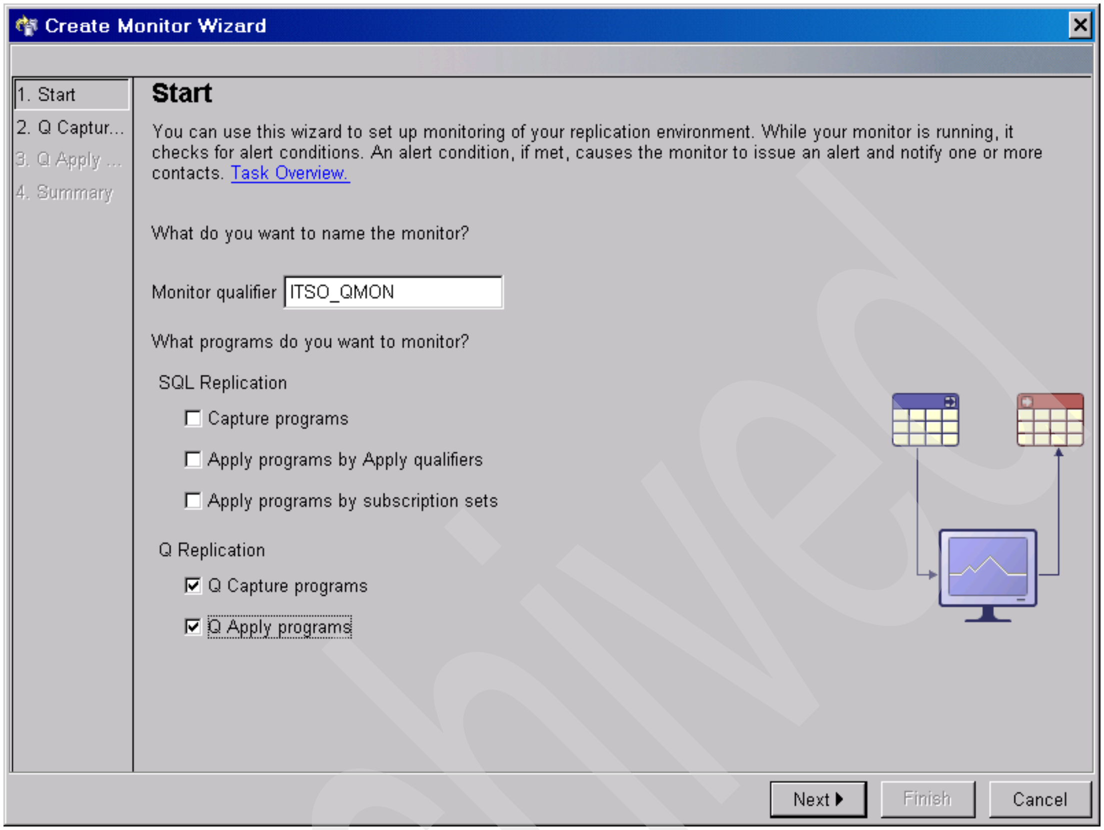


Figure 3-96 Create Monitor Wizard

A monitor qualifier is a grouping of monitors. One can have multiple monitors to monitor multiple programs. These multiple monitors are identified via the monitor qualifiers. We chose to monitor Q Capture and Q Apply programs in Q Replication using a single monitor qualifier, ITSO_QMON (reference M.5f in Example 3-4 on page 89), entered in the Monitor qualifier field. We therefore selected the **Q Capture programs** and **Q Apply programs** settings under Q Replication, and clicked **Next** to select the alert conditions for Q Capture programs in Figure 3-97.

> **Note:** The Replication Alert Monitor has the schema name of ASN, which cannot be changed. Since the Replication Alert Monitor can monitor SQL replication as well as Q Replication, the control tables start with IBMSNAP instead of IBMQREP.

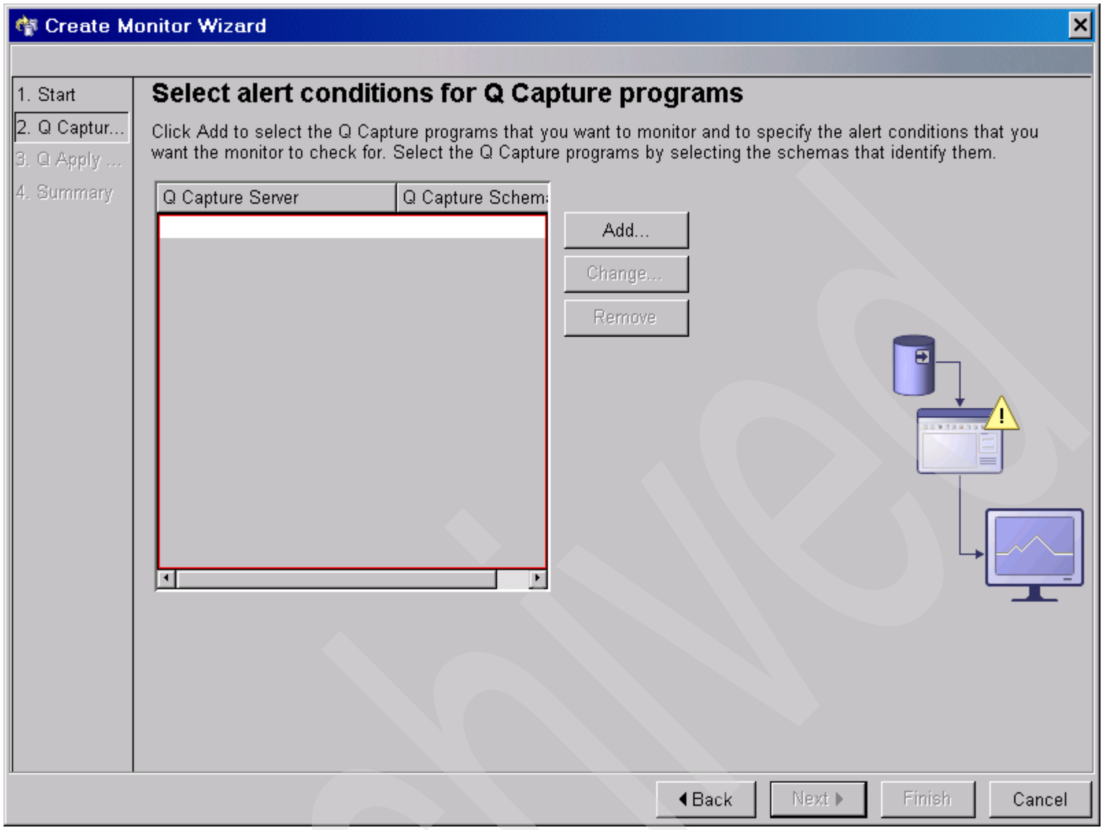


Figure 3-97 Select alert conditions for Q Capture programs

The Q Capture Server list is empty since this is the first time the create monitor has been invoked; therefore, candidate Q Capture Servers need to be added. This is done by clicking the **Add...** button (Figure 3-97) to select the Q Capture program and the alert conditions to be monitored, as shown in Figure 3-98.

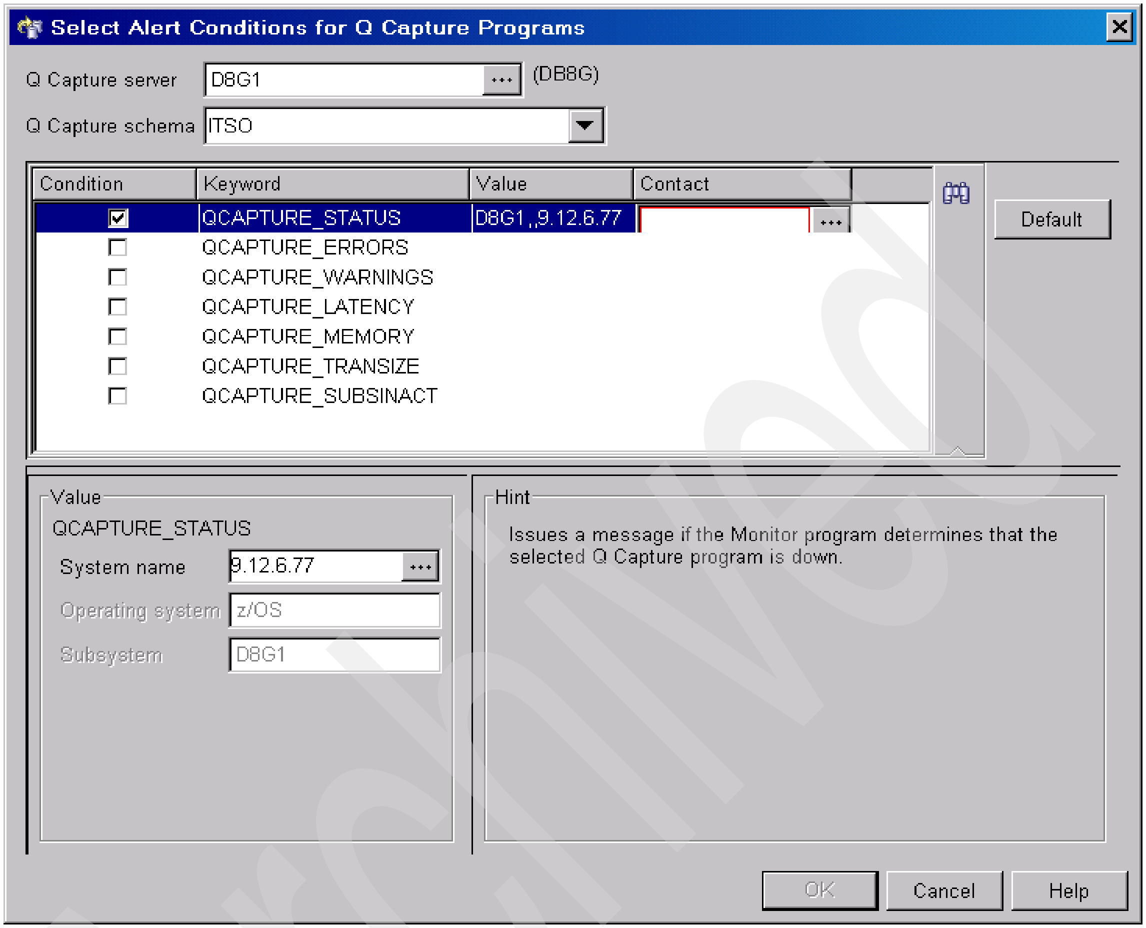


Figure 3-98 Select Q Capture program and alert conditions

Supply information about the Q Capture Server (D8G1 reference B.7 in Table 3-1 on page 85), Q Capture schema (ITSO reference B.22 in Table 3-3 on page 88), and check the alert conditions to monitor, such as QCAPTURE_STATUS. Some alert conditions require additional parameters such as system name.

Click the **...** button under the column Contact to specify the contact person for each alert condition, as shown in Figure 3-99.

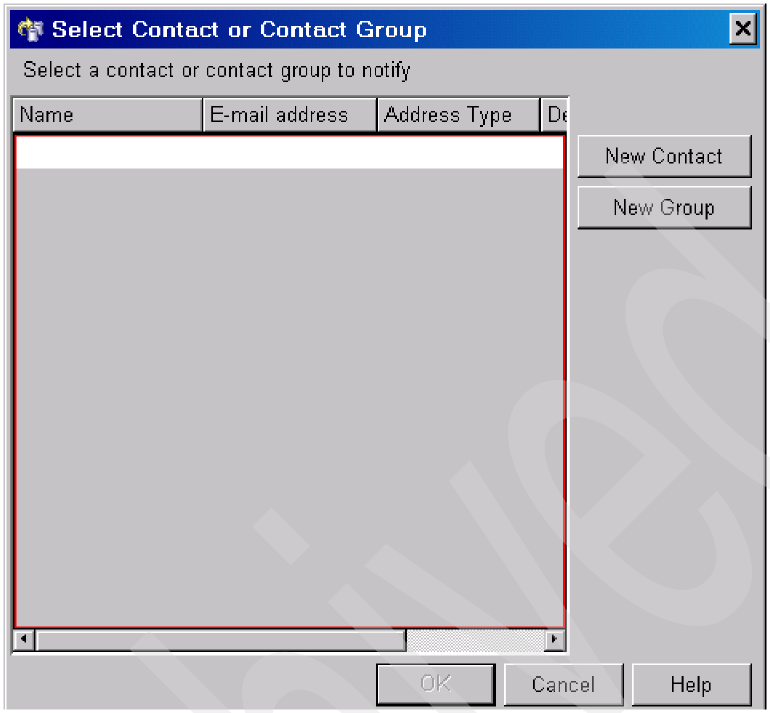


Figure 3-99 Select Contact or Contact Group

Since no contact information list is available, new contacts need to be created by clicking the **New Contact** button in Figure 3-99, which displays Figure 3-100.

Note: Contact groups may also be created here, that are collections of contacts that receive the same alerts.

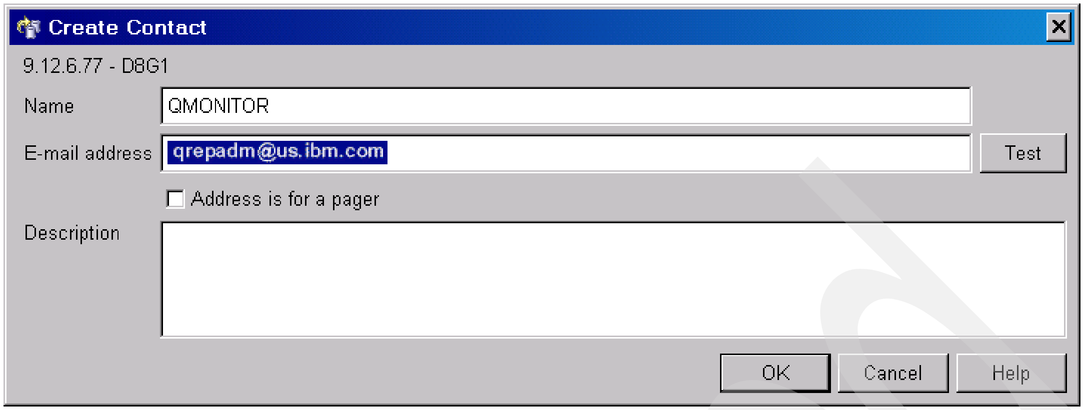


Figure 3-100 Create Contact

Specify any name for the contact in Figure 3-100 in the Name field (reference M.5d in Table 3-4 on page 89), and provide an e-mail address of the contact person in the E-mail address field (reference M.5e in Table 3-4 on page 89) where the alert message should be delivered by the Replication Alert Monitor.

The Test button provides a quick way of determining whether the mail server is running or the address entered is correct and can be used.

Check **Address is for a pager** if appropriate.

Click **OK** to generate the script for defining the contact information specified, as shown in Figure 3-101.

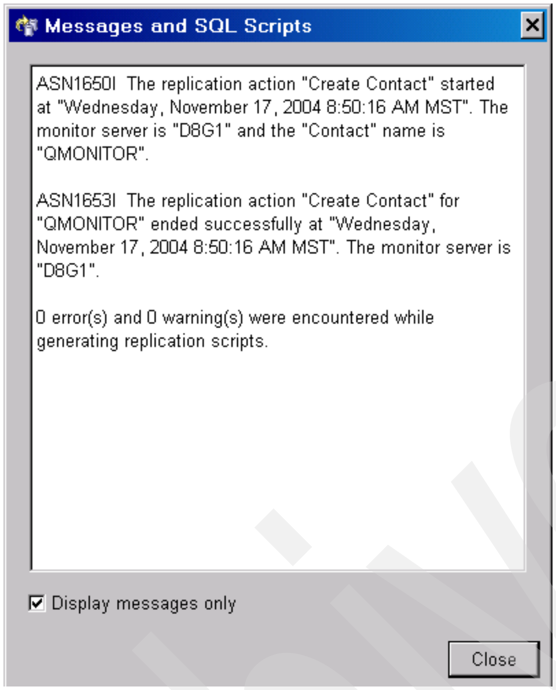


Figure 3-101 Message and SQL Scripts

Check that the errors are 0. Click **Close** to display Figure 3-102 to decide whether to execute the script or save it for later execution.

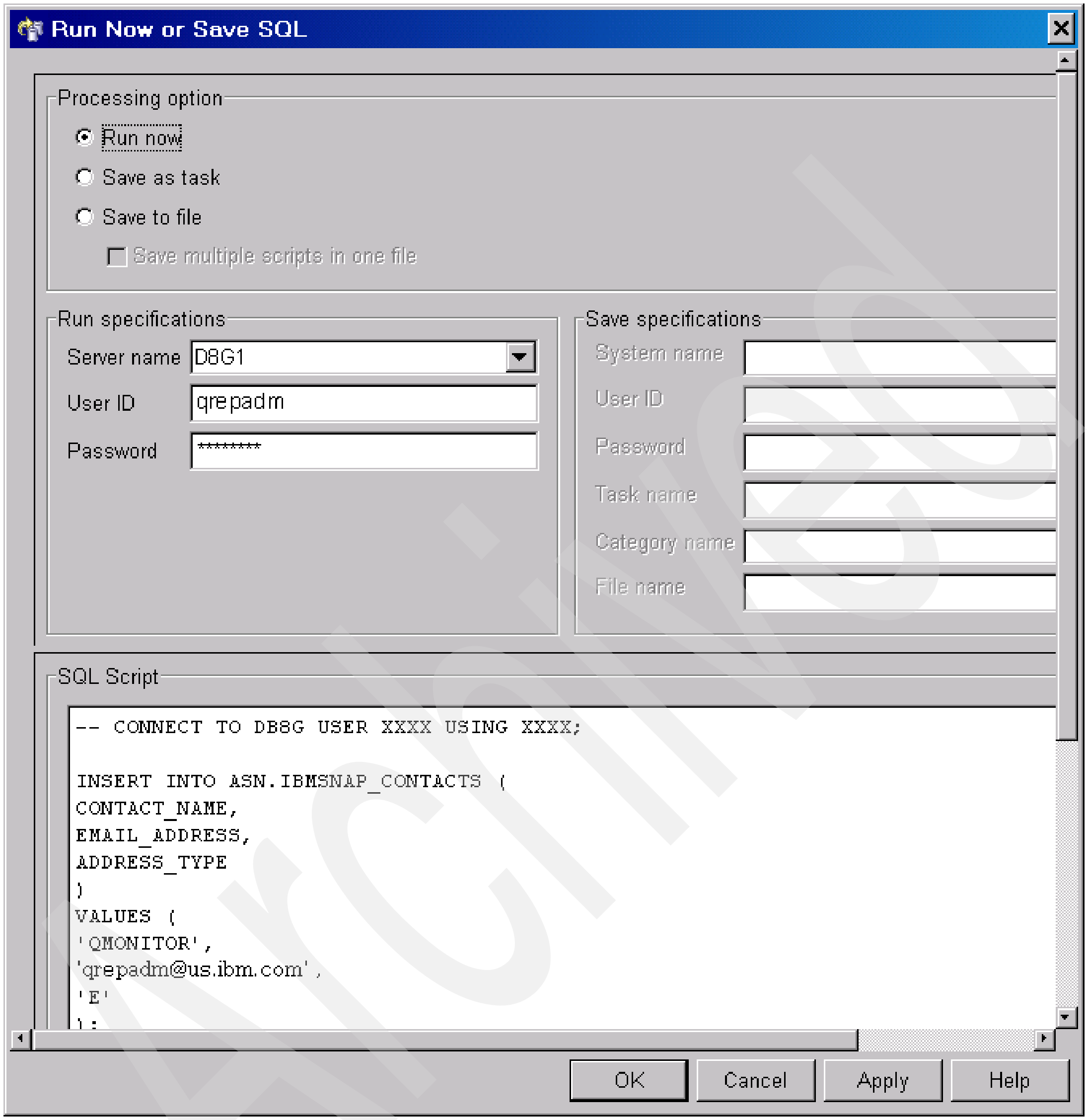


Figure 3-102 Run Now or Save SQL

Select the **Run now** option and click **OK** to display the message that the scripts to generate the contact information objects executed successfully, as shown in Figure 3-103. Alternatively, the script may be saved to a file, and ported to z/OS for execution under SPUFI or DSNTEP2.

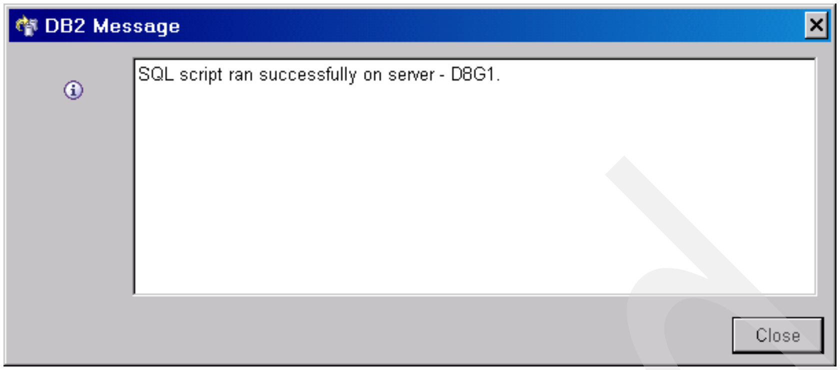


Figure 3-103 SQL script ran successfully message

Click **Close** to continue to Figure 3-104, which shows the newly added contact information.

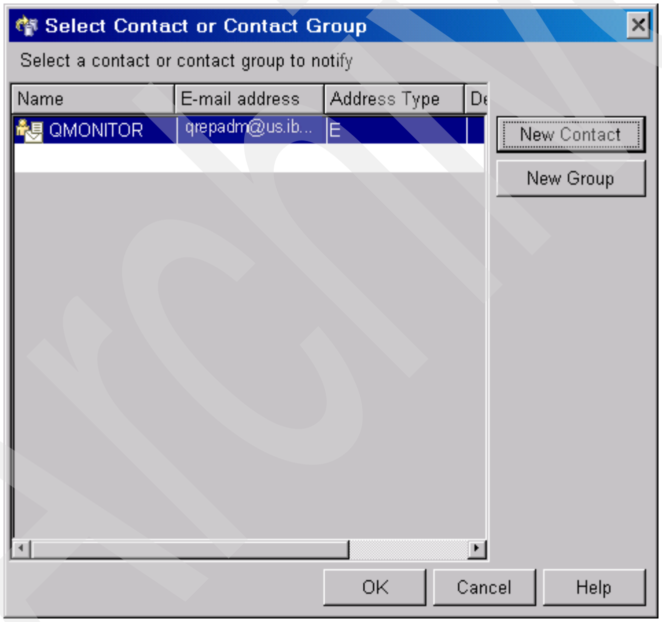


Figure 3-104 Select Contact or Contact Group

Select the contact just created and click **OK** to proceed to Figure 3-105, which lists the rest of the alert conditions for possible selection.

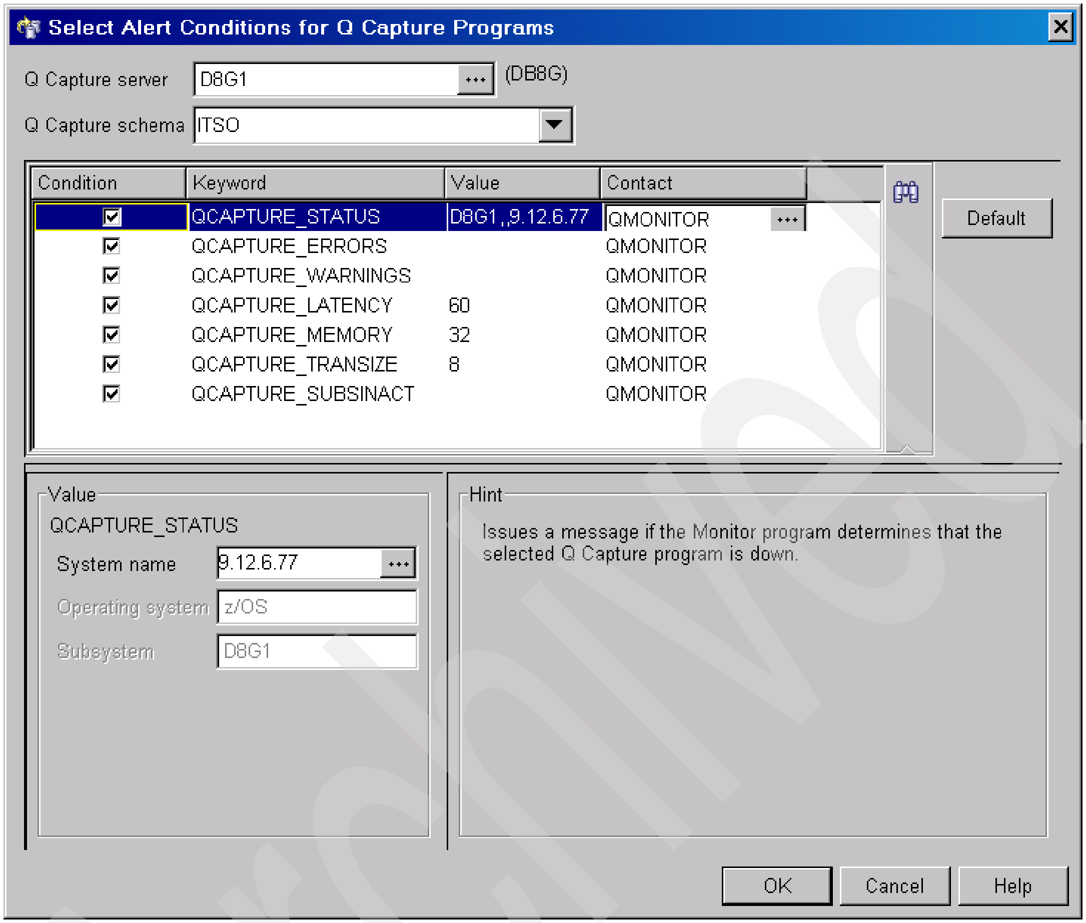


Figure 3-105 Alert Conditions

We selected all the alert conditions and chose the same contact name created earlier for each of them. Click **OK** to proceed to Figure 3-106.

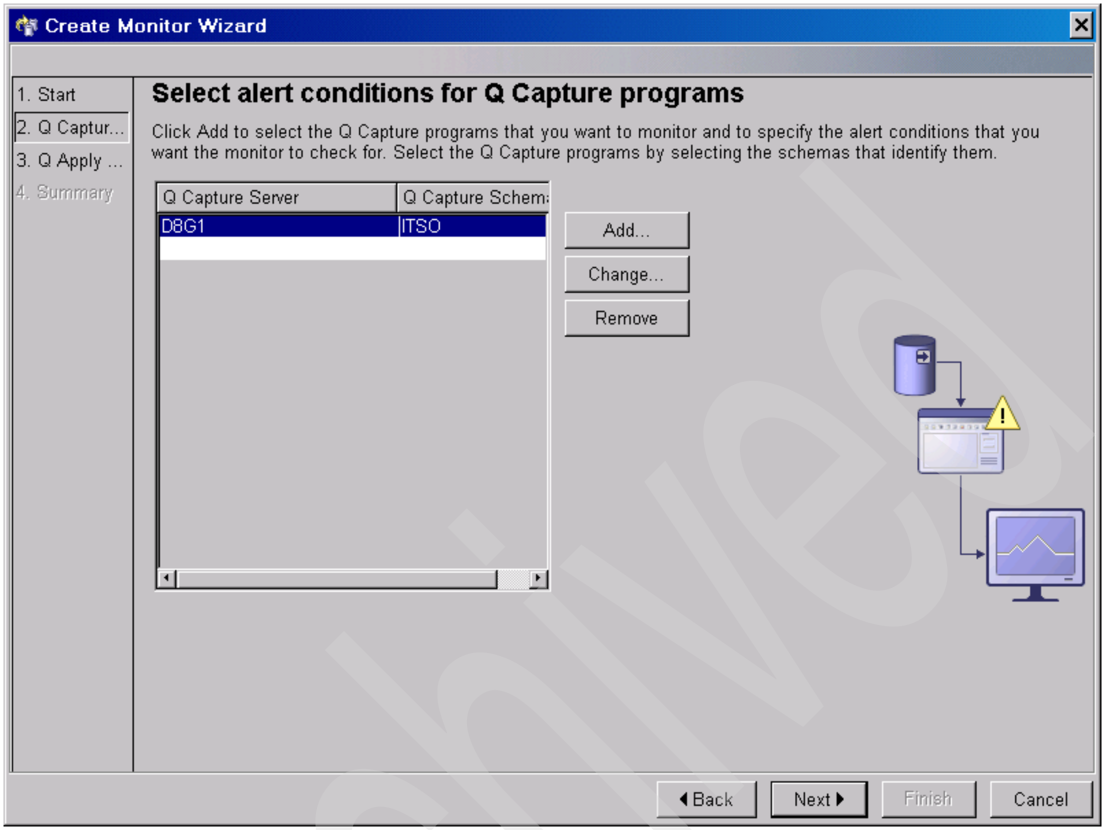


Figure 3-106 Select alert conditions for Q Capture programs

Additional Q Capture Servers may be added in Figure 3-106 by clicking the **Add...** button. Since we needed to monitor both the Q Capture servers in our environment, we clicked **Add...** to include the DT11 Q Capture Server, as shown in Figure 3-107.

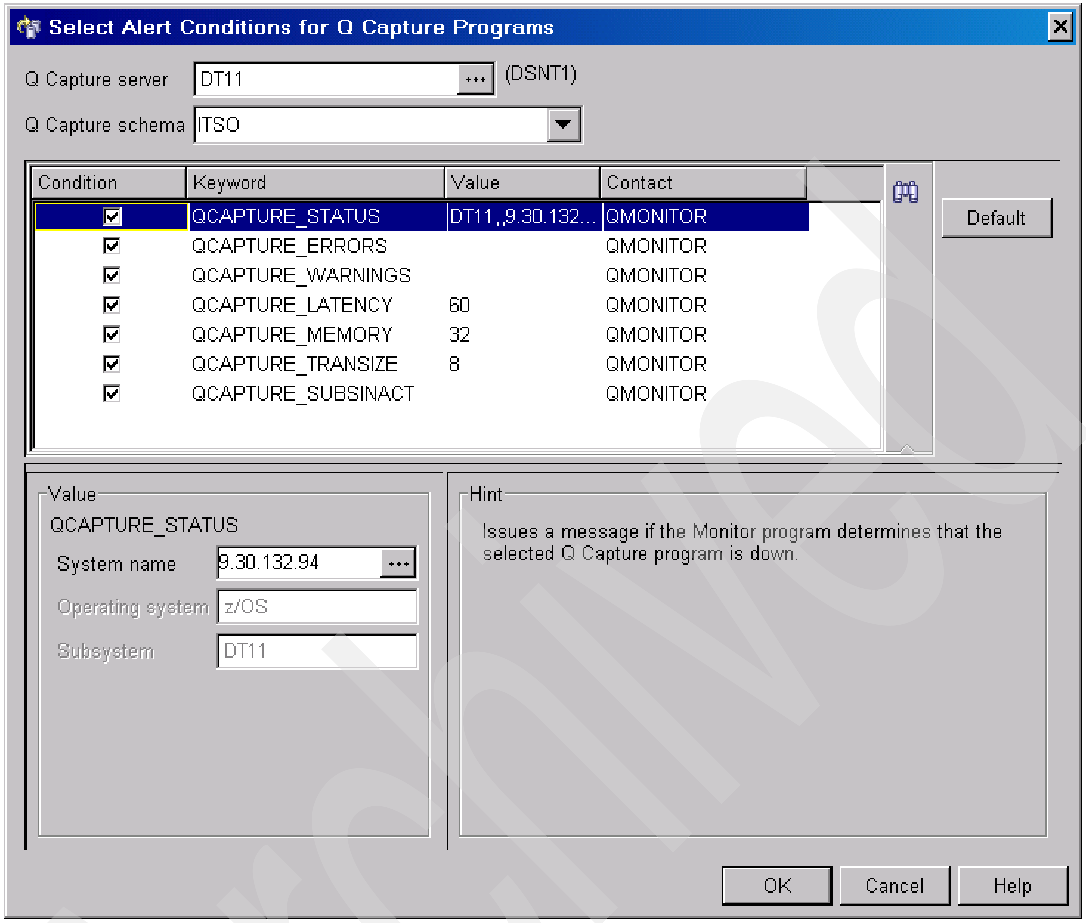


Figure 3-107 Select Alert Conditions for Q Capture Programs

Supply information for the “DT11” Q Capture Server and the “ITSO” Q Capture schema, and select all the alert conditions and associate the same contact name defined earlier for each one of them. The System name (“9.30.132.94”) information needs to be provided, then click **OK** to proceed to Figure 3-108, which completes the monitoring configuration for the Q Capture programs on the DT11 and D8G1 database servers.

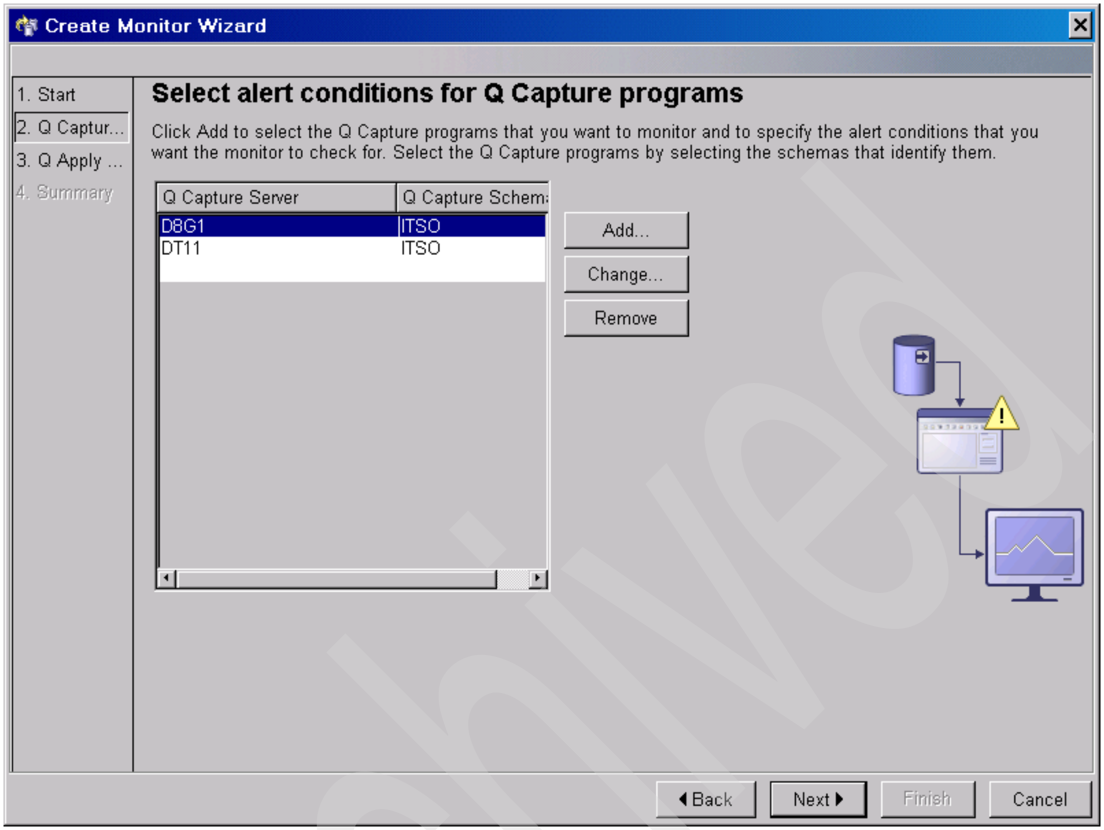


Figure 3-108 Select alert conditions for Q Capture programs

Click **Next** to set the alert conditions for the Q Apply programs.

Step7c: Create the alert conditions and contact list for Apply

The same series of steps performed for setting the alert conditions for the Q Capture program (as described in Figure 3-96 on page 245 through Figure 3-108) should also be performed for the Q Apply program, as shown in Figure 3-109 through Figure 3-113 on page 261.

Once all the alert conditions are provided, the Create Monitor Wizard generates a summary of all the information provided, as shown in Figure 3-114 on page 262.

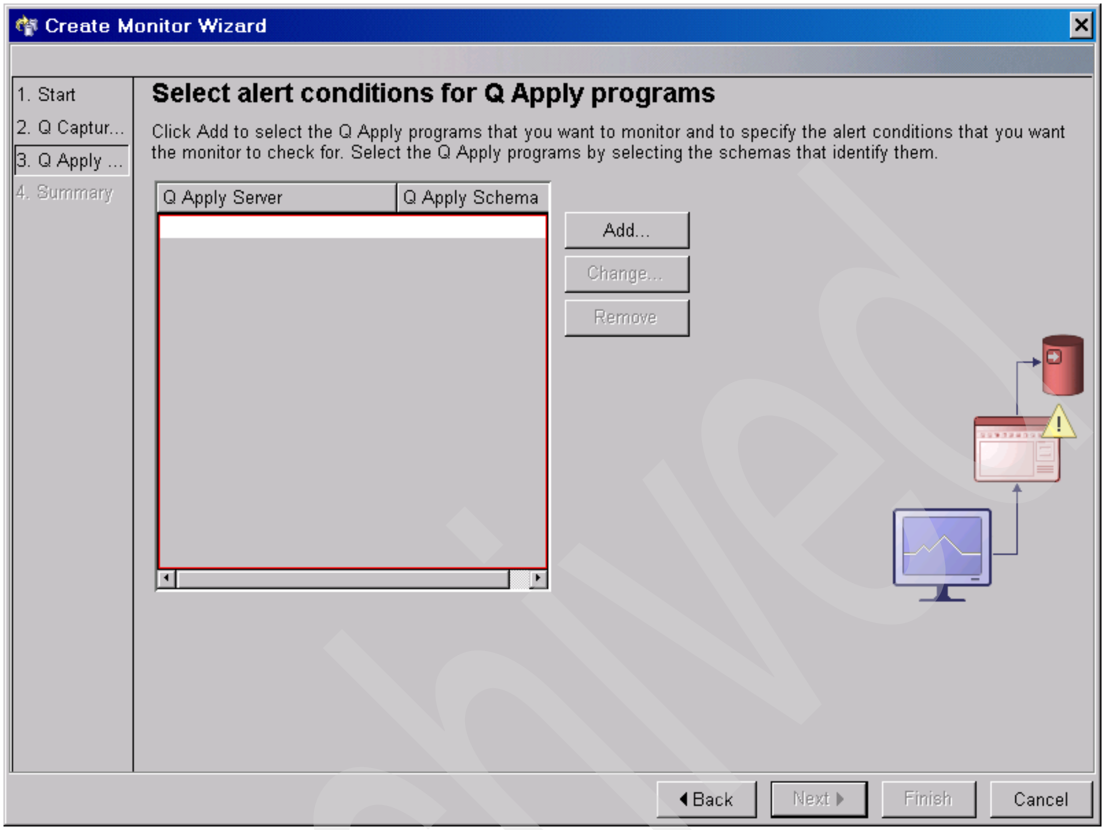


Figure 3-109 Create Monitor Wizard - Select alert conditions for Q Apply programs

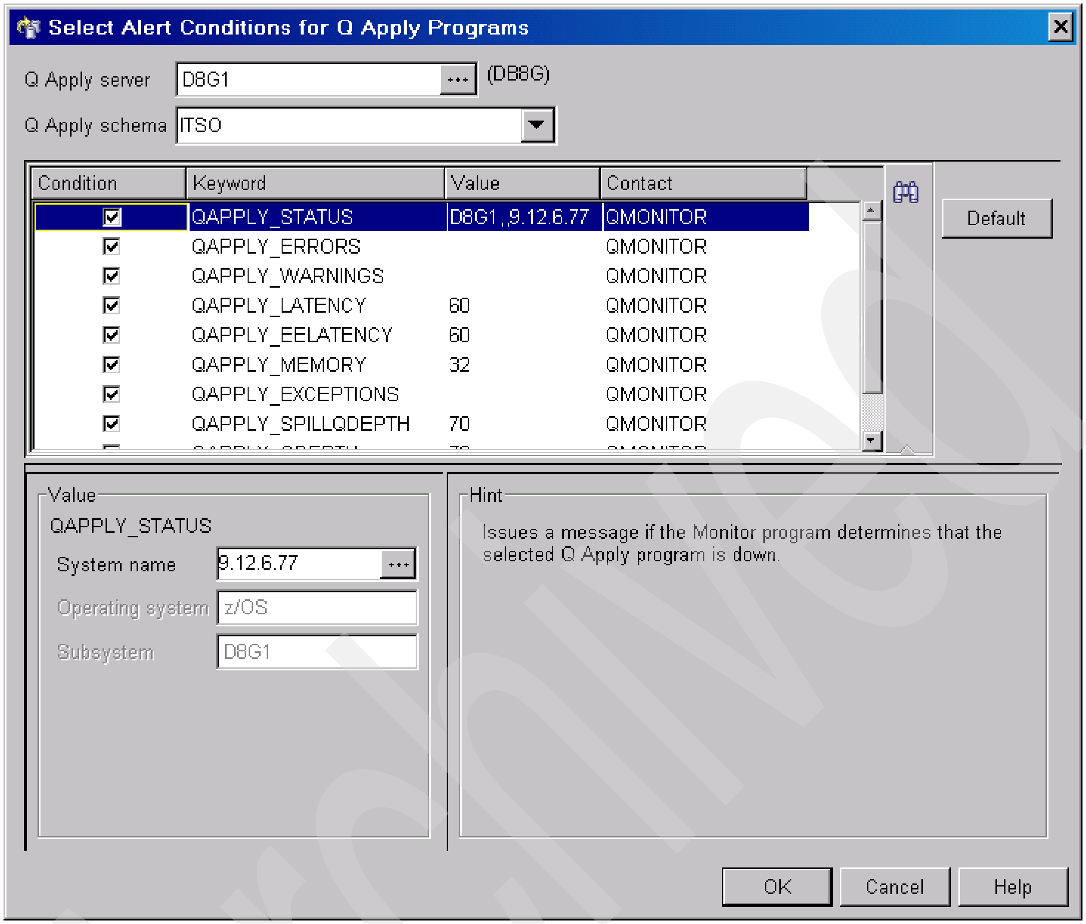


Figure 3-110 Alert Conditions

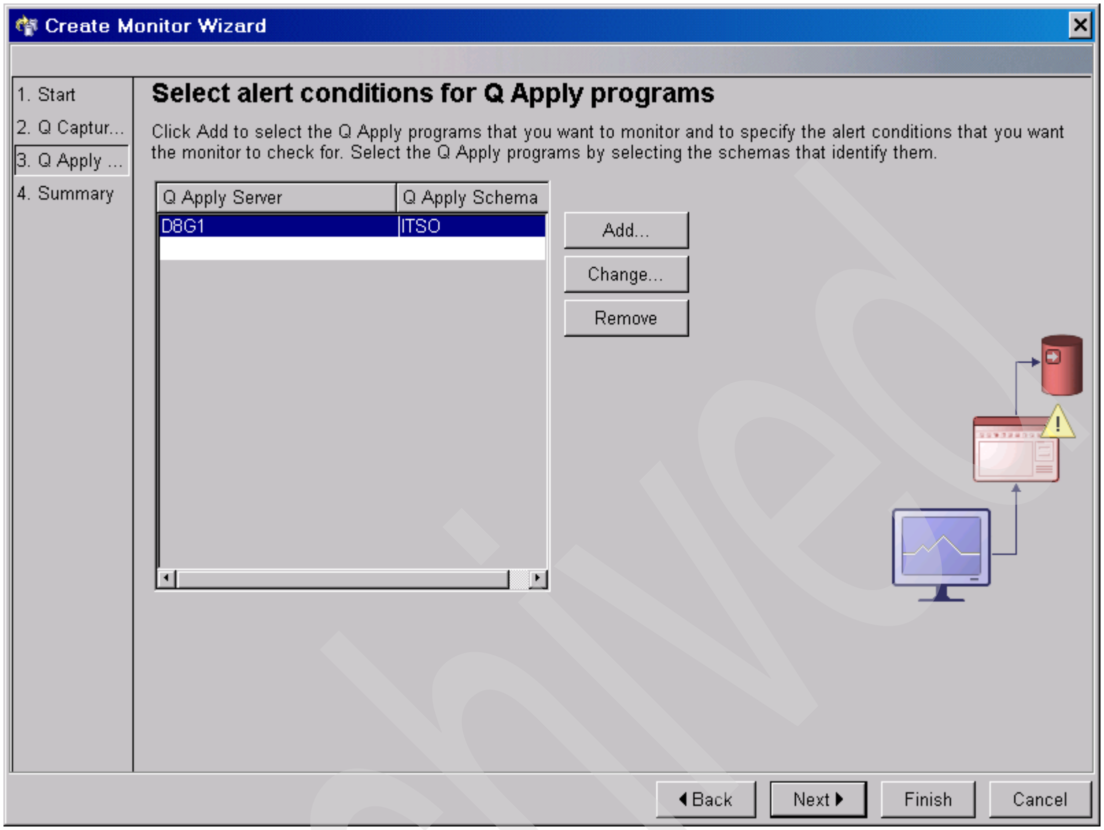


Figure 3-111 Create Monitor Wizard - Select alert conditions for Q Apply programs

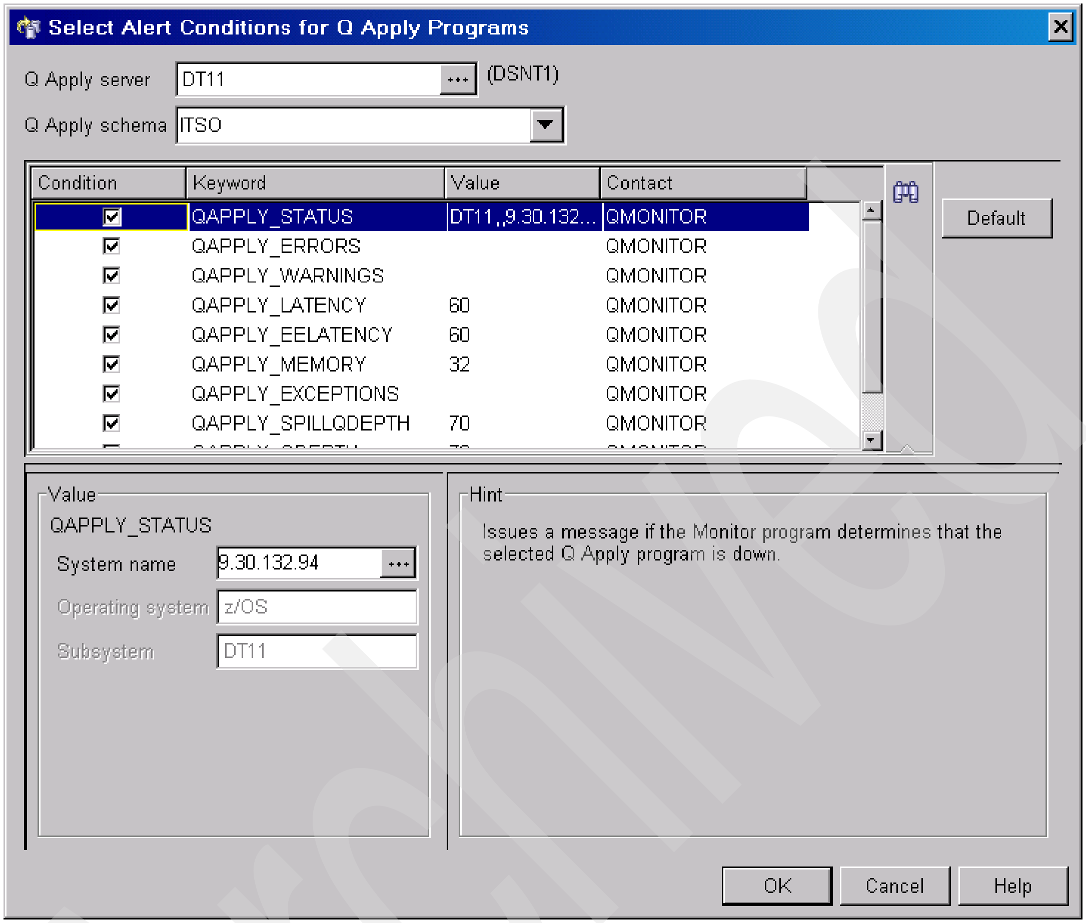


Figure 3-112 Select Alert Conditions for Q Apply Programs

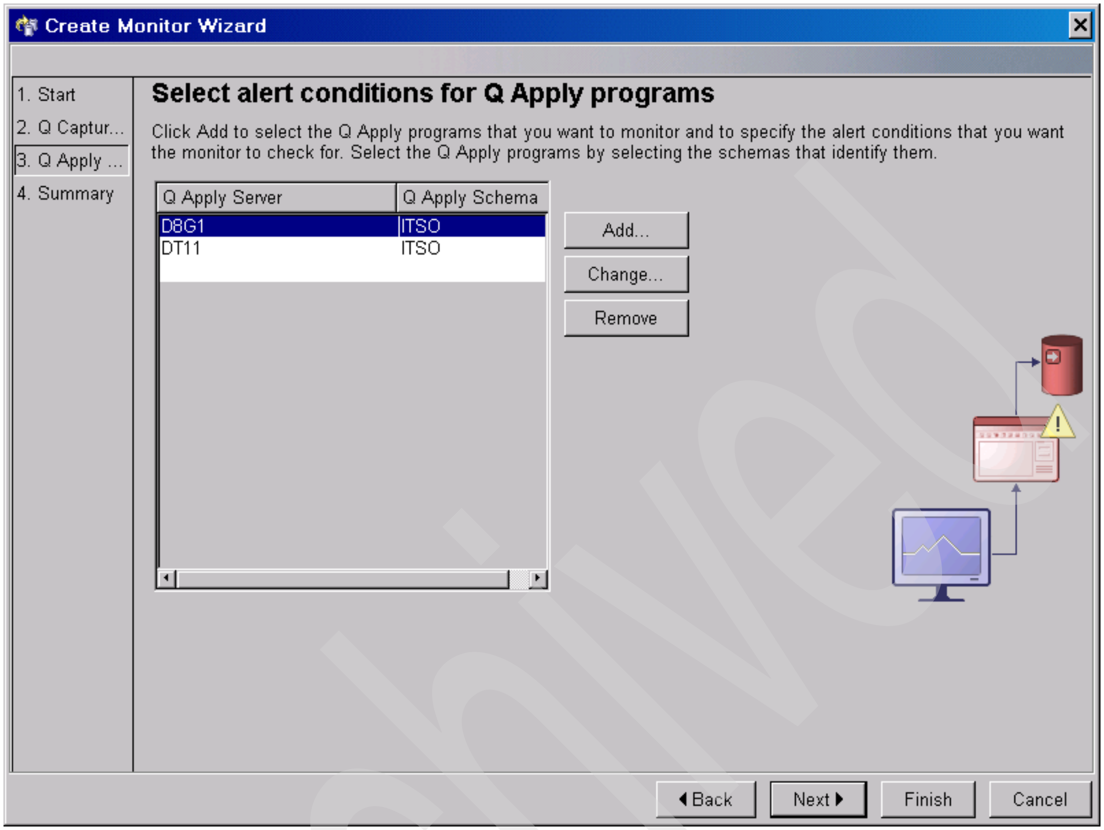


Figure 3-113 Create Monitor Wizard - Select alert conditions for Q Apply programs

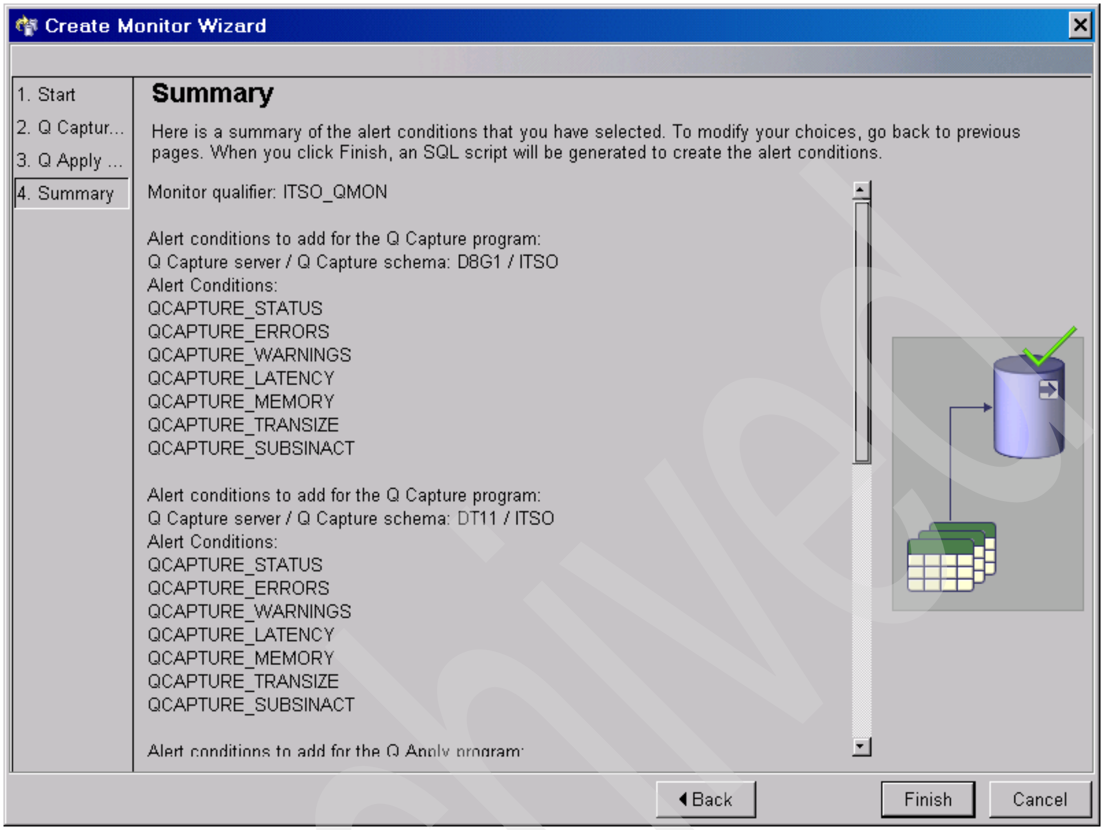


Figure 3-114 Summary

The summary of the alert conditions for the Q Capture and Q Apply programs shown in Figure 3-114 should be reviewed for correctness. Click the **Back** button to correct any errors, or click **Finish** to proceed to generating the script for defining the various alert condition objects specified, as shown in Figure 3-115.

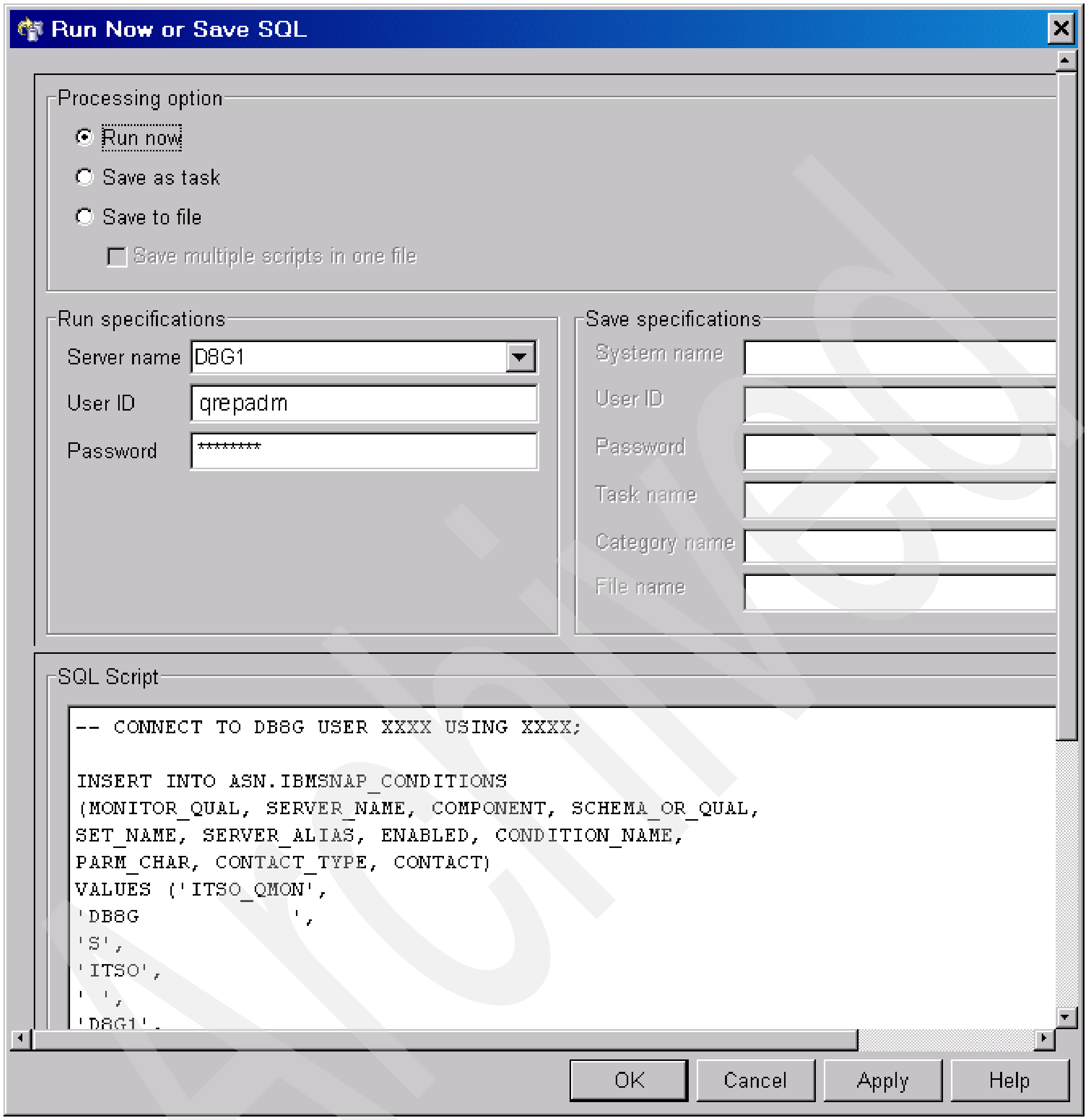


Figure 3-115 Run Now or Save SQL

Select the **Run now** option and click **OK** to display the message that the scripts executed successfully, as shown in Figure 3-116.

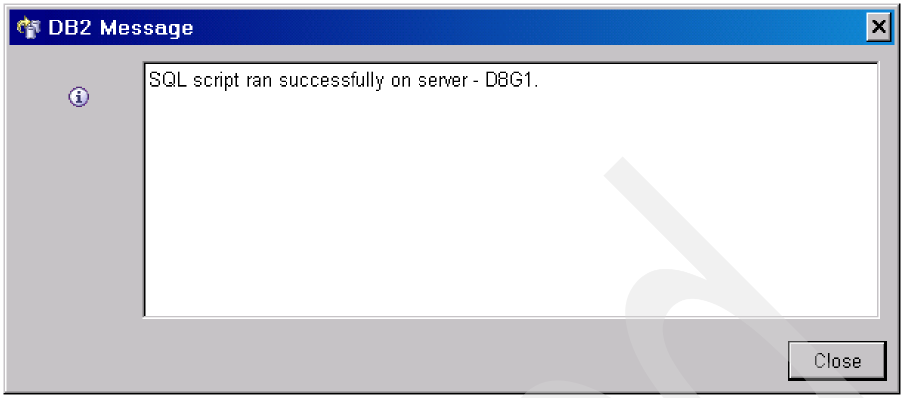


Figure 3-116 SQL script ran successfully message

The monitor definitions have been created.

Step 7d: Start the Alert Monitor

This alert monitor has been defined to check on the replication status on both servers DT11 and D8G1 from a single server D8G1. The alert monitor will run only on D8G1.

As mentioned earlier, DAS was *not* installed, and in many MVS environments the control of tasks tends to remain within the security of the MVS operating system itself. The alert monitor can run either as a batch program or a started task. Most customers will opt for the started task as a result, but the example shown in Example 3-52 uses a batch address space.

> **Note:** The sample in SASNSAMP member ASNSTRM was modified to run the monitor.

Example 3-52 Starting the Alert Monitor

```
//QMONITSO JOB NOTIFY=&SYSUID,
//         MSGCLASS=H,MSGLEVEL=(1,1),
//         REGION=OM,TIME=NOLIMIT
/*JOBPARM S=SC53
//****************************************************************/
//*                                                              */
//*    IBM DB2 Information Integrator Replication Version 8.2    */
//*    for z/OS (5655-I60)                                       */
//*                                                              */
//*    Sample job to run the Replication alert monitor           */
//*                                                              */
//*    monitor_server is a mandatory parameter.                  */
```

```
//*    monitor_qual is a mandatory parameter.                       */
//*                                                                 */
//*    Locate and change all occurrences of the following           */
//*    strings                                                      */
//*    (a) The subsystem name DSN!! to the name of your DB2         */
//*    (b) ASNQ!!0 to the name of your WebSphere II Replication           */
//*        target library                                           */
//*                                                                 */
//*****************************************************************/
//*##############################################################
//*##The variables in the commands are shown unbold, and
//*##these values are substituted from the templates described
//*##in Table 3-4 on page 89
//*##############################################################
//* DB2 subsystem name D8G1 M.3 in Table 3-4 on page 89
//* Monitor qualifier ITSO_QMON M.5f in Table 3-4 on page 89
//ASNMON   EXEC PGM=ASNMON,REGION=0M,
// PARM='monitor_server=D8G1 monitor_qual=ITSO_QMON'
//*PARM='monitor_server=D8G1 monitor_qual=ITSO_QMON runonce=y'
//STEPLIB  DD DSN=ASN.V8R2M0.SASNLOAD,DISP=SHR
//         DD DSN=DB8G8.SDSNLOAD,DISP=SHR
//MSGS     DD   PATH='/usr/lpp/db2repl_08_02/msg/En_US/db2asn.cat'
//CEEDUMP  DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSTERM  DD DUMMY
//*
```

Step 7e: View the alerts

The alert monitor information for z/OS can be viewed from the Replication Center, and modifications can be made to the parameters via the Replication Center shown in Figure 3-117.

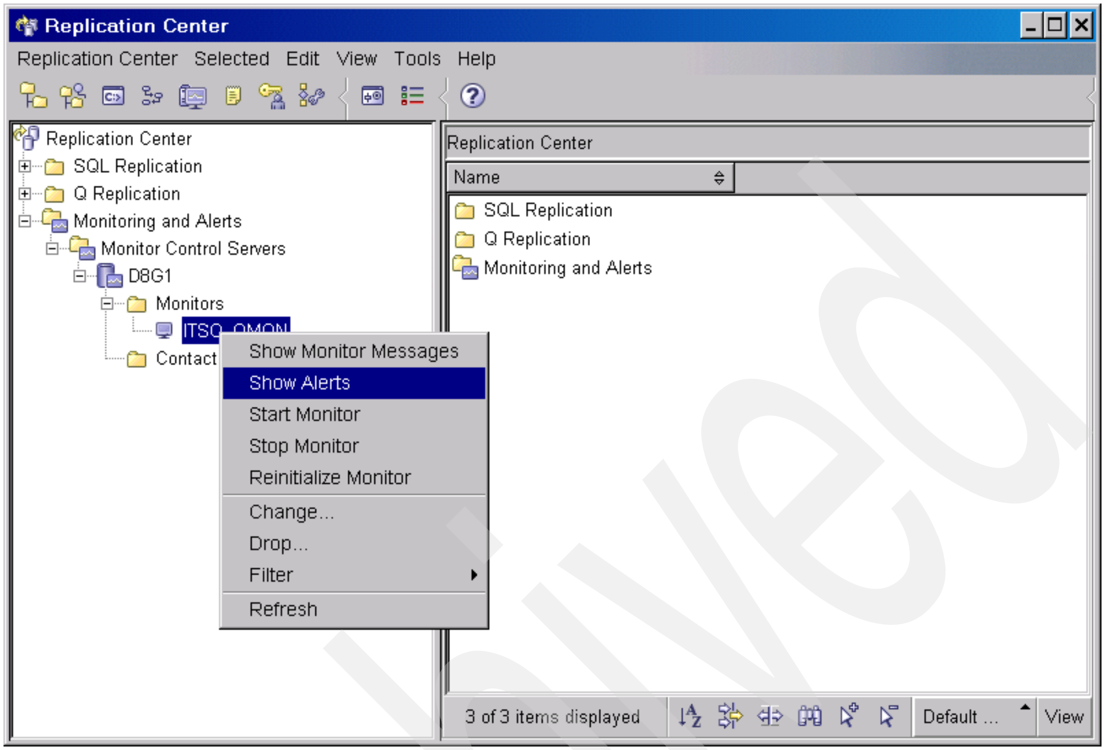


Figure 3-117 Replication Center - Show Alerts

In Figure 3-117, expand the **Monitoring and Alerts** folder, right-click the **ITSO_QMON** monitor name, and select **Show Alerts** to view Figure 3-118.

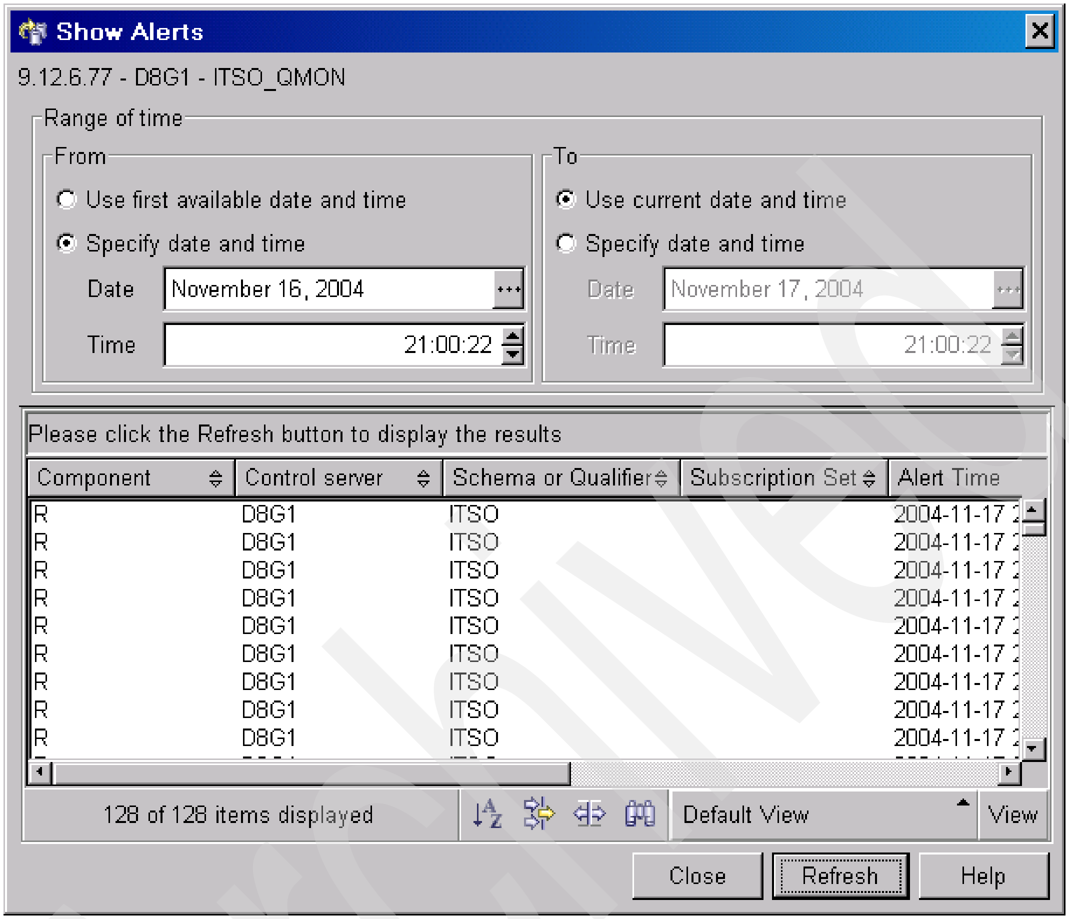


Figure 3-118 Show Alerts - Filter criteria

Figure 3-118 provides a filtering capability to limit the number of alerts to be viewed. After specifying the filter criteria, click the **Refresh** button to view the alerts recorded in the IBMSNAP_ALERTS table. Click **Close** to return to the Replication Center menu, as shown in Figure 3-119.

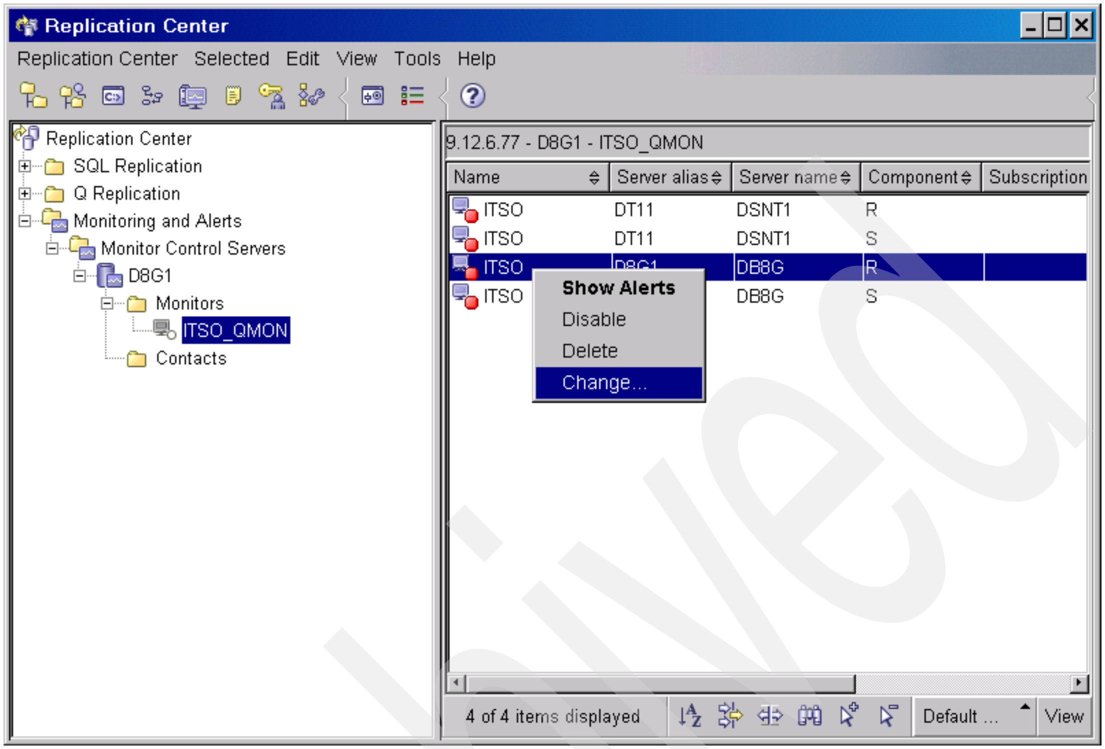


Figure 3-119 Show Alerts

To modify the alert criteria, click the monitor name **ITSO_QMON** from the expanded Replication Center menu (Figure 3-119) to display the servers being monitored in the right pane. Right-click the server whose parameters require changing, and select **Change...** to view Figure 3-120.

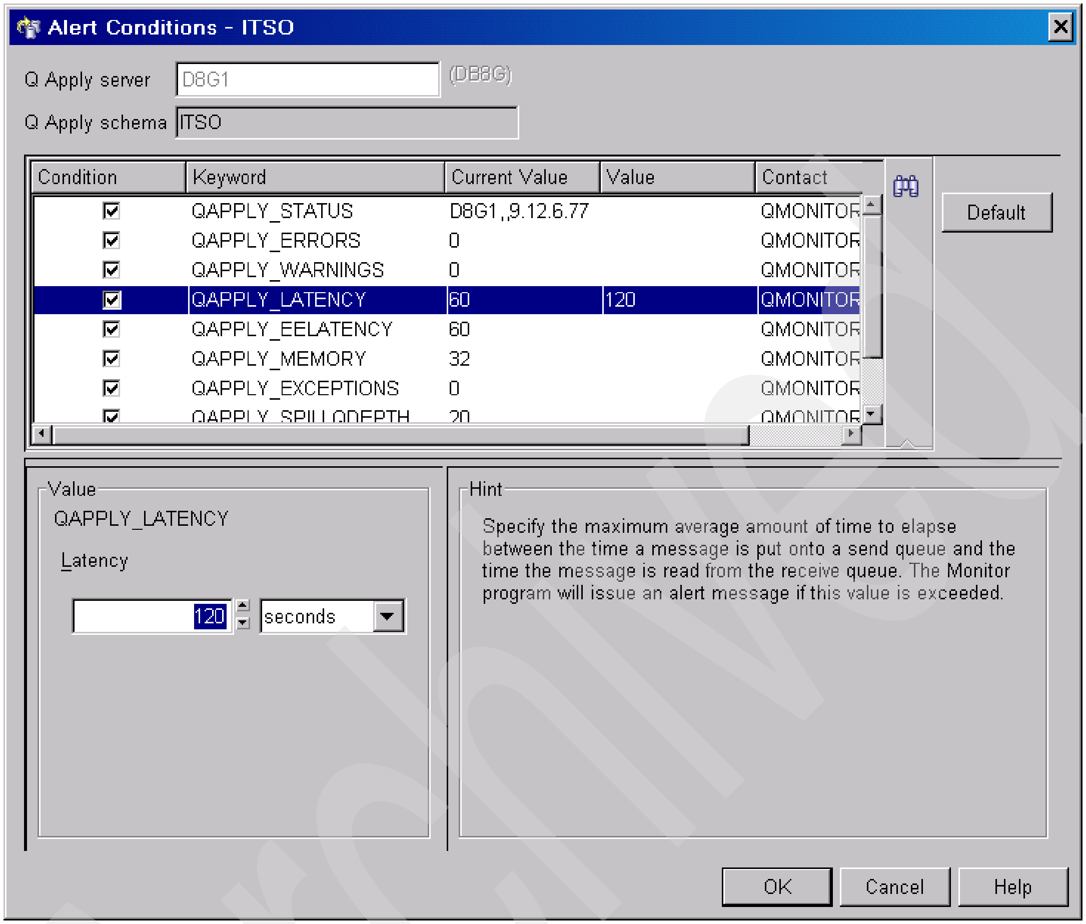


Figure 3-120 Alert Conditions

In Figure 3-120, highlight the alert condition to be modified, and set the new value in the Value field in the bottom left portion of the screen and click **OK**. To remove or add and alert, check or remove the check mark to the left of the alert condition and click **OK** to generate the SQL script, as shown in Figure 3-121.

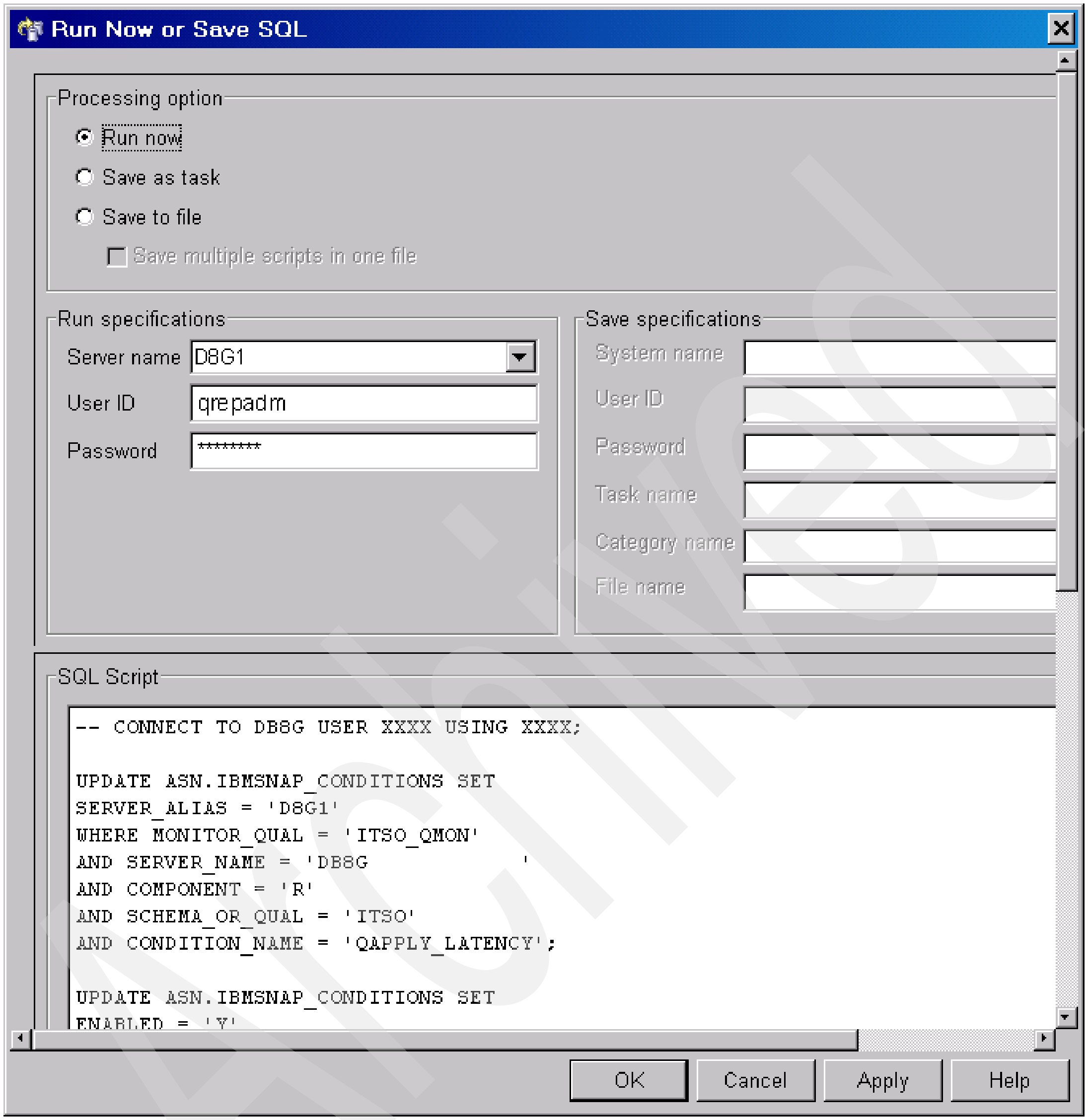


Figure 3-121 Run Now or Save SQL

Select **Run Now** and click **OK** to execute the generated script immediately. Figure 3-122 displays the successful execution of the generated script on the D8G1 database server.

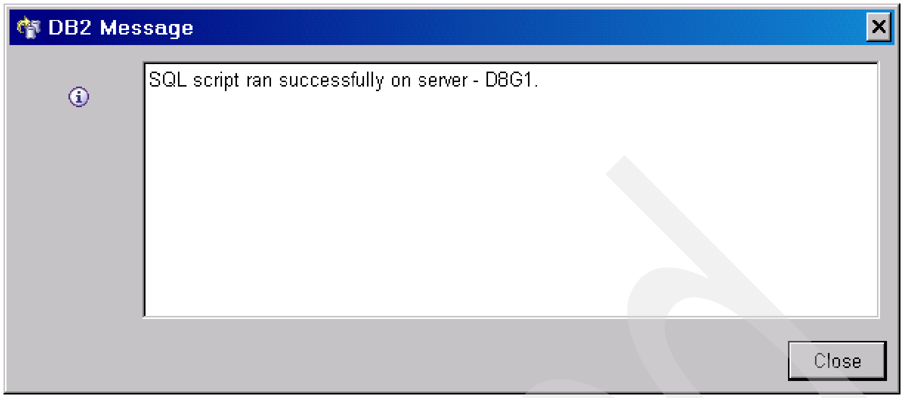


Figure 3-122 SQL script ran successfully message

The alert conditions have been changed and will take effect when the alert monitor is re-initialized either via the Replication Center (if DAS has been installed), or by running the USS command job **asnmcmd**.

> **Note:** Several options for communicating with Q replication programs running on the z/OS platform are discussed in Appendix G, "Communicating with Q replication programs in z/OS" on page 873.

3.5.9 Step 7: Set up monitoring environment using commands

WebSphere II Replication Alert Monitor configuration may be performed using GUI, as described in 3.5.8, "Step 7: Set up monitoring environment using GUI" on page 230, or by using ASNCLP commands to generate SQL scripts.

As mentioned earlier, ASNCLP does not run directly in a z/OS environment, therefore the SQL scripts cannot be executed from ASNCLP as they are generated. ASNCLP may be executed on UNIX or Windows to generate SQL that can then be ported to z/OS and executed under SPUFI or DSNTEP2.

> **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 on the workstation, the user environment must be configured to provide support for Java, as shown in Example 3-37 on page 193.

To ensure that the environment has been set up properly, run the **asnclp** command. If the `>Repl` command line appears, the setup was successful. Enter `quit` to exit the Repl> command line, as shown in Example 3-38.

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 3-39.

The ASNCLP commands in the following sections are categorized according to the following tasks:

The steps involved are mostly the same as with the GUI approach, as follows:

1. Create the Replication Alert Monitor control tables.
2. Create the Replication Alert Monitor contact.
3. Create the Replication Alert Monitor alert conditions.

These steps are described in detail in the following subsections.

Create the Replication Alert Monitor Control Tables in D8G1

Example 3-53 lists the ASNCLP commands for creating the Replication Alert Monitor Control Tables on the D8G1 server. These commands should be saved in a file that is then executed using the command described in Example 3-39 on page 194.

Example 3-53 Create Replication Alert Monitor Control Tables on D8G1

```
#############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 3-1 on page 85 and Table 3-4 on page 89
############################################################
##D8G1 from reference M.3 in Table 3-4 on page 89
##QREPADM from reference M.5c in Table 3-4 on page 89
##QREPITSO from reference B.10 in Table 3-1 on page 85
#############################################################

SET SERVER MONITOR TO DB D8G1 ID QREPADM PASSWORD "xxxxxxxx" ;
SET RUN SCRIPT LATER ;

SET PROFILE MYPROF FOR OBJECT PAGE LOCK TABLESPACE OPTIONS ZOS DB QREPITSO
BUFFERPOOL BP2;

SET PROFILE MYPROF2 FOR OBJECT ROW LOCK TABLESPACE OPTIONS ZOS DB QREPITSO
BUFFERPOOL BP2;
```

```
CREATE CONTROL TABLES FOR MONITOR CONTROL SERVER IN ZOS PAGE LOCK DB "QREPITSO"
NAMING PREFIX QM CREATE USING PROFILE MYPROF ROW LOCK DB "QREPITSO" NAMING
PREFIX QM CREATE USING PROFILE MYPROF2;
```

The generated SQL[3] shown in Example 3-54 can be ported to z/OS for execution under SPUFI or DSNTEP2.

Example 3-54 Generated script for creating monitor control tables on D8G1

```
CONNECT TO D8G1 USER XXXX USING XXXX;
CREATE  TABLESPACE TSMROW1
 IN QREPITSO
 SEGSIZE 4
 LOCKSIZE ROW
 CCSID EBCDIC;
CREATE TABLE ASN.IBMSNAP_CONTACTS(
CONTACT_NAME                    VARCHAR(127) NOT NULL,
EMAIL_ADDRESS                   VARCHAR(128) NOT NULL,
ADDRESS_TYPE                    CHAR(1) NOT NULL,
DELEGATE                        VARCHAR(127),
DELEGATE_START                  DATE,
DELEGATE_END                    DATE,
DESCRIPTION                     VARCHAR(1024))
IN QREPITSO.TSMROW1;
CREATE TYPE 2 UNIQUE INDEX ASN.IBMSNAP_CONTACTSX
ON ASN.IBMSNAP_CONTACTS(
CONTACT_NAME                    ASC);
CREATE  TABLESPACE TSMROW2
 IN QREPITSO
 SEGSIZE 4
 LOCKSIZE ROW
 CCSID EBCDIC;
CREATE TABLE ASN.IBMSNAP_ALERTS(
MONITOR_QUAL                    CHAR(18) NOT NULL,
ALERT_TIME                      TIMESTAMP NOT NULL,
COMPONENT                       CHAR( 1) NOT NULL,
SERVER_NAME                     CHAR(18) NOT NULL,
SERVER_ALIAS                    CHAR( 8),
SCHEMA_OR_QUAL                  VARCHAR(128) NOT NULL,
SET_NAME                        CHAR(18) NOT NULL WITH DEFAULT ' ',
CONDITION_NAME                  CHAR(18) NOT NULL,
OCCURRED_TIME                   TIMESTAMP NOT NULL,
ALERT_COUNTER                   SMALLINT NOT NULL,
ALERT_CODE                      CHAR( 10) NOT NULL,
```

[3] When ASNCLP specifies SET RUN SCRIPT LATER, the SQL that is generated by ASNCLP will be stored in a file on the workstation, in the same directory that ASNCLP was executed from. It will have the name of the DB from the SET SERVER Statement with an .sql extension, for example, D8G1.sql. If the ASNCLP generates sql for two servers there will be two files generated.

```
RETURN_CODE                      INT NOT NULL,
NOTIFICATION_SENT                CHAR(1) NOT NULL,
ALERT_MESSAGE                    VARCHAR(1024) NOT NULL)
IN QREPITSO.TSMROW2;
CREATE TYPE 2 INDEX ASN.IBMSNAP_ALERTX
ON ASN.IBMSNAP_ALERTS(
MONITOR_QUAL                     ASC,
COMPONENT                        ASC,
SERVER_NAME                      ASC,
SCHEMA_OR_QUAL                   ASC,
SET_NAME                         ASC,
CONDITION_NAME                   ASC,
ALERT_CODE                       ASC);
CREATE  TABLESPACE TSMPAGE
 IN QREPITSO
 SEGSIZE 4
 LOCKSIZE PAGE
 CCSID EBCDIC;
CREATE TABLE ASN.IBMSNAP_MONPARMS(
MONITOR_QUAL                     CHAR( 18) NOT NULL,
ALERT_PRUNE_LIMIT                INT WITH DEFAULT 10080,
AUTOPRUNE                        CHAR(  1) WITH DEFAULT 'Y',
EMAIL_SERVER                     VARCHAR(128),
LOGREUSE                         CHAR(  1) WITH DEFAULT 'N',
LOGSTDOUT                        CHAR(  1) WITH DEFAULT 'N',
NOTIF_PER_ALERT                  INT WITH DEFAULT 3,
NOTIF_MINUTES                    INT WITH DEFAULT 60,
MONITOR_ERRORS                   VARCHAR(128),
MONITOR_INTERVAL                 INT WITH DEFAULT 300,
MONITOR_PATH                     VARCHAR(1040),
RUNONCE                          CHAR(  1) WITH DEFAULT 'N',
TERM                             CHAR(  1) WITH DEFAULT 'N',
TRACE_LIMIT                      INT WITH DEFAULT 10080,
ARCH_LEVEL                       CHAR(  4) WITH DEFAULT '0810')
IN QREPITSO.TSMPAGE;
CREATE TYPE 2 UNIQUE INDEX ASN.IBMSNAP_MONPARMSX
ON ASN.IBMSNAP_MONPARMS(
MONITOR_QUAL                     ASC);
CREATE TABLE ASN.IBMSNAP_GROUPS(
GROUP_NAME                       VARCHAR(127) NOT NULL,
DESCRIPTION                      VARCHAR(1024))
IN QREPITSO.TSMROW1;
CREATE TYPE 2 UNIQUE INDEX ASN.IBMSNAP_GROUPSX
ON ASN.IBMSNAP_GROUPS(
GROUP_NAME                       ASC);
CREATE TABLE ASN.IBMSNAP_CONTACTGRP(
GROUP_NAME                       VARCHAR(127) NOT NULL,
CONTACT_NAME                     VARCHAR(127) NOT NULL)
IN QREPITSO.TSMROW1;
```

```
CREATE TYPE 2 UNIQUE INDEX ASN.IBMSNAP_CONTACTGPX
ON ASN.IBMSNAP_CONTACTGRP(
GROUP_NAME                      ASC,
CONTACT_NAME                    ASC);
CREATE TABLE ASN.IBMSNAP_CONDITIONS(
MONITOR_QUAL                    CHAR(18) NOT NULL,
SERVER_NAME                     CHAR(18) NOT NULL,
COMPONENT                       CHAR( 1) NOT NULL,
SCHEMA_OR_QUAL                  VARCHAR(128) NOT NULL,
SET_NAME                        CHAR(18) NOT NULL WITH DEFAULT ' ',
SERVER_ALIAS                    CHAR( 8),
ENABLED                         CHAR( 1) NOT NULL,
CONDITION_NAME                  CHAR(18) NOT NULL,
PARM_INT                        INT,
PARM_CHAR                       VARCHAR(128),
CONTACT_TYPE                    CHAR( 1) NOT NULL,
CONTACT                         VARCHAR(127) NOT NULL)
IN QREPITSO.TSMROW1;
CREATE TYPE 2 UNIQUE INDEX ASN.IBMSNAP_MONCONDX
ON ASN.IBMSNAP_CONDITIONS(
MONITOR_QUAL                    ASC,
SERVER_NAME                     ASC,
COMPONENT                       ASC,
SCHEMA_OR_QUAL                  ASC,
SET_NAME                        ASC,
CONDITION_NAME                  ASC);
CREATE TABLE ASN.IBMSNAP_MONSERVERS(
MONITOR_QUAL                    CHAR(18) NOT NULL,
SERVER_NAME                     CHAR(18) NOT NULL,
SERVER_ALIAS                    CHAR( 8),
LAST_MONITOR_TIME               TIMESTAMP NOT NULL,
START_MONITOR_TIME              TIMESTAMP,
END_MONITOR_TIME                TIMESTAMP,
LASTRUN                         TIMESTAMP NOT NULL,
LASTSUCCESS                     TIMESTAMP,
STATUS                          SMALLINT NOT NULL)
IN QREPITSO.TSMROW1;
CREATE TYPE 2 UNIQUE INDEX ASN.IBMSNAP_MONSERVERX
ON ASN.IBMSNAP_MONSERVERS(
MONITOR_QUAL                    ASC,
SERVER_NAME                     ASC);
CREATE TABLE ASN.IBMSNAP_MONENQ(
MONITOR_QUAL                    CHAR( 18) NOT NULL)
IN QREPITSO.TSMROW1;
CREATE TABLE ASN.IBMSNAP_MONTRACE(
MONITOR_QUAL                    CHAR(18) NOT NULL,
TRACE_TIME                      TIMESTAMP NOT NULL,
OPERATION                       CHAR( 8) NOT NULL,
DESCRIPTION                     VARCHAR(1024) NOT NULL)
```

```
IN QREPITSO.TSMPAGE;
CREATE TYPE 2 INDEX ASN.IBMSNAP_MONTRACEX
ON ASN.IBMSNAP_MONTRACE(
MONITOR_QUAL                    ASC,
TRACE_TIME                      ASC);
CREATE TABLE ASN.IBMSNAP_MONTRAIL(
MONITOR_QUAL                    CHAR(18) NOT NULL,
SERVER_NAME                     CHAR(18) NOT NULL,
SERVER_ALIAS                    CHAR( 8),
STATUS                          SMALLINT NOT NULL,
LASTRUN                         TIMESTAMP NOT NULL,
LASTSUCCESS                     TIMESTAMP,
ENDTIME                         TIMESTAMP NOT NULL WITH DEFAULT ,
LAST_MONITOR_TIME               TIMESTAMP NOT NULL,
START_MONITOR_TIME              TIMESTAMP,
END_MONITOR_TIME                TIMESTAMP,
SQLCODE                         INT,
SQLSTATE                        CHAR(5),
NUM_ALERTS                      INT NOT NULL,
NUM_NOTIFICATIONS               INT NOT NULL)
IN QREPITSO.TSMPAGE;
COMMIT;
```

Attention: The generated script *must be executed before* the next step of creating the contact lists.

Create the Replication Alert Monitor Contact

Example 3-55 shows the ASNCLP commands for creating contact information.

Example 3-55 Create replication alert monitor contact

```
#############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 3-4 on page 89
############################################################
##D8G1 from reference M.3 in Table 3-4 on page 89
##QREPADM from reference M.5c in Table 3-4 on page 89
##QMONITOR from reference M.5d in Table 3-4 on page 89
##QREPADM@us.ibm.com from reference M.5e in Table 3-4 on page 89
#############################################################
SET SERVER MONITOR TO DB D8G1 ID QREPADM PASSWORD "xxxxxxxx" ;
SET RUN SCRIPT LATER;

CREATE CONTACT "QMONITOR" EMAIL "qrepadm@us.ibm.com";
```

The generated SQL shown in Example 3-56 can be ported to z/OS for execution under SPUFI or DSNTEP2.

Example 3-56 Generated SQL for creating contact list on D8G1

```
CONNECT TO DB8G USER XXXX USING XXXX;
INSERT INTO ASN.IBMSNAP_CONTACTS (
CONTACT_NAME,
EMAIL_ADDRESS,
ADDRESS_TYPE
)
VALUES (
'QMONITOR',
'qrepadm@us.ibm.com',
'E'
);
COMMIT;
```

Attention: The generated script *must be executed before* the next step of creating the alert conditions.

Create Replication Alert Monitor alert conditions

Example 3-57 shows the ASNCLP commands for creating alert conditions for Q Capture and Q Apply programs on the DT11 and D8G1 servers.

Example 3-57 Create Replication Alert Monitor alert conditions

```
#############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 3-1 on page 85, Table 3-3 on page 88 and Table 3-4 on page 89
############################################################
##D8G1 from reference M.3 in Table 3-4 on page 89
##QREPADM from reference M.5c in Table 3-4 on page 89
##QMONITOR from reference M.5d in Table 3-4 on page 89
##QREPADM from reference A.7a in Table 3-1 on page 85
##QREPADM from reference B.7a in Table 3-1 on page 85
##Capture schema ITSO from reference B.22 in Table 3-3 on page 88
##ITSO_QMON from reference M.5f in Table 3-4 on page 89
##Apply schema ITSO from reference B.23 in Table 3-3 on page 88
#############################################################
ASNCLP SESSION SET TO Q REPLICATION ;

SET SERVER MONITOR TO DB D8G1 ID QREPADM PASSWORD "xxxxxxxx" ;
SET SERVER CAPTURE TO DB D8G1 ID QREPADM PASSWORD "xxxxxxxx" ;
SET SERVER TARGET TO DB D8G1 ID QREPADM PASSWORD "xxxxxxxx" ;

SET RUN SCRIPT LATER;
```

```
CREATE ALERT CONDITIONS FOR QCAPTURE SCHEMA ITSO MONITOR QUALIFIER ITSO_QMON
NOTIFY CONTACT "QMONITOR"(STATUS DOWN, ERRORS, WARNINGS, LATENCY 60, MEMORY 32,
TRANSACTION SIZE 8, SUBSCRIPTIONS INACTIVE) ;

CREATE ALERT CONDITIONS FOR QAPPLY SCHEMA ITSO MONITOR QUALIFIER ITSO_QMON
NOTIFY CONTACT "QMONITOR"(STATUS DOWN, ERRORS, WARNINGS, LATENCY 60, EELATENCY
60, MEMORY 32, EXCEPTIONS, SPILL QUEUES DEPTH 70,QUEUE DEPTH 70) ;

SET SERVER CAPTURE TO DB DT11 ID QREPADM PASSWORD "xxxxxxxx" ;
SET SERVER TARGET TO DB DT11 ID QREPADM PASSWORD "xxxxxxxx" ;

CREATE ALERT CONDITIONS FOR QCAPTURE SCHEMA ITSO MONITOR QUALIFIER ITSO_QMON
NOTIFY CONTACT "QMONITOR"(STATUS DOWN, ERRORS, WARNINGS, LATENCY 60, MEMORY 32,
TRANSACTION SIZE 8, SUBSCRIPTIONS INACTIVE) ;

CREATE ALERT CONDITIONS FOR QAPPLY SCHEMA ITSO MONITOR QUALIFIER ITSO_QMON
NOTIFY CONTACT "QMONITOR"(STATUS DOWN, ERRORS, WARNINGS, LATENCY 60, EELATENCY
60, MEMORY 32, EXCEPTIONS, SPILL QUEUES DEPTH 70,QUEUE DEPTH 70) ;
```

The generated SQL shown in Example 3-58 can be ported to z/OS for execution under SPUFI or DSNTEP2.

Example 3-58 Generated SQL for creating alert conditions

```
CONNECT TO DB8G USER XXXX USING XXXX;
INSERT INTO ASN.IBMSNAP_CONDITIONS
(MONITOR_QUAL, SERVER_NAME, COMPONENT, SCHEMA_OR_QUAL,
SET_NAME, SERVER_ALIAS, ENABLED, CONDITION_NAME,
PARM_CHAR, CONTACT_TYPE, CONTACT)
VALUES ('ITSO_QMON',
'DB8G            ',
'S',
'ITSO',
' ',
'D8G1',
'Y',
'QCAPTURE_STATUS',
'D8G1,,9.12.6.77',
'C',
'QMONITOR');
INSERT INTO ASN.IBMSNAP_CONDITIONS
(MONITOR_QUAL, SERVER_NAME, COMPONENT, SCHEMA_OR_QUAL,
SET_NAME, SERVER_ALIAS, ENABLED, CONDITION_NAME,
CONTACT_TYPE, CONTACT)
VALUES ('ITSO_QMON',
'DB8G            ',
'S',
'ITSO',
```

```
' ',
'D8G1',
'Y',
'QCAPTURE_ERRORS',
'C',
'QMONITOR');
INSERT INTO ASN.IBMSNAP_CONDITIONS
(MONITOR_QUAL, SERVER_NAME, COMPONENT, SCHEMA_OR_QUAL,
SET_NAME, SERVER_ALIAS, ENABLED, CONDITION_NAME,
CONTACT_TYPE, CONTACT)
VALUES ('ITSO_QMON',
'DB8G            ',
'S',
'ITSO',
' ',
'D8G1',
'Y',
'QCAPTURE_WARNINGS',
'C',
'QMONITOR');
INSERT INTO ASN.IBMSNAP_CONDITIONS
(MONITOR_QUAL, SERVER_NAME, COMPONENT, SCHEMA_OR_QUAL,
SET_NAME, SERVER_ALIAS, ENABLED, CONDITION_NAME,
PARM_INT, CONTACT_TYPE, CONTACT)
VALUES ('ITSO_QMON',
'DB8G            ',
'S',
'ITSO',
' ',
'D8G1',
'Y',
'QCAPTURE_LATENCY',
60,
'C',
'QMONITOR');
INSERT INTO ASN.IBMSNAP_CONDITIONS
(MONITOR_QUAL, SERVER_NAME, COMPONENT, SCHEMA_OR_QUAL,
SET_NAME, SERVER_ALIAS, ENABLED, CONDITION_NAME,
PARM_INT, CONTACT_TYPE, CONTACT)
VALUES ('ITSO_QMON',
'DB8G            ',
'S',
'ITSO',
' ',
'D8G1',
'Y',
'QCAPTURE_MEMORY',
32,
'C',
```

```
'QMONITOR');
INSERT INTO ASN.IBMSNAP_CONDITIONS
(MONITOR_QUAL, SERVER_NAME, COMPONENT, SCHEMA_OR_QUAL,
SET_NAME, SERVER_ALIAS, ENABLED, CONDITION_NAME,
PARM_INT, CONTACT_TYPE, CONTACT)
VALUES ('ITSO_QMON',
'DB8G            ',
'S',
'ITSO',
' ',
'D8G1',
'Y',
'QCAPTURE_TRANSIZE',
8,
'C',
'QMONITOR');
INSERT INTO ASN.IBMSNAP_CONDITIONS
(MONITOR_QUAL, SERVER_NAME, COMPONENT, SCHEMA_OR_QUAL,
SET_NAME, SERVER_ALIAS, ENABLED, CONDITION_NAME,
CONTACT_TYPE, CONTACT)
VALUES ('ITSO_QMON',
'DB8G            ',
'S',
'ITSO',
' ',
'D8G1',
'Y',
'QCAPTURE_SUBSINACT',
'C',
'QMONITOR');
INSERT INTO ASN.IBMSNAP_CONDITIONS
(MONITOR_QUAL, SERVER_NAME, COMPONENT, SCHEMA_OR_QUAL,
SET_NAME, SERVER_ALIAS, ENABLED, CONDITION_NAME,
PARM_CHAR, CONTACT_TYPE, CONTACT)
VALUES ('ITSO_QMON',
'DSNT1           ',
'S',
'ITSO',
' ',
'DT11',
'Y',
'QCAPTURE_STATUS',
'DT11,,9.30.132.94',
'C',
'QMONITOR');
INSERT INTO ASN.IBMSNAP_CONDITIONS
(MONITOR_QUAL, SERVER_NAME, COMPONENT, SCHEMA_OR_QUAL,
SET_NAME, SERVER_ALIAS, ENABLED, CONDITION_NAME,
CONTACT_TYPE, CONTACT)
```

```
VALUES ('ITSO_QMON',
'DSNT1           ',
'S',
'ITSO',
' ',
'DT11',
'Y',
'QCAPTURE_ERRORS',
'C',
'QMONITOR');
INSERT INTO ASN.IBMSNAP_CONDITIONS
(MONITOR_QUAL, SERVER_NAME, COMPONENT, SCHEMA_OR_QUAL,
SET_NAME, SERVER_ALIAS, ENABLED, CONDITION_NAME,
CONTACT_TYPE, CONTACT)
VALUES ('ITSO_QMON',
'DSNT1           ',
'S',
'ITSO',
' ',
'DT11',
'Y',
'QCAPTURE_WARNINGS',
'C',
'QMONITOR');
INSERT INTO ASN.IBMSNAP_CONDITIONS
(MONITOR_QUAL, SERVER_NAME, COMPONENT, SCHEMA_OR_QUAL,
SET_NAME, SERVER_ALIAS, ENABLED, CONDITION_NAME,
PARM_INT, CONTACT_TYPE, CONTACT)
VALUES ('ITSO_QMON',
'DSNT1           ',
'S',
'ITSO',
' ',
'DT11',
'Y',
'QCAPTURE_LATENCY',
60,
'C',
'QMONITOR');
INSERT INTO ASN.IBMSNAP_CONDITIONS
(MONITOR_QUAL, SERVER_NAME, COMPONENT, SCHEMA_OR_QUAL,
SET_NAME, SERVER_ALIAS, ENABLED, CONDITION_NAME,
PARM_INT, CONTACT_TYPE, CONTACT)
VALUES ('ITSO_QMON',
'DSNT1           ',
'S',
'ITSO',
' ',
'DT11',
```

```
'Y',
'QCAPTURE_MEMORY',
32,
'C',
'QMONITOR');
INSERT INTO ASN.IBMSNAP_CONDITIONS
(MONITOR_QUAL, SERVER_NAME, COMPONENT, SCHEMA_OR_QUAL,
SET_NAME, SERVER_ALIAS, ENABLED, CONDITION_NAME,
PARM_INT, CONTACT_TYPE, CONTACT)
VALUES ('ITSO_QMON',
'DSNT1           ',
'S',
'ITSO',
' ',
'DT11',
'Y',
'QCAPTURE_TRANSIZE',
8,
'C',
'QMONITOR');
INSERT INTO ASN.IBMSNAP_CONDITIONS
(MONITOR_QUAL, SERVER_NAME, COMPONENT, SCHEMA_OR_QUAL,
SET_NAME, SERVER_ALIAS, ENABLED, CONDITION_NAME,
CONTACT_TYPE, CONTACT)
VALUES ('ITSO_QMON',
'DSNT1           ',
'S',
'ITSO',
' ',
'DT11',
'Y',
'QCAPTURE_SUBSINACT',
'C',
'QMONITOR');
COMMIT;
CONNECT TO DB8G USER XXXX USING XXXX;
INSERT INTO ASN.IBMSNAP_CONDITIONS
(MONITOR_QUAL, SERVER_NAME, COMPONENT, SCHEMA_OR_QUAL,
SET_NAME, SERVER_ALIAS, ENABLED, CONDITION_NAME,
PARM_CHAR, CONTACT_TYPE, CONTACT)
VALUES ('ITSO_QMON',
'DB8G            ',
'R',
'ITSO',
' ',
'D8G1',
'Y',
'QAPPLY_STATUS',
'D8G1,,9.12.6.77',
```

```
'C',
'QMONITOR');
INSERT INTO ASN.IBMSNAP_CONDITIONS
(MONITOR_QUAL, SERVER_NAME, COMPONENT, SCHEMA_OR_QUAL,
SET_NAME, SERVER_ALIAS, ENABLED, CONDITION_NAME,
CONTACT_TYPE, CONTACT)
VALUES ('ITSO_QMON',
'DB8G            ',
'R',
'ITSO',
' ',
'D8G1',
'Y',
'QAPPLY_ERRORS',
'C',
'QMONITOR');
INSERT INTO ASN.IBMSNAP_CONDITIONS
(MONITOR_QUAL, SERVER_NAME, COMPONENT, SCHEMA_OR_QUAL,
SET_NAME, SERVER_ALIAS, ENABLED, CONDITION_NAME,
CONTACT_TYPE, CONTACT)
VALUES ('ITSO_QMON',
'DB8G            ',
'R',
'ITSO',
' ',
'D8G1',
'Y',
'QAPPLY_WARNINGS',
'C',
'QMONITOR');
INSERT INTO ASN.IBMSNAP_CONDITIONS
(MONITOR_QUAL, SERVER_NAME, COMPONENT, SCHEMA_OR_QUAL,
SET_NAME, SERVER_ALIAS, ENABLED, CONDITION_NAME,
PARM_INT, CONTACT_TYPE, CONTACT)
VALUES ('ITSO_QMON',
'DB8G            ',
'R',
'ITSO',
' ',
'D8G1',
'Y',
'QAPPLY_LATENCY',
60,
'C',
'QMONITOR');
INSERT INTO ASN.IBMSNAP_CONDITIONS
(MONITOR_QUAL, SERVER_NAME, COMPONENT, SCHEMA_OR_QUAL,
SET_NAME, SERVER_ALIAS, ENABLED, CONDITION_NAME,
PARM_INT, CONTACT_TYPE, CONTACT)
```

```
VALUES ('ITSO_QMON',
'DB8G            ',
'R',
'ITSO',
' ',
'D8G1',
'Y',
'QAPPLY_EELATENCY',
60,
'C',
'QMONITOR');
INSERT INTO ASN.IBMSNAP_CONDITIONS
(MONITOR_QUAL, SERVER_NAME, COMPONENT, SCHEMA_OR_QUAL,
SET_NAME, SERVER_ALIAS, ENABLED, CONDITION_NAME,
PARM_INT, CONTACT_TYPE, CONTACT)
VALUES ('ITSO_QMON',
'DB8G            ',
'R',
'ITSO',
' ',
'D8G1',
'Y',
'QAPPLY_MEMORY',
32,
'C',
'QMONITOR');
INSERT INTO ASN.IBMSNAP_CONDITIONS
(MONITOR_QUAL, SERVER_NAME, COMPONENT, SCHEMA_OR_QUAL,
SET_NAME, SERVER_ALIAS, ENABLED, CONDITION_NAME,
CONTACT_TYPE, CONTACT)
VALUES ('ITSO_QMON',
'DB8G            ',
'R',
'ITSO',
' ',
'D8G1',
'Y',
'QAPPLY_EXCEPTIONS',
'C',
'QMONITOR');
INSERT INTO ASN.IBMSNAP_CONDITIONS
(MONITOR_QUAL, SERVER_NAME, COMPONENT, SCHEMA_OR_QUAL,
SET_NAME, SERVER_ALIAS, ENABLED, CONDITION_NAME,
PARM_INT, CONTACT_TYPE, CONTACT)
VALUES ('ITSO_QMON',
'DB8G            ',
'R',
'ITSO',
' ',
```

```
'D8G1',
'Y',
'QAPPLY_SPILLQDEPTH',
70,
'C',
'QMONITOR');
INSERT INTO ASN.IBMSNAP_CONDITIONS
(MONITOR_QUAL, SERVER_NAME, COMPONENT, SCHEMA_OR_QUAL,
SET_NAME, SERVER_ALIAS, ENABLED, CONDITION_NAME,
PARM_INT, CONTACT_TYPE, CONTACT)
VALUES ('ITSO_QMON',
'DB8G            ',
'R',
'ITSO',
' ',
'D8G1',
'Y',
'QAPPLY_QDEPTH',
70,
'C',
'QMONITOR');
INSERT INTO ASN.IBMSNAP_CONDITIONS
(MONITOR_QUAL, SERVER_NAME, COMPONENT, SCHEMA_OR_QUAL,
SET_NAME, SERVER_ALIAS, ENABLED, CONDITION_NAME,
PARM_CHAR, CONTACT_TYPE, CONTACT)
VALUES ('ITSO_QMON',
'DSNT1           ',
'R',
'ITSO',
' ',
'DT11',
'Y',
'QAPPLY_STATUS',
'DT11,,9.30.132.94',
'C',
'QMONITOR');
INSERT INTO ASN.IBMSNAP_CONDITIONS
(MONITOR_QUAL, SERVER_NAME, COMPONENT, SCHEMA_OR_QUAL,
SET_NAME, SERVER_ALIAS, ENABLED, CONDITION_NAME,
CONTACT_TYPE, CONTACT)
VALUES ('ITSO_QMON',
'DSNT1           ',
'R',
'ITSO',
' ',
'DT11',
'Y',
'QAPPLY_ERRORS',
'C',
```

```
'QMONITOR');
INSERT INTO ASN.IBMSNAP_CONDITIONS
(MONITOR_QUAL, SERVER_NAME, COMPONENT, SCHEMA_OR_QUAL,
SET_NAME, SERVER_ALIAS, ENABLED, CONDITION_NAME,
CONTACT_TYPE, CONTACT)
VALUES ('ITSO_QMON',
'DSNT1           ',
'R',
'ITSO',
' ',
'DT11',
'Y',
'QAPPLY_WARNINGS',
'C',
'QMONITOR');
INSERT INTO ASN.IBMSNAP_CONDITIONS
(MONITOR_QUAL, SERVER_NAME, COMPONENT, SCHEMA_OR_QUAL,
SET_NAME, SERVER_ALIAS, ENABLED, CONDITION_NAME,
PARM_INT, CONTACT_TYPE, CONTACT)
VALUES ('ITSO_QMON',
'DSNT1           ',
'R',
'ITSO',
' ',
'DT11',
'Y',
'QAPPLY_LATENCY',
60,
'C',
'QMONITOR');
INSERT INTO ASN.IBMSNAP_CONDITIONS
(MONITOR_QUAL, SERVER_NAME, COMPONENT, SCHEMA_OR_QUAL,
SET_NAME, SERVER_ALIAS, ENABLED, CONDITION_NAME,
PARM_INT, CONTACT_TYPE, CONTACT)
VALUES ('ITSO_QMON',
'DSNT1           ',
'R',
'ITSO',
' ',
'DT11',
'Y',
'QAPPLY_EELATENCY',
60,
'C',
'QMONITOR');
INSERT INTO ASN.IBMSNAP_CONDITIONS
(MONITOR_QUAL, SERVER_NAME, COMPONENT, SCHEMA_OR_QUAL,
SET_NAME, SERVER_ALIAS, ENABLED, CONDITION_NAME,
PARM_INT, CONTACT_TYPE, CONTACT)
```

```
VALUES ('ITSO_QMON',
'DSNT1           ',
'R',
'ITSO',
' ',
'DT11',
'Y',
'QAPPLY_MEMORY',
32,
'C',
'QMONITOR');
INSERT INTO ASN.IBMSNAP_CONDITIONS
(MONITOR_QUAL, SERVER_NAME, COMPONENT, SCHEMA_OR_QUAL,
SET_NAME, SERVER_ALIAS, ENABLED, CONDITION_NAME,
CONTACT_TYPE, CONTACT)
VALUES ('ITSO_QMON',
'DSNT1           ',
'R',
'ITSO',
' ',
'DT11',
'Y',
'QAPPLY_EXCEPTIONS',
'C',
'QMONITOR');
INSERT INTO ASN.IBMSNAP_CONDITIONS
(MONITOR_QUAL, SERVER_NAME, COMPONENT, SCHEMA_OR_QUAL,
SET_NAME, SERVER_ALIAS, ENABLED, CONDITION_NAME,
PARM_INT, CONTACT_TYPE, CONTACT)
VALUES ('ITSO_QMON',
'DSNT1           ',
'R',
'ITSO',
' ',
'DT11',
'Y',
'QAPPLY_SPILLQDEPTH',
70,
'C',
'QMONITOR');
INSERT INTO ASN.IBMSNAP_CONDITIONS
(MONITOR_QUAL, SERVER_NAME, COMPONENT, SCHEMA_OR_QUAL,
SET_NAME, SERVER_ALIAS, ENABLED, CONDITION_NAME,
PARM_INT, CONTACT_TYPE, CONTACT)
VALUES ('ITSO_QMON',
'DSNT1           ',
'R',
'ITSO',
' ',
```

```
'DT11',
'Y',
'QAPPLY_QDEPTH',
70,
'C',
'QMONITOR');
COMMIT;
```

> **Attention:** The generated script *must be executed before* the next step of activating the alert monitor.

> **Note:** As mentioned earlier, ASNCLP cannot be used to start programs on z/OS. Refer to “Step 7d: Start the Alert Monitor” on page 264 and “Step 7e: View the alerts” on page 265 for a description of the process of starting and verifying the status of the alert monitor program.

3.6 Failover considerations

Ithaca’s Q replication bidirectional topology involves two z/OS servers, STLABD1 and WTSC53, where:

- STLABD1 is the primary server that is available for read and write applications.
- WTSC53 is the non-dedicated secondary server that is available only for read applications against the replica.

The requirement is for the secondary server WTSC53 to take over the update processing of applications in the event of failure of the primary server STLABD1. When the primary server is restored, the requirement is for the secondary server to revert back to read only (switchback). Until the primary server is restored, updates occurring at the secondary server are accumulated to be applied later to the primary server.

> **Attention:** This section provides a very high-level overview of some of the considerations involved with failover processing associated with bidirectional replication. Depending upon the particular environment, the process involved in ensuring satisfactory failover processing can get quite complex. A full discussion of these considerations is beyond the scope of this book.

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 3-123), or messages in the WebSphere MQ queue that had not been transmitted to the secondary server (item 2 in Figure 3-123). These un-replicated changes should be considered to be “data loss” at the secondary server, at least until the primary server is restored. If there are messages in the receive queue on the secondary server that have not been drained (item 4 in Figure 3-123) 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.

Important: Identifying the data loss suffered can be critical for certain types of applications. Approaches for identifying the data loss suffered is beyond the scope of this book.

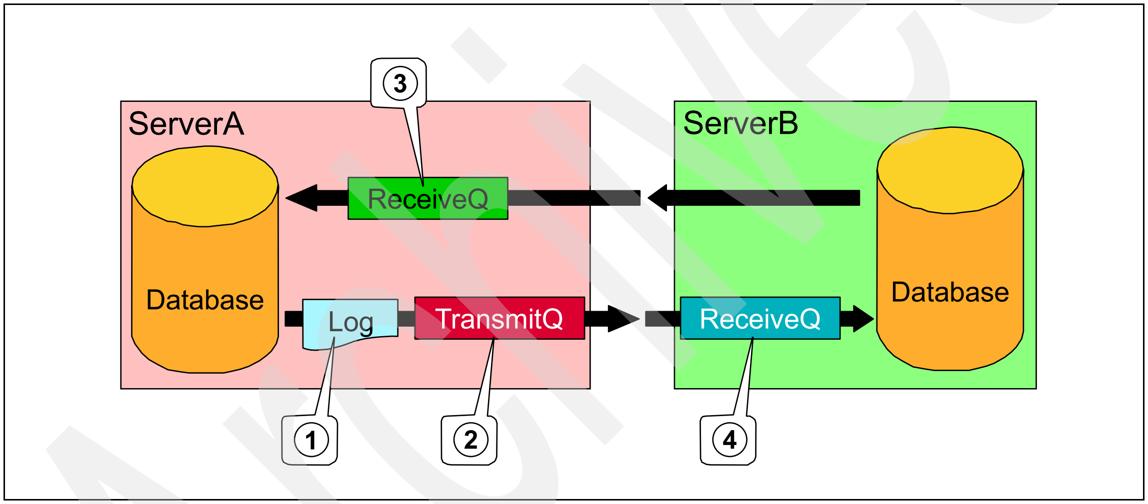


Figure 3-123 Potential sources of conflicts

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 3-123). 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 3-123).

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

To support the Ithaca 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.

Note: The triggering event for failover is assumed to originate external to the Q replication environment, and enable updating applications on the secondary server. Therefore, 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. The current depth of the queue may be determined from the MQ ISPF panels in TSO, or by executing the MVS console command shown in Example 3-59.

Example 3-59 Console command to determine depth of transmit queue on MQZ1

```
=MQZ1 dis ql(MQS1XMIT) curdepth
```

4. 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. Example 3-60 shows an SQL script that can be run on D8G1 to identify the states and load phases of all subscriptions.

Example 3-60 Check the subscriptions states

```
SELECT SUBNAME, STATE, HAS_LOADPHASE FROM ITSO.IBMQREP_SUBS;
```

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 the updates on the DB2 log (after a Q Capture shutdown) 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.

3.7 Switchback 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 Ithaca's bidirectional topology this involves restoring STLABD1 as the primary server with updating applications, and the secondary server reverting back to a read-only environment against the replica. Switchback should involve minimum data loss.

Important: Identifying the data loss suffered can be critical for certain types of applications. Approaches for identifying the data loss suffered are beyond the scope of this book.

Attention: This section provides a very 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. A full discussion of these considerations is beyond the scope of this book.

The effort involved in restoring the normal operating environment depends upon the expected duration of the outage of the primary server.

- Short outage

  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 resynchronize and resume normal operations simply by restarting the failed server, and warm starting Q Capture on the primary server.

- Extended outage

  If the duration of the primary server outage was estimated to be so long that Q Capture was shut down to prevent the queues from filling up, then after restarting the failed server, adequate time must be provided so that the messages already stored in the WebSphere MQ queues can be processed by Q Apply on the primary server. This may be determined by starting the Q Apply proc (as described in Example 3-27 on page 176) after modifying it to add an additional parameter (as shown in Example 3-61). This will cause the Q Apply program to terminate after all receive queues are emptied once. After the messages have been processed, Q Capture can be restarted by starting its proc (as described in Example 3-20 on page 173) after modifying it to add an additional parameter (as shown in Example 3-62). This will cause the Q Capture to stop after reaching the end of the DB2 active log.

Example 3-61 Modify Q Apply proc parameter

```
APARM='autostop=y'
```

Example 3-62 Modify Q Capture proc parameter

```
APARM='autostop=y'
```

  If, despite adequate planning, Q Capture on the secondary server deactivated the subscriptions, then a careful analysis is required to determine the appropriate recovery action, such as:

  - If the Q Subscription was configured for automatic load, then reactivation of the subscription will cause the target table to be emptied and reloaded by the method shown in the subscription, as described in Figure 3-61 on page 167. For a large table this could take significant time and resources. The CAPSTART signal must be issued to the Q Capture program on the surviving side to designate source and target tables.
  - If the Q Subscription was configured for a manual load, then a CAPSTART signal must be issued. Reconfirm that the tables are synchronized (using the TDIFF / TREPAIR utility, or a method of one’s choice) and then a LOADDONE signal is issued to complete subscription activation.

– If appropriate, the existing subscriptions can be dropped and recreated.

Note: It is assumed that the primary server with WebSphere II is restarted successfully to the point of failure.

4

Peer-to-peer Q replication on z/OS platforms

In this chapter we describe a step-by-step approach to implementing a peer-to-peer queue replication financial industry solution on z/OS platforms.

The topics covered are:

- Business requirements
- Rationale for the peer-to-peer solution
- Environment configuration
- Step-by-step setup
- Monitoring considerations
- Failover considerations
- Switchback considerations

4.1 Introduction

Peer-to-peer 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 or more servers where updates are expected to occur at any of the servers as part of normal business activity. Scenarios that may be appropriate for a peer-to-peer replication solution include the following:

- Simultaneous updates to all the replicated tables are expected to occur, with a high potential for the same data to be updated on the different servers concurrently.
- Application users can connect to either server to accomplish their tasks, since all the servers are considered to be peers.

In this chapter we describe a high-availability business solution implementation cycle involving peer-to-peer 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 are documented as follows:

- Business requirements
- Rationale for choosing the peer-to-peer solution
- Environment configuration
- Step-by-step setup
- Failover considerations
- Switchback considerations

4.2 Business requirement

Our fictitious company Carthage is a financial organization that must provide 24x7 access to its customers. Given the critical nature of the business, the implementation must support a high-availability scenario involving two servers. In the event of a failure of either server, the remaining server must take over the workload of the failing site transparently and instantaneously. In addition, for scalability reasons Carthage requires the two servers to act as peers—both for queries as well as updates—and present a single system image to applications. Therefore, in the event of simultaneous updates to the same data on the two servers, the more recent update must prevail.

These requirements may be summarized as follows:

- Provide a single system image to applications accessing the two servers, which means maintaining a peer relationship between the servers.

- When a server fails, there must be no outage perceived by the user, since the the remaining server must take over the workload of the failing server transparently and instantaneously.
- Replicated tables on both servers are peers, and updates may occur at either server—there is no concept of primary or secondary server.
- Update conflicts on the two servers must be resolved in favor of the most recent update.
- Failing server must be resynchronized with the active server after it is restored to service.

4.3 Rationale for the peer-to-peer solution

Table 2-1 on page 39 lists the evaluation criteria for choosing between unidirectional, bidirectional, and peer-to-peer replication topologies.

Since Carthage's business requirement is support for a high-availability peer environment, the choice is a peer-to-peer replication topology.

The peer-to-peer replication topology is appropriate for Carthage for the following reasons:

1. Totally transparent and instantaneous takeover in the event of failure of any server (in seconds).
2. Expected conflicts[1] due to simultaneous updates to be resolved in favor of the most recent update.
3. Adequate computing resources available since either server is meant to pick up the workload of the failing server.

Carthage's requirement for a controlled switchback to the failing server from the active server is handled well by a peer-to-peer replication topology.

4.4 Environment configuration

Figure 4-1 shows the configuration used in the Carthage peer-to-peer replication topology.

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

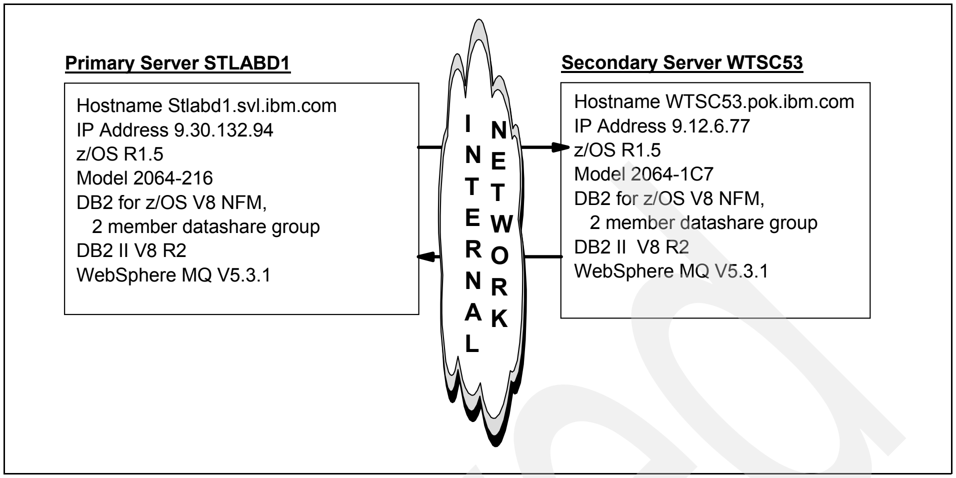


Figure 4-1 Peer-to-peer replication topology configuration for Carthage

We installed a set of 23 tables with referential integrity constraints defined between some of them. Full details of these tables are documented in Appendix F, "Tables used in Q replication scenarios" on page 781.

4.5 Step-by-step setup

In this section we document the step-by-step setup of the peer-to-peer replication topology in our fictitious company. Figure 4-2 lists the main steps involved in setting up the environment. Each of these steps is described in detail in the following subsections.

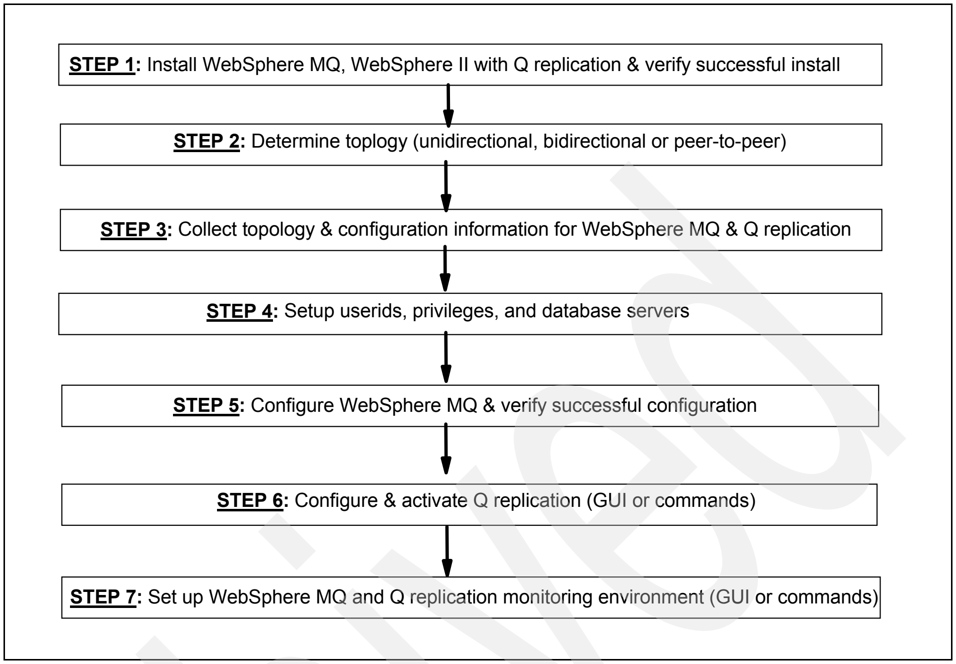


Figure 4-2 Overview of setup steps

4.5.1 Step 1: Install WebSphere MQ, WebSphere II with Q replication

The steps involved are identical to the setup described in 3.5.1, “Step 1: Install WebSphere MQ, WebSphere II with Q replication” on page 77.

4.5.2 Step 2: Determine topology

We chose the peer-to-peer replication topology to address the Carthage business requirement, as described in 4.3, “Rationale for the peer-to-peer solution” on page 297.

4.5.3 Step 3: Collect topology and configuration information

Implementing peer-to-peer replication is a complex task involving effective coordination of the configuration settings of the operating system, database management system, WebSphere MQ, and WebSphere Information Integrator Q replication offerings.

Important: To ensure successful and error-free implementation, we strongly encourage systematic and careful planning that involves identifying and gathering all required information prior to commencing implementation.

Towards this end, we have developed a template that identifies all the information required to implement a peer-to-peer replication topology, and the cross-relationships between the information elements to ensure a smooth implementation.

Figure 4-3 provides a high-level overview of the various objects involved in implementing a peer-to-peer replication topology involving two servers, and serves as a reference for the host and DB2 system information template (Table 4-1 on page 301), WebSphere MQ configuration information template (Table 4-2 on page 302), Q replication configuration information template (Table 4-3 on page 304), and Replication Alert Monitor configuration information template (Table 4-4 on page 305). Each of the information elements for each server is associated with a reference dentification such as A.7a or B.7b, where A represents one of the servers, and B" the other server. These reference IDs are then cross-referenced in the template itself, as well as the WebSphere II Q replication configuration screens and scripts. For example, in Table 4-2 on page 302 on WebSphere MQ information, for the XMITQ parameter (reference ID A.18a for Server A) in the SendQ, we require reference ID A.13 (which is the name of the TransmitQ), and that value is MQZ1XMITQ.

Attention: In Figure 4-3 there appear to be two sets of transmission queues, and sender and receiver channels on each server. However, there is only one set on each server, as can be deduced from the identical names. Figure 4-3 has the appearance of two sets so that the flow of data and messages between the two servers is easily understood.

Once all the information identified in the template had been collected, we can proceed with the actual implementation.

Note: The template shows certain parameter values that are required for Q replication (such as parameter DEFPSIST *must be* set to YES for the TransmitQ [reference ID A.13] and does *not* have a reference ID), while others can be customized for a particular environment (such as reference ID A.13d parameter MAXDEPTH *may be* set to any value).

We have collected all the identified information for the Carthage peer-to-peer replication implemented and recorded them in the templates in Table 4-1 on

page 301, Table 4-2 on page 302, Table 4-3 on page 304, and Table 4-4 on page 305. We are now ready to proceed with configuring the various resources.

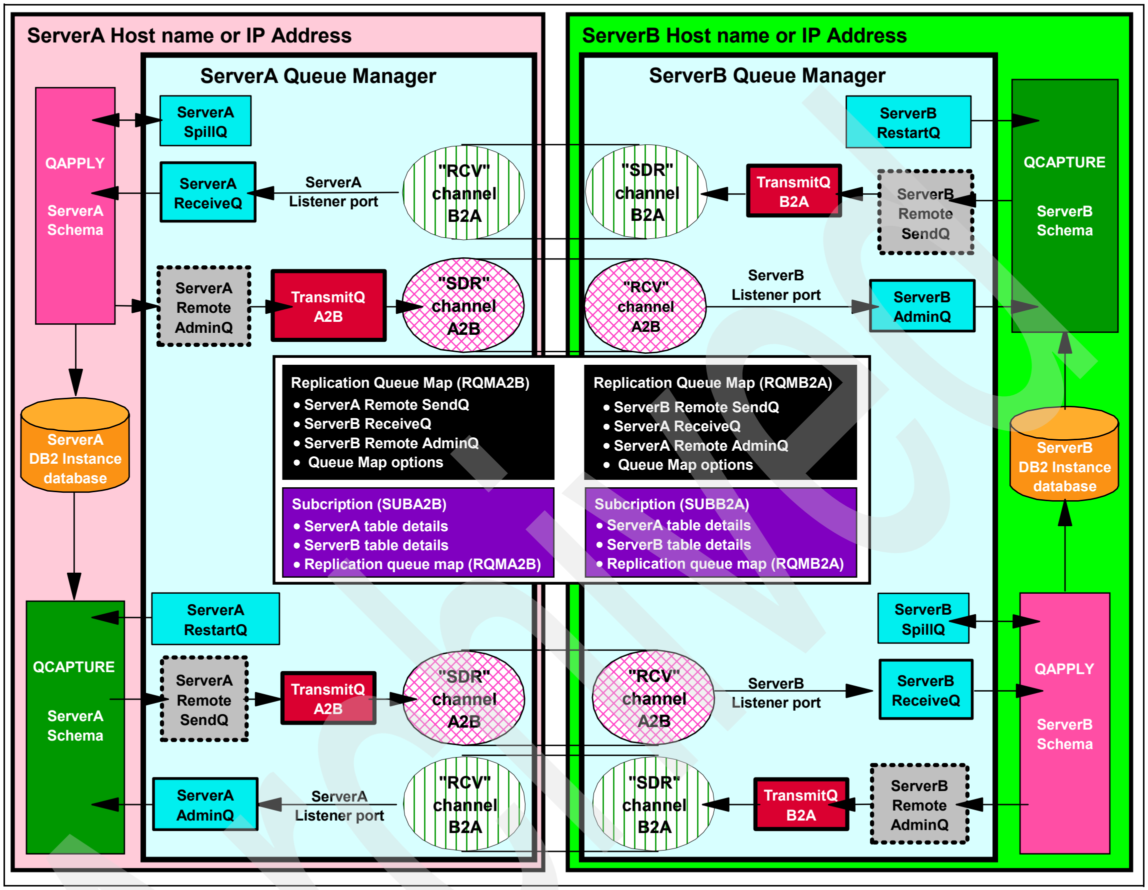


Figure 4-3 Peer-to-peer replication topology objects overview

Table 4-1 Host and DB2 system information

Description	Server A	Server B
Host name	(A.1) STLABD1	(B.1) WTSC53
Host IP address	(A.2) 9.30.132.94	(B.2) 9.12.6.77
DB2 subsystem name or instance owner	(A.3) DT11	(B.3) D8G1
DB2 group name (z/OS only)	(A.4) DT1G	(B.4) D8GG
DB2 location name (z/OS only)	(A.5) DSNT1	(B.5) DB8G

Description	Server A	Server B
DB2 tcp port	(A.6) 8010	(B.6) 38060
Database server/alias information ► Database server/alias ► Q Capture user ID and password ► Q Apply user ID and password	 ► (A.7a) DT11 ► (A.7b) DB2PROD/xxx ► (A.7c) DB2PROD/xxx	 ► (B.7a) D8G1 ► (B.7b) DB2PROD/xxx ► (B.7c) DB2PROD/xxx
User ID group (Unix only)	(A.8)	(B.8)
Other configuration user IDs requiring access	(A.9) QREPADM/xxx	(B.9) QREPADM/xxx
Logical database for control tables (z/OS only -- must preexist)	(A.10) QREPITSO	(B.10) QREPITSO

Table 4-2 WebSphere MQ information

Description	Server A	Server B
Queue Manager ► DEADQ (optional) ► MAXUMSGS (or use default) ► MAXMSGL (or use default)	(A.11) MQS1 ► (A.11a) ► (A.11b) 10000 ► (A.11c) 4194304	(B.11) MQZ1 ► (B.11a) ► (B.11b) 10000 ► (B.11c) 4194304
Listener port	(A.12) 1414	(B.12) 1540
TransmitQ ► USAGE ► PUT ► GET ► DEFPSIST ► TRIGGER ► TRIGTYPE ► TRIGDATA ► INITQ ► MAXMSGL (or use default) ► MAXDEPTH (or use default)	(A.13) MQZ1XMITQ ► XMITQ ► ENABLED ► ENABLED ► YES ► YES ► FIRST ► (A.13a)=(A14) MQS1.TO.MQZ1 ► (A.13b) SYSTEM.CHANNEL.INITQ ► (A.13c) 4194304 ► (A.13d) 999,999,999	(B.13) MQS1XMITQ ► XMITQ ► ENABLED ► ENABLED ► YES ► YES ► FIRST ► (B.13a)=(B14) MQZ1.TO.MQS1 ► (B.13b) SYSTEM.CHANNEL.INITQ ► (B.13c) 4194304 ► (B.13d) 999,999,999
SDR channel ► CHLTYPE ► TRPTYPE ► XMITQ ► CONNAME ► HBINT (or use default)	(A.14) MQS1.TO.MQZ1 ► SDR ► (A.14a) TCP ► (A.14b)=(A.13) MQZ1XMITQ ► (A.14c) 9.12.6.77 (1540) ► (A.14d) 300	(B.14) MQZ1.TO.MQS1 ► SDR ► (B.14a) TCP ► (B.14b)=(B.13) MQS1XMITQ ► (B.14c) 9.30.132.94 (1414) ► (B.14d) 300
RCV channel ► CHLTYPE ► TRPTYPE ► HBINT (or use default)	(A.15)=(B.14) MQZ1.TO.MQS1 ► RCVR ► (A.15a) TCP ► (A.15b) 300	(B.15)=(A.14) MQS1.TO.MQZ1 ► RCVR ► (B.15a) TCP ► (B.15b) 300

Description	Server A	Server B
RestartQ ► PUT ► GET ► SHARE ► DEFSOPT ► DEFPSIST ► MAXMSGL (or use default) ► MAXDEPTH (or use default)	(A.16) QREP.STLA.RESTARTQ ► ENABLED ► ENABLED ► YES ► SHARED ► YES ► (A.16a) 4194304 ► (A.16b) 999,999,999	(B.16) QREP.POKA.RESTARTQ ► ENABLED ► ENABLED ► YES ► SHARED ► YES ► (B.16a) 4194304 ► (B.16b) 999,999,999
AdminQ ► PUT ► GET ► SHARE ► DEFSOPT ► DEFPSIST ► MAXMSGL (or use default) ► MAXDEPTH (or use default)	(A.17) QREP.STLA.ADMINQ ► ENABLED ► ENABLED ► YES ► SHARED ► YES ► (A.17a) 4194304 ► (A.17b) 999,999,999	(B.17) QREP.POKA.ADMINQ ► ENABLED ► ENABLED ► YES ► SHARED ► YES ► (B.17a) 4194304 ► (B.17b) 999,999,999
SendQ (remote) ► PUT ► DEFPSIST ► XMITQ ► RNAME ► RQMNAME	(A.18) QREP.STLA.TO.POKA.SENDQ ► ENABLED ► YES ► (A.18a)=(A.13) MQZ1XMITQ ► (A.18b)=(B.19) QREP.STLA.TO.POKA.SENDQ ► (A.18c)=(B.11) MQZ1	(B.18) QREP.POKA.TO.STLA.SENDQ ► ENABLED ► YES ► (B.18a)=(B.13) MQS1XMITQ ► (B.18b)=(A.19) QREP.POKA.TO.STLA.SENDQ ► (B.18c)=(A.11) MQS1
ReceiveQ (local) ► PUT ► GET ► SHARE ► DEFSOPT ► DEFPSIST ► INDXTYPE ► MSGDLVSQ ► MAXMSGL (or use default) ► MAXDEPTH (or use default)	(A.19) = (B.18b) QREP.POKA.TO.STLA.RECVQ ► ENABLED ► ENABLED ► YES ► SHARED ► YES ► MSGID ► PRIORITY ► (A.19a) 4194304 ► (A.19b) 999,999,999	(B.19) = (A.18b) QREP.STLA.TO.POKA.RECVQ ► ENABLED ► ENABLED ► YES ► SHARED ► YES ► MSGID ► PRIORITY ► (B.19a) 4194304 ► (B.19b) 999,999,999
SpillQ ► DEFTYPE ► DEFSOPT ► MSGDLVSQ ► MAXMSGL (or use default) ► MAXDEPTH (or use default)	IBMQREP.SPILL.MODELQ ► PERMDYN ► SHARED ► FIFO ► (A.20a) 4194304 ► (A.20b) 999,999,999	IBMQREP.SPILL.MODELQ ► PERMDYN ► SHARED ► FIFO ► (B.20a) 4194304 ► (B.20b) 999,999,999

Description	Server A	Server B
AdminQ (remote) ► PUT ► DEFPSIST ► XMITQ ► RNAME ► RQMNAME	(A.21) = (B.17) QREP.POKA.ADMINQ ► ENABLED ► YES ► (A.21a)=(A.13) MQZ1XMITQ ► (A.21b)=(B.17) QREP.POKA.ADMINQ ► (A.21c)=(B.11) MQZ1	(B.21) = (A.17) QREP.STLA.ADMINQ ► ENABLED ► YES ► (B.21a)=(B.13) MQS1XMITQ ► (B.21b)=(A.17) QREP.STLA.ADMINQ ► (B.21c)=(A.11) MQS1

Table 4-3 Q replication configuration information

Description	Server A	Server B
QCAPTURE Schema	(A.22) ITSO	(B.22) ITSO
QAPPLY Schema	(A.23) ITSO	(B.23) ITSO
Q Capture path (log files)	(A.24) //DB2PROD	(B.24) //DB2PROD
Q Apply path (log files)	(A.25) //DB2PROD	(B.25) //DB2PROD
Q replication type (U/B/P)	(A.26) P	(B.26) P
Replication Queue Map name ► Max message length (KB) ► Error handling ► Num apply agents ► Memory buffer for Recvq (MB) ► Allow QCapture to send heartbeat ► Heartbeat interval (sec)	(A.27) QMAP_STL_TO_POK ► (A.27a) 64 ► (A.27b) Stop Q Capture ► (A.27c) 16 ► (A.27d) 2 ► (A.27e) yes ► (A.27f) 60	(B.27) QMAP_POK_TO_STL ► (B.27a) 64 ► (B.27b) Stop Q Capture ► (B.27c) 16 ► (B.27d) 2 ► (B.27e) yes ► (B.27f) 60
Q Subscriptions parameters ► Target table schema/creator ► Target table index schema/name ► Target tablespaces ► Check for conflicts setting ► Conflict resolution action ► Error response action ► Initial load option ► Source server for initial load ► Qsub start automatic	► (A.28a) same ► (A.28b) same ► (A.28c) each gets own ► (A.28d) all columns ► (A.28e) ► (A.28f) Stop Q Apply ► (A.28g) Manual ► (A.28h) DT11 ► (A.28i) yes	► (B.28a) same ► (B.28b) same ► (B.28c) each gets own ► (B.28d) all columns ► (B.28e) ► (B.28f) Stop Q Apply ► (B.28g) Manual ► (B.28h) DT11 ► (B.28i) yes
Q Subscriptions list	(A.29) 23 ITSO-xxxx tables	(B.29) 23 ITSO-xxxx tables

Table 4-4 Replication Alert Monitor configuration information

Description	**Designated server**
Host name	(M.1) WTSC53
Host IP Address	(M.2) 9.12.6.77
DB2 subsystem name or instance owner	(M.3) D8G1
DB2 tcp port	(M.4) 38060
Monitor database server/alias information ► Database server/alias ► db2instance ► User ID/password ► Contact name ► Contact email address ► Monitor qualifier ► Schema name	 ► (M.5a) DB8G ► (M.5b) D8G1 ► (M.5c) QREPADM/xxx ► (M.5d) QMONITOR ► (M.5e) QREPADM@us.ibm.com ► (M.5f) ITSO_QMON ► ASN
Monitor log path (not for z/OS)	(M.6)
DB2 group name (z/OS only)	(M.7) D8GG
DB2 location name (z/OS only)	(M.8) DB8G

Note: These tables, and our configuration, use a single subsystem member of the DB2 share group as the database alias for connecting replication programs to the database. If one is running in a share group, for greater flexibility consider using the DB2 group name as the alias, and placing the MQ objects in an MQ cluster. That will allow replication programs to run on any member of the sysplex.

4.5.4 Step 4: Set up user IDs, privileges, and database servers

The steps involved are identical to those in the setup described in 3.5.4, “Step 4: Set up user IDs, privileges, and database servers” on page 89.

Time synchronization

When using peer-to-peer replication it is critical to keep all system times synchronized. Otherwise one system will always take precedence over the others.

For details on system time information and synchronization with external time references, refer to Appendix E of *IBM 9037 Sysplex Timer and System/390 Time Management*, GC66-3264-00.

4.5.5 Step 5: Configure WebSphere MQ

The steps involved are identical to those for the setup described in 3.5.5, "Step 5: Configure WebSphere MQ" on page 101.

4.5.6 Step 6: Configure and activate Q replication using GUI

In this section we document the step-by-step configuration of Q replication for the peer-to-peer replication topology in our fictitious company. Figure 4-4 expands "STEP 6: Configure & activate Q replication (GUI or commands)" (in Figure 4-2 on page 299) into a number of substeps involved in configuring Q replication. Each of these substeps is described in detail in the following subsections.

Very important: The Replication Center is typically used to configure and manage an SQL or Q replication environment because of its ease-of-use GUI interface. In order for a Replication Center client to be aware of the database servers that need to be defined as a first and second server in Figure 4-34 on page 337 in a replication environment, the Q Capture and Q Apply control tables *must* be created from the *same* Replication Center client. If the Q Capture and Q Apply control tables are created by ASNCLP scripts, or using *another* Replication Center client, then those database servers will *not* appear in the list of available servers for the first and second server selection. In such cases, you must catalog them in this Replication Center client using the process described in Appendix H, "Cataloging remote database servers" on page 881.

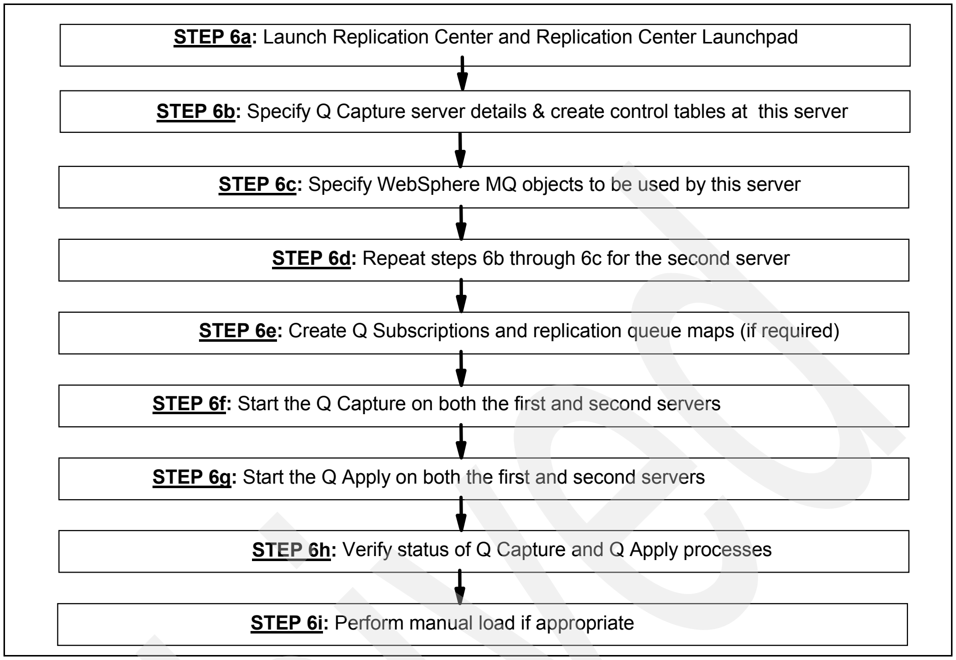


Figure 4-4 Overview of Q replication configuration steps

Step 6a: Launch Replication Center

Figure 4-5 shows the launching of the Replication Center from the DB2 Control Center by clicking the **Tools** tab and selecting **Replication Center**. This displays the Replication Center Launchpad screen shown in Figure 4-6.

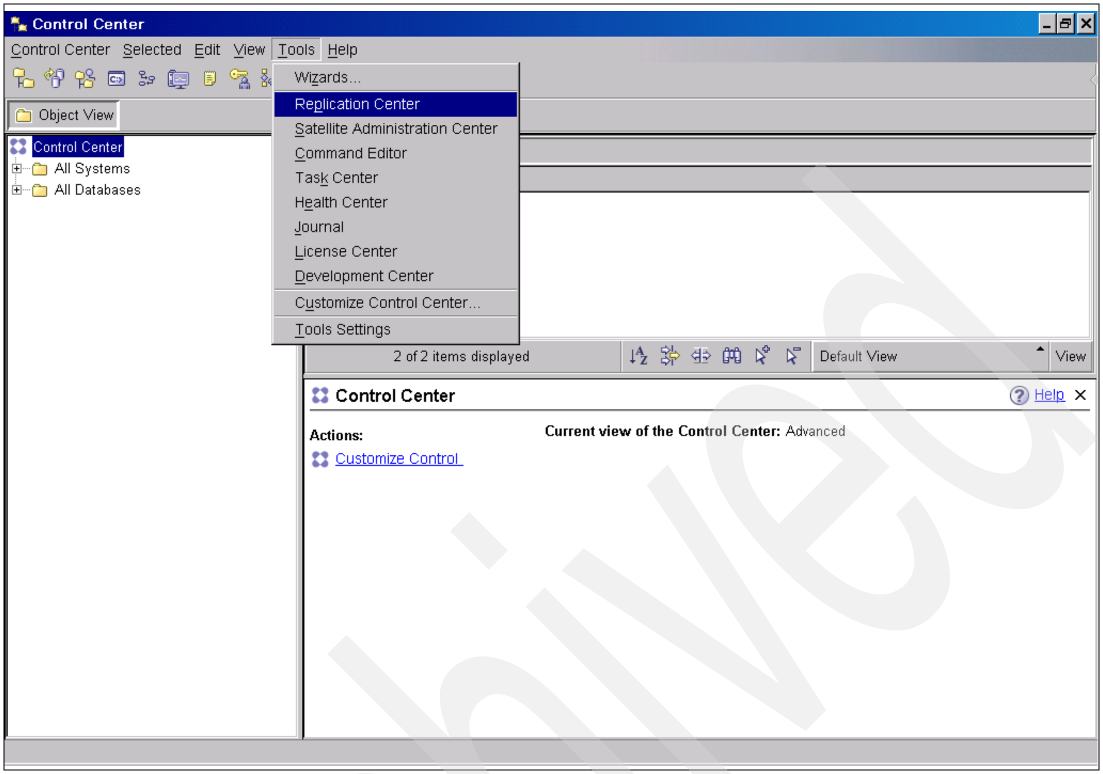


Figure 4-5 Launch Replication Center

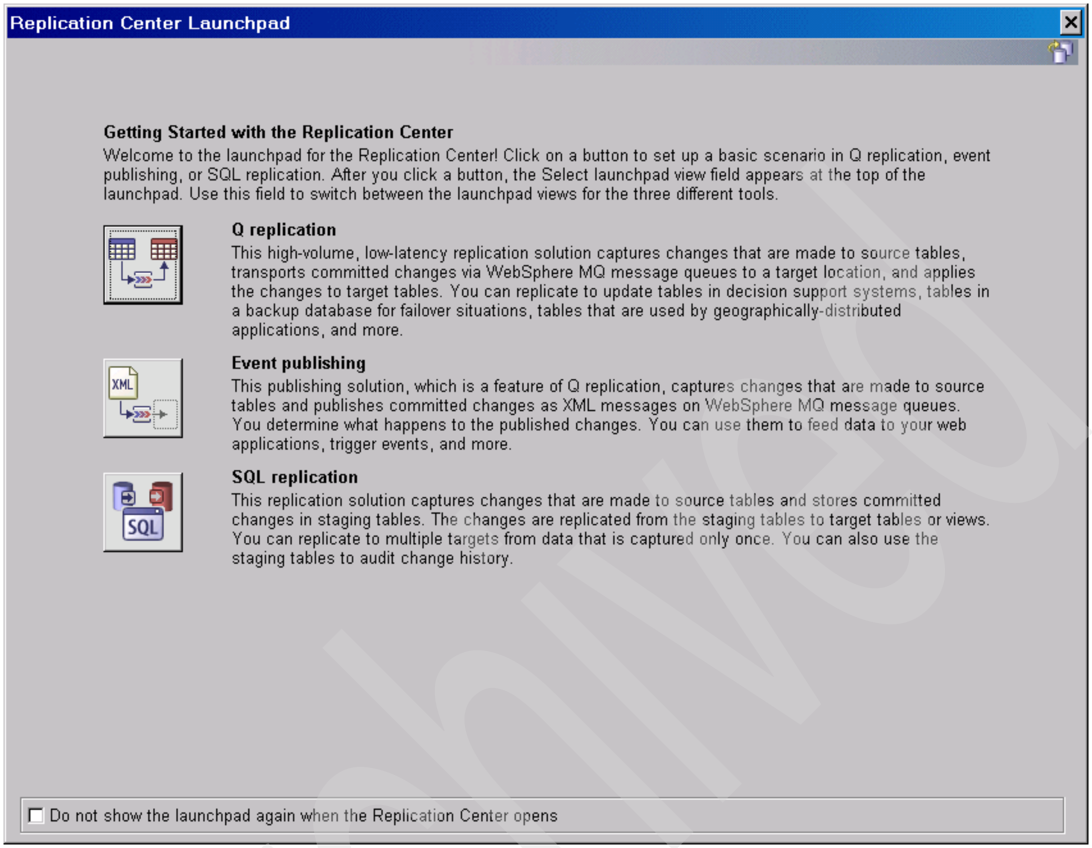


Figure 4-6 Replication Center Launchpad

> **Note:** Figure 4-6 can be skipped in the future by selecting the **Do not show this launchpad again when the Replication Center opens** option.

Step 6b: Specify Q Capture details

Click **Q replication** in Figure 4-6 to display Figure 4-7, which describes the five steps in setting up and activating Q replication, as follows:

1. Create Q Capture Control Tables.
2. Create Q Apply Control Tables.
3. Create a Q Subscription.
4. Start a Q Capture program.
5. Start a Q Apply program.

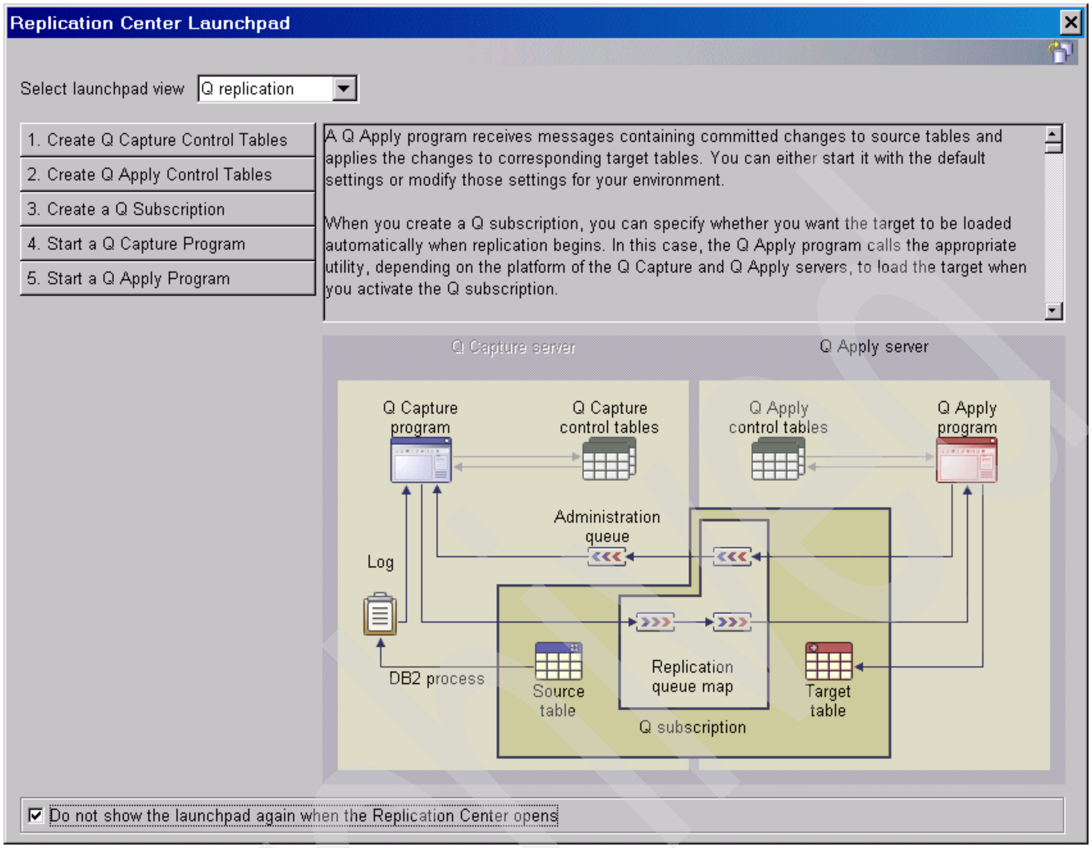


Figure 4-7 Five steps in setting up Q replication infrastructure

Click **Create Q Capture Control Tables** to display Figure 4-8, which enables the specification of details for the Q Capture infrastructure.

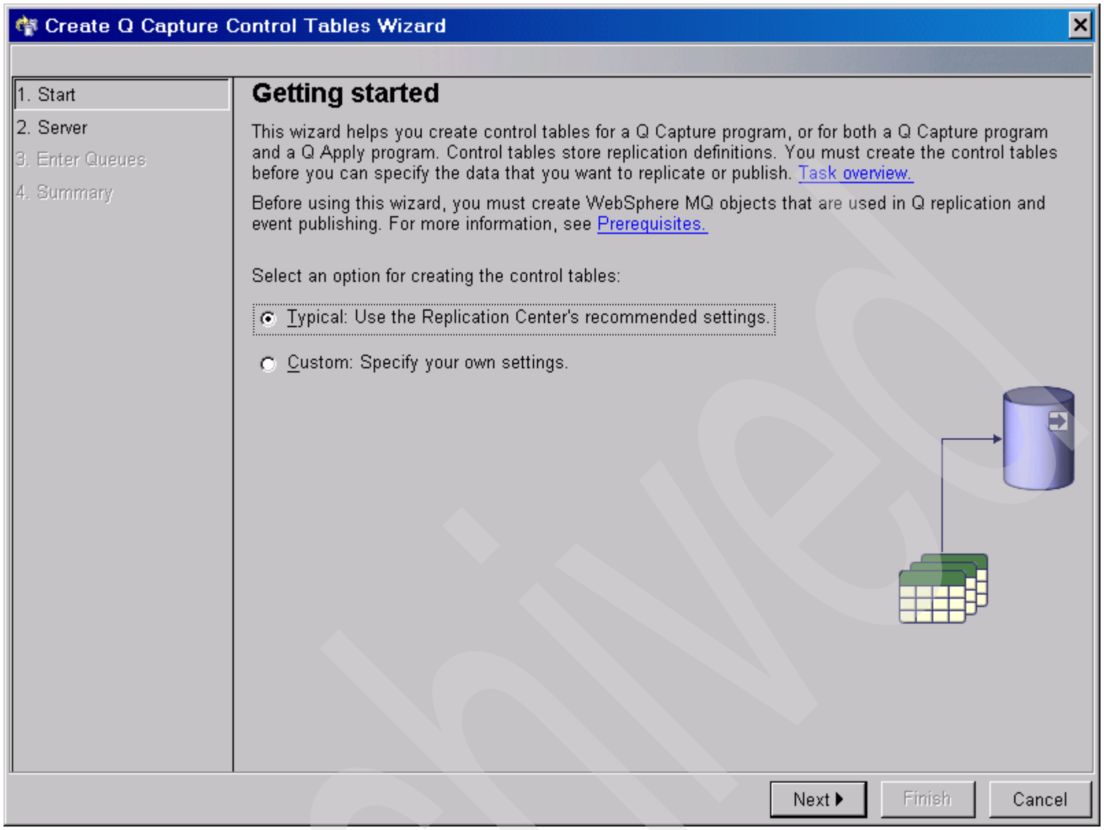


Figure 4-8 Getting Started - Create Q Capture Control Tables Wizard

We selected the **Typical** setting and clicked **Next** to display Figure 4-9, where the Q Capture server details are specified.

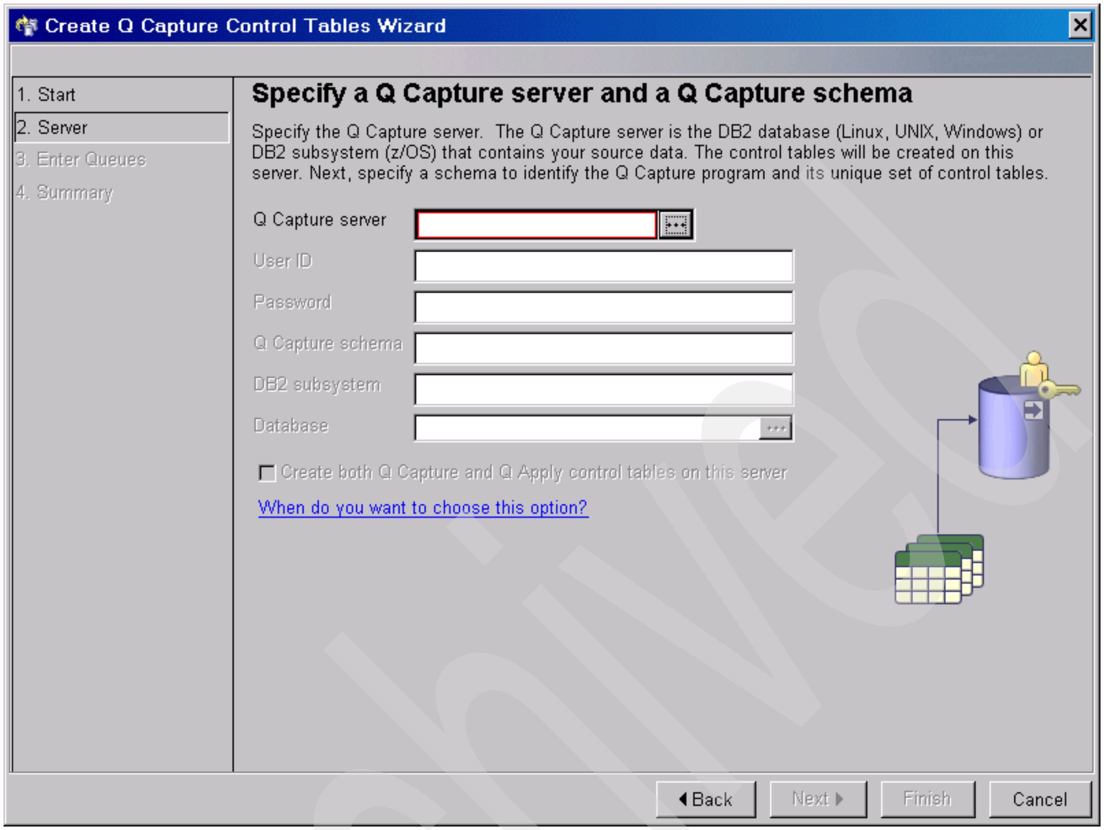


Figure 4-9 Server - Create Q Capture Control Tables Wizard

Click the **...** tab to display the list of available servers, as shown in Figure 4-10.

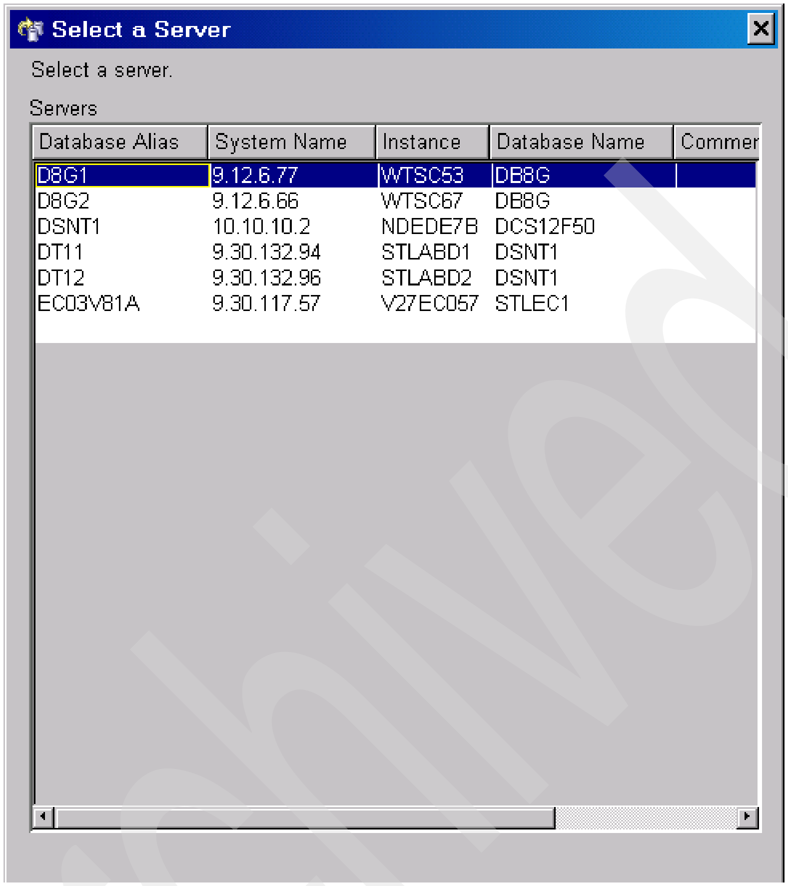


Figure 4-10 Select a Server

Select the **D8G1** database alias on WTSC53 as the database server for Q Capture, and click **OK** to display Figure 4-11 for providing Q Capture details.

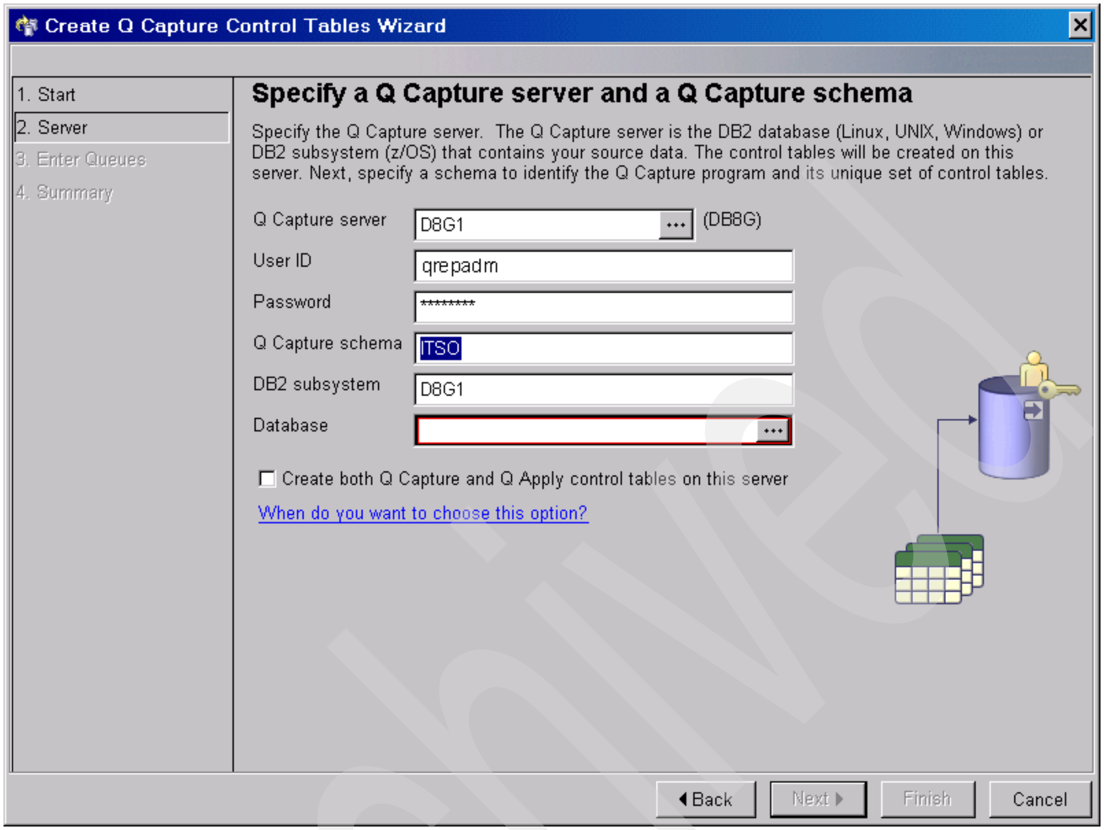


Figure 4-11 Q Capture server and schema details

Supply the Q Capture user ID and password "qrepadm/xxx" (reference B.9b in Table 4-2 on page 302) and Q Capture schema "ITSO" (reference B.22 in Table 4-3 on page 304). Click the **...** button to obtain a list of available databases, as shown in Figure 4-12.

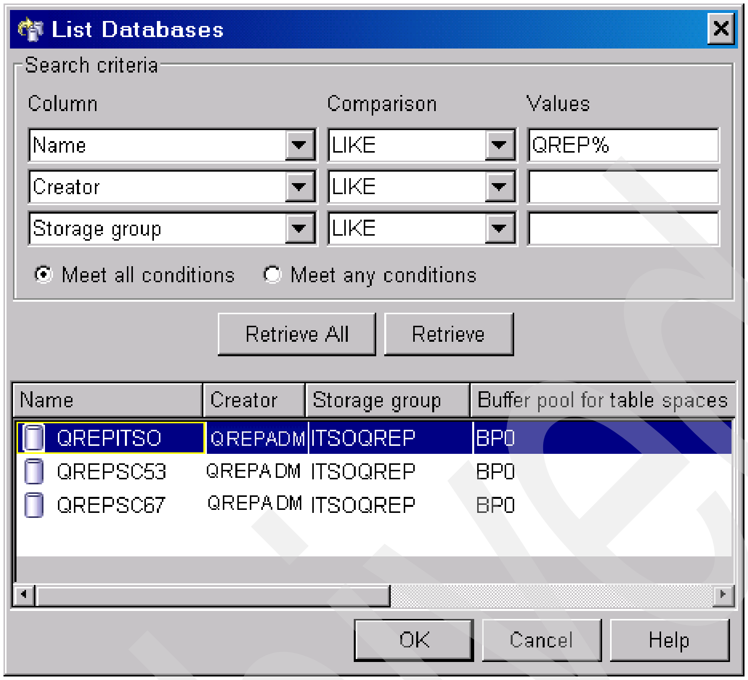


Figure 4-12 List Databases

Specify the selection criteria for the names of databases beginning with the characters QREP and click the **Retrieve** button to the list of qualifying databases as shown. Highlight the database named **QREPITSO** (which will contain the control tables—reference B.10 in Table 4-1 on page 301) and click **OK** to proceed to Figure 4-13.

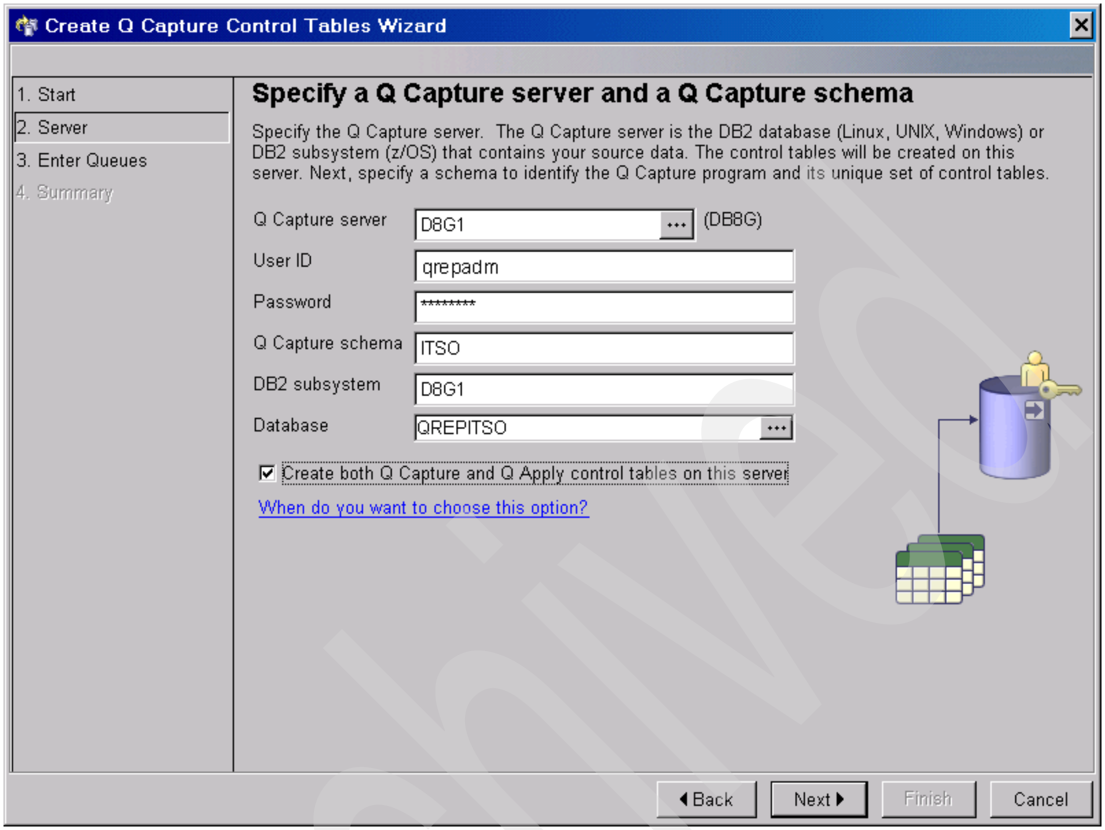


Figure 4-13 Q Capture server and schema details

Select the **Create both Q Capture and Q Apply control tables on this server** option in Figure 4-13.

Click **Next** to display Figure 4-14 for defining WebSphere MQ objects.

Step 6c: Specify WebSphere MQ objects

Supply WebSphere MQ queues for this server. This includes the Queue Manager “MQZ1” (reference B.11 in Table 4-3 on page 304), Administration queue “QREP.POKA.ADMINQ” (reference B.17 in Table 4-3 on page 304), and Restart queue “QREP.POKA.RESTARTQ” (reference B.16 in Table 4-3 on page 304).

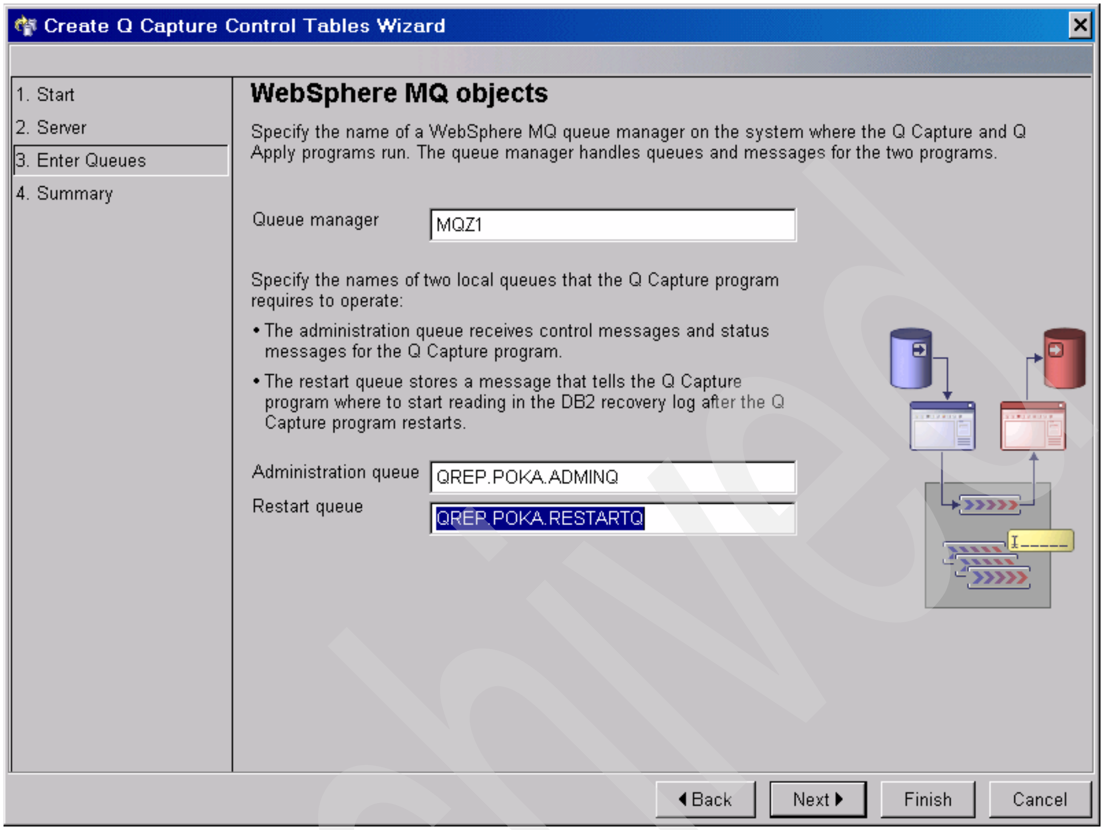


Figure 4-14 Q information for Q Capture server

Click **Next** to continue to Figure 4-15, which summarizes the information provided in the previous screens.

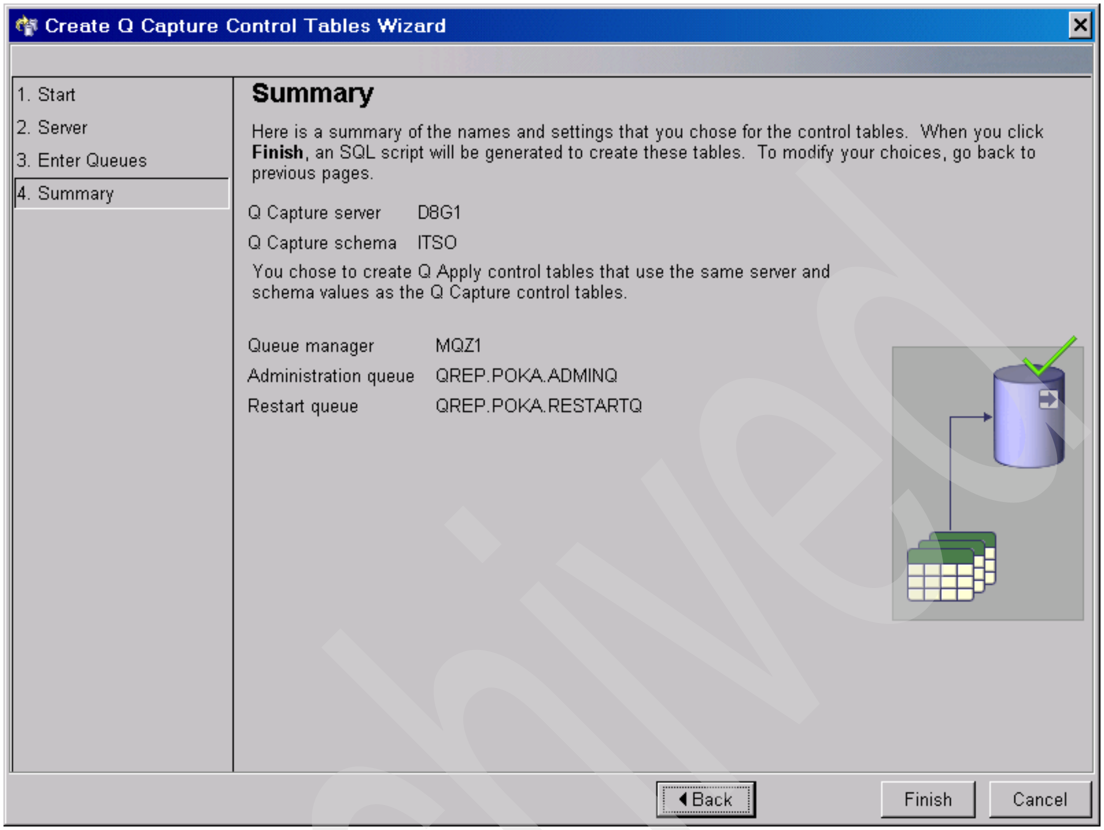


Figure 4-15 Q Capture server summary information

Verify the accuracy of the information supplied, and click **Finish** to generate the scripts that need to be executed on the Q Capture server to define the specified objects, as shown in Figure 4-16.

Click the **Back** button to go back and correct any errors.

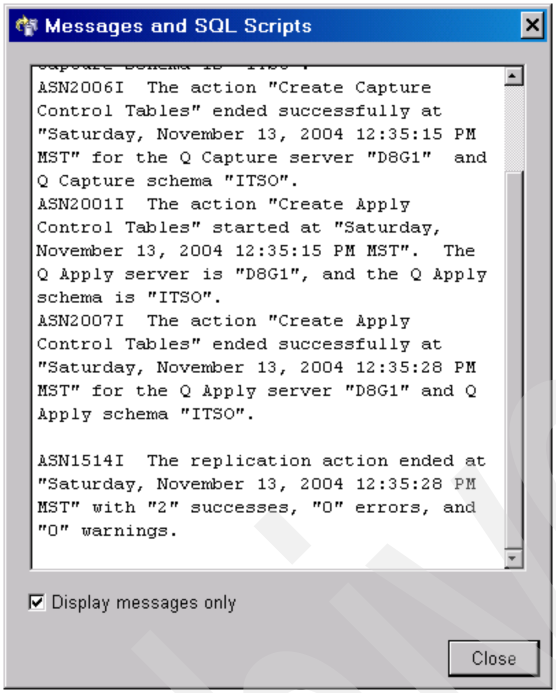


Figure 4-16 Messages and SQL Scripts screen for Q Capture server

Check that the errors are 0. Click **Close** to display Figure 4-17 and decide whether to execute the script or save it for later execution.

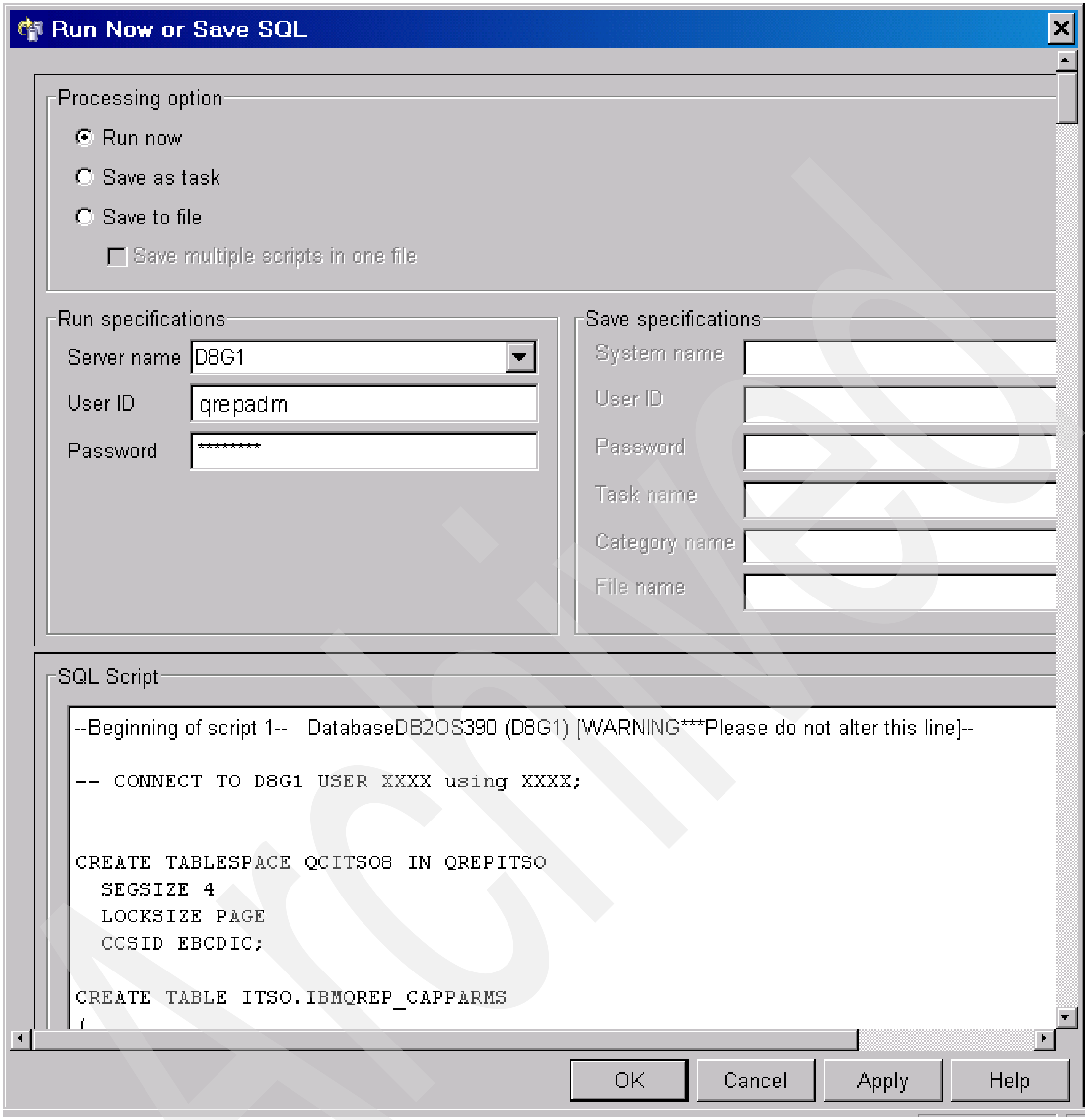


Figure 4-17 Run Now or Save SQL screen

> **Note:** At this point, one may choose to save the generated SQL to a file by selecting the **Save to file** option. Port it to MVS and execute it via SPUFI or DSNTEP2. If one chooses to save and execute the SQL scripts under MVS control, one must execute it before generating further SQL scripts, as the tables need to exist prior to the generation of the next SQL script.

However, we chose to execute the generated scripts immediately by selecting the **Run now** option.

Attention: Database Administration Server (DAS) is *not* required to execute these scripts from the workstation. DAS is only required to start and communicate with the Capture, Apply, and Monitor programs from the workstation.

Click **OK** to execute the SQL scripts, and wait for the successful completion message, as shown in Figure 4-18.

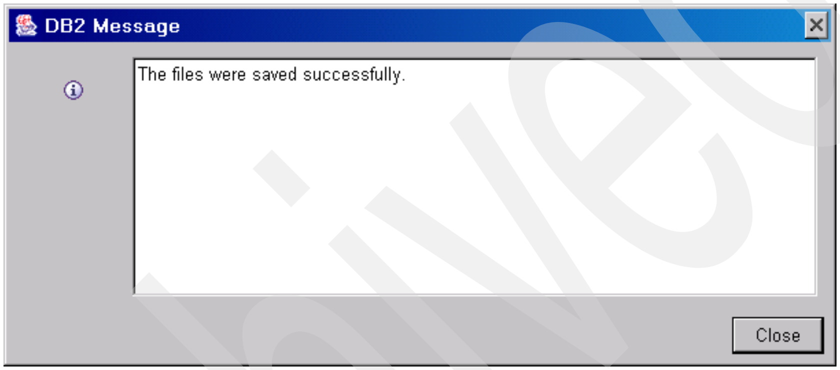


Figure 4-18 SQL scripts ran successfully message

Click **Close.** The control tables for both Q Capture and Q Apply on D8G1 (reference B.7 in template Table 4-1 on page 301) have been created.

Step 6d: Repeat steps 6b through 6c for second server

The same steps for the Q Capture server on WTSC53 need to be repeated for the STLABD1 using the appropriate reference information in Table 4-1 on page 301, Table 4-2 on page 302, and Table 4-3 on page 304 (as shown in Figure 4-19 through Figure 4-30 on page 333).

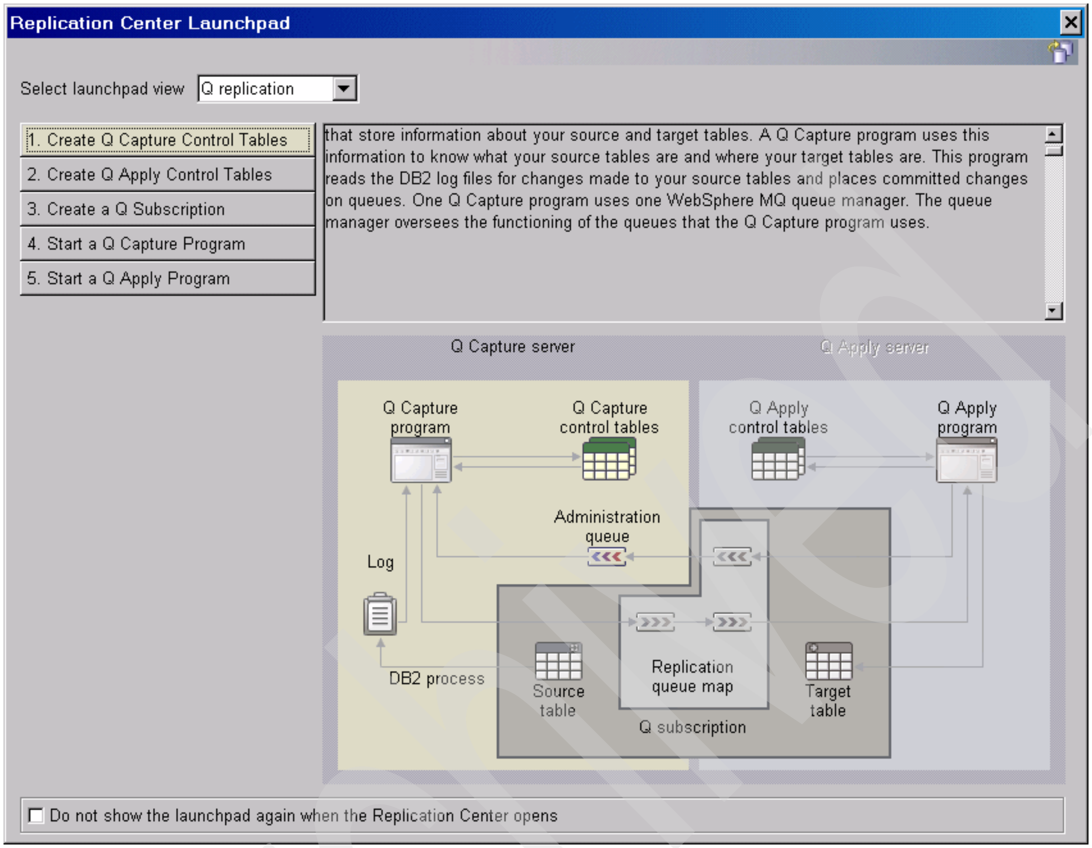


Figure 4-19 Five steps in setting up Q replication infrastructure

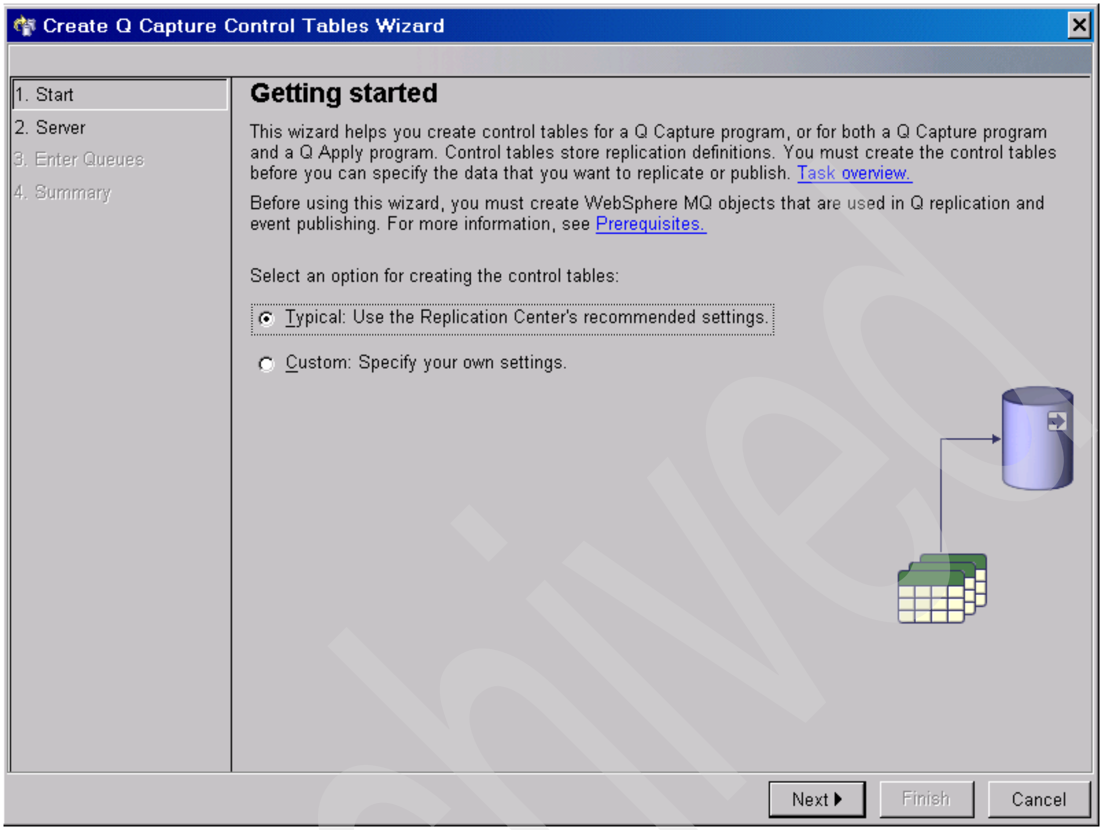


Figure 4-20 Getting started - Create Q Capture Control Tables Wizard

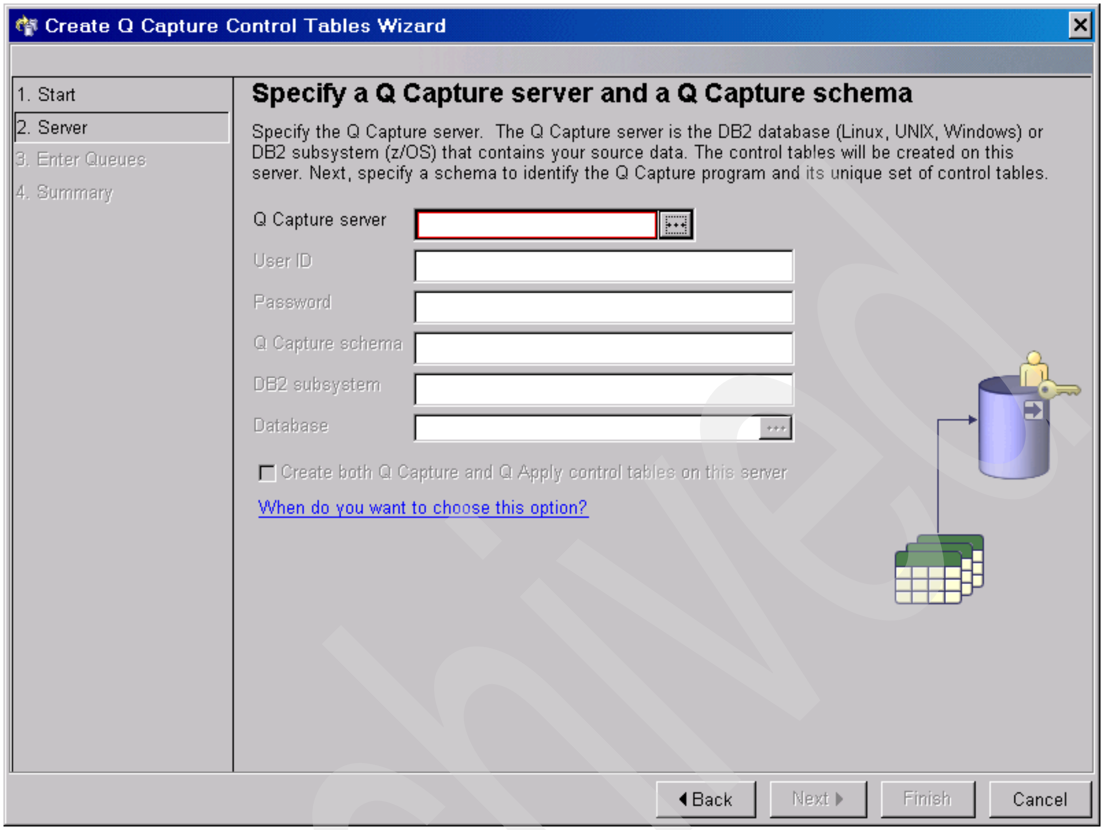


Figure 4-21 Server - Create Q Capture Control Tables Wizard

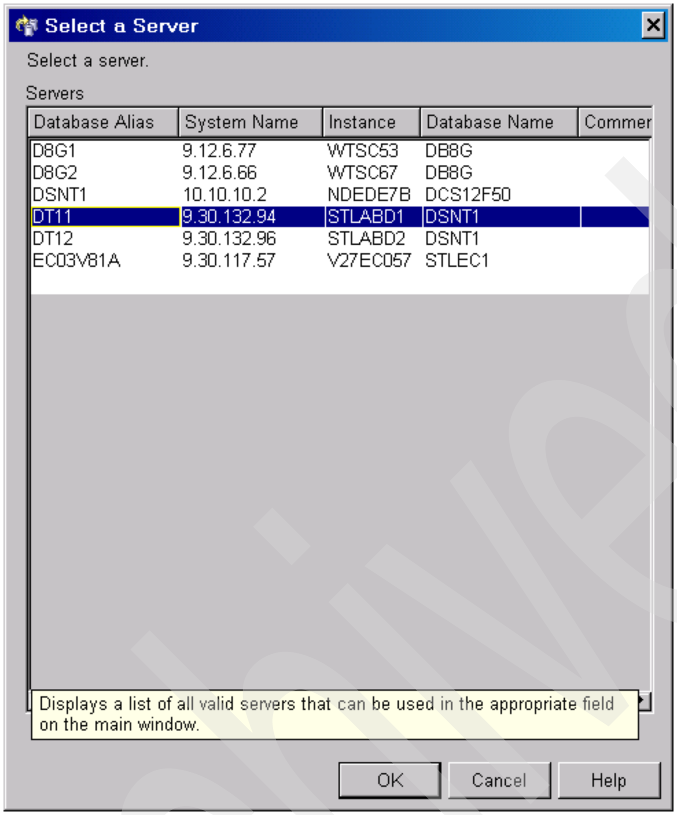


Figure 4-22 Select a Server

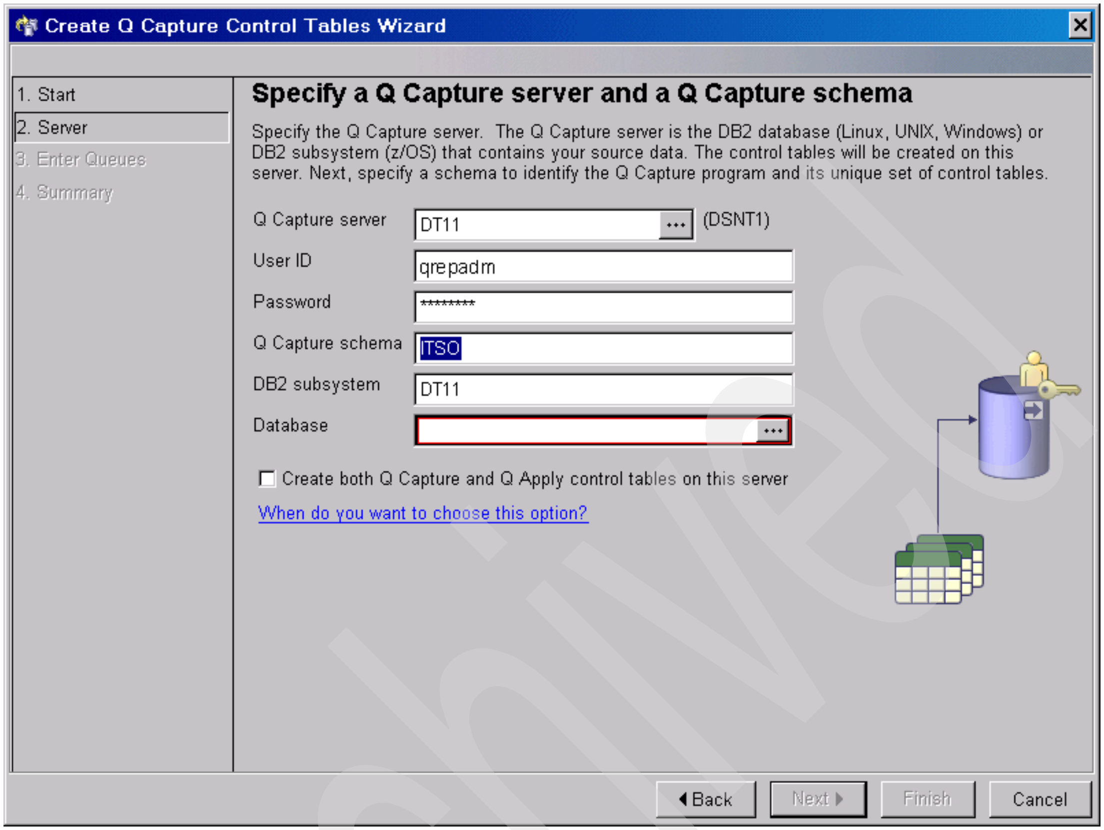


Figure 4-23 Q Capture server and schema details

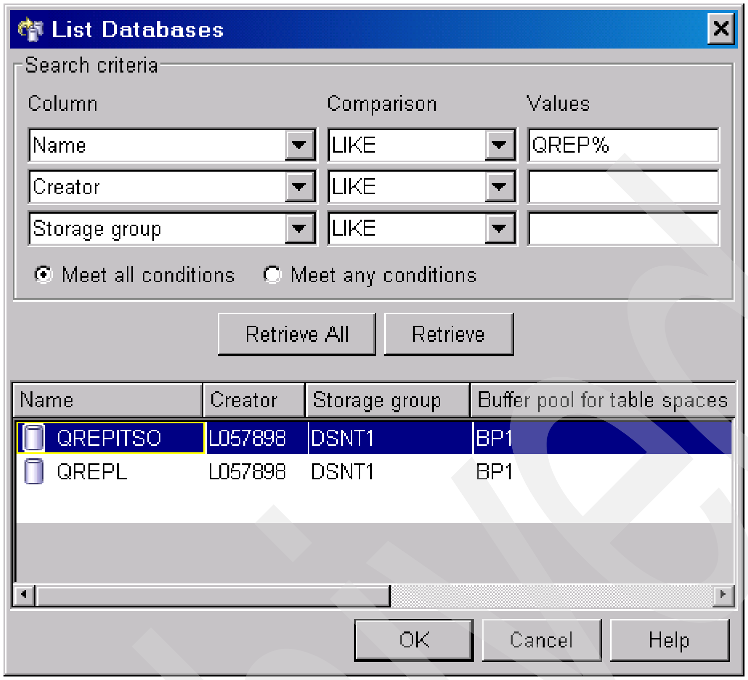


Figure 4-24 List Databases

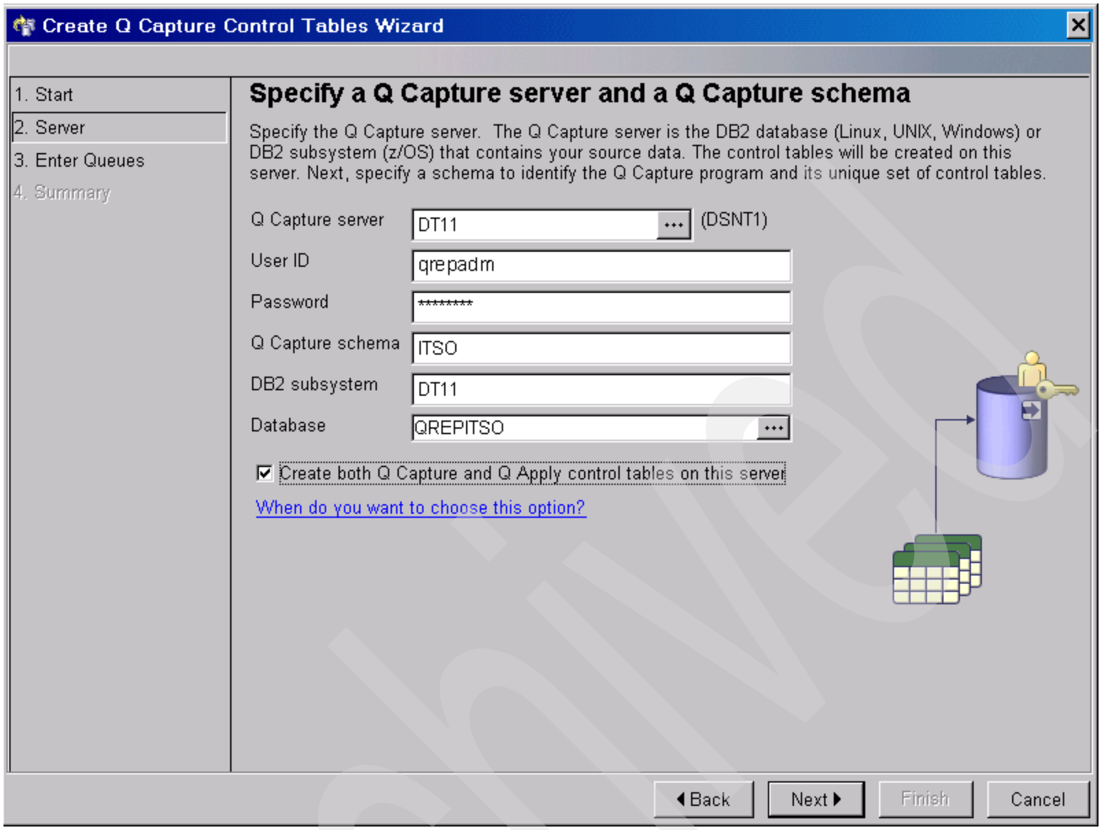


Figure 4-25 Q information for Q Capture server

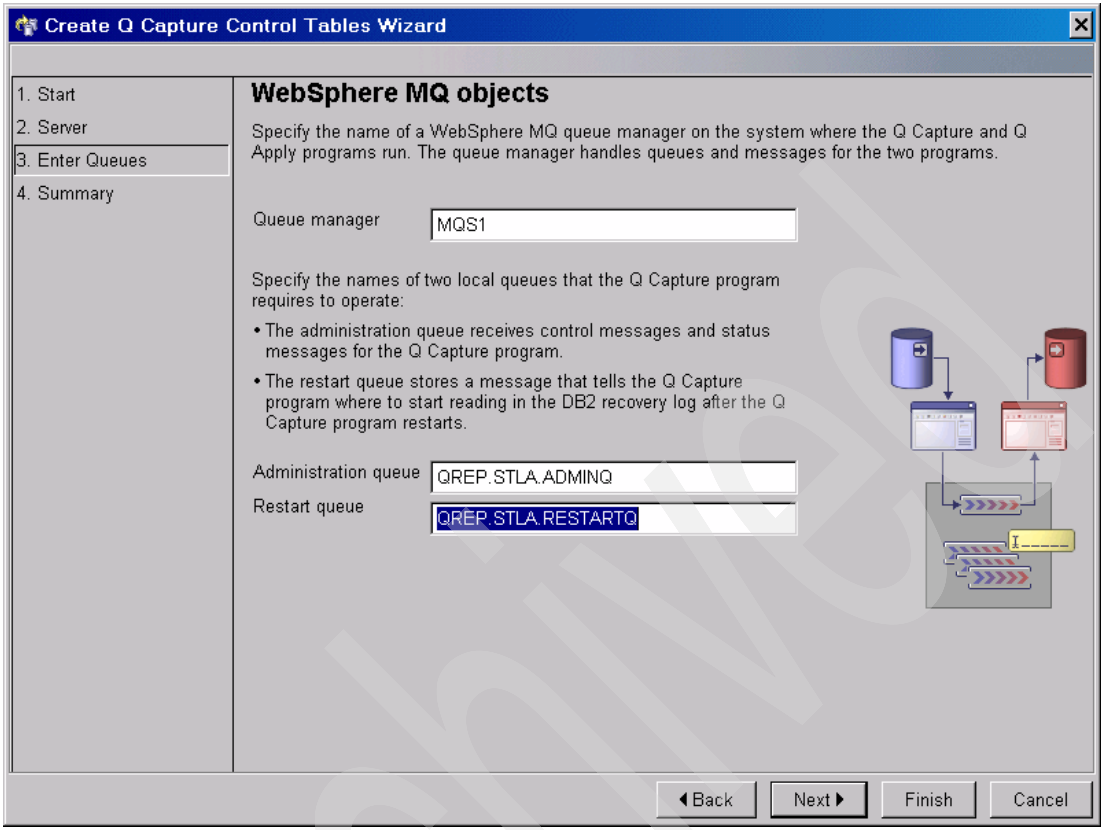


Figure 4-26 Queue information for Q Capture server

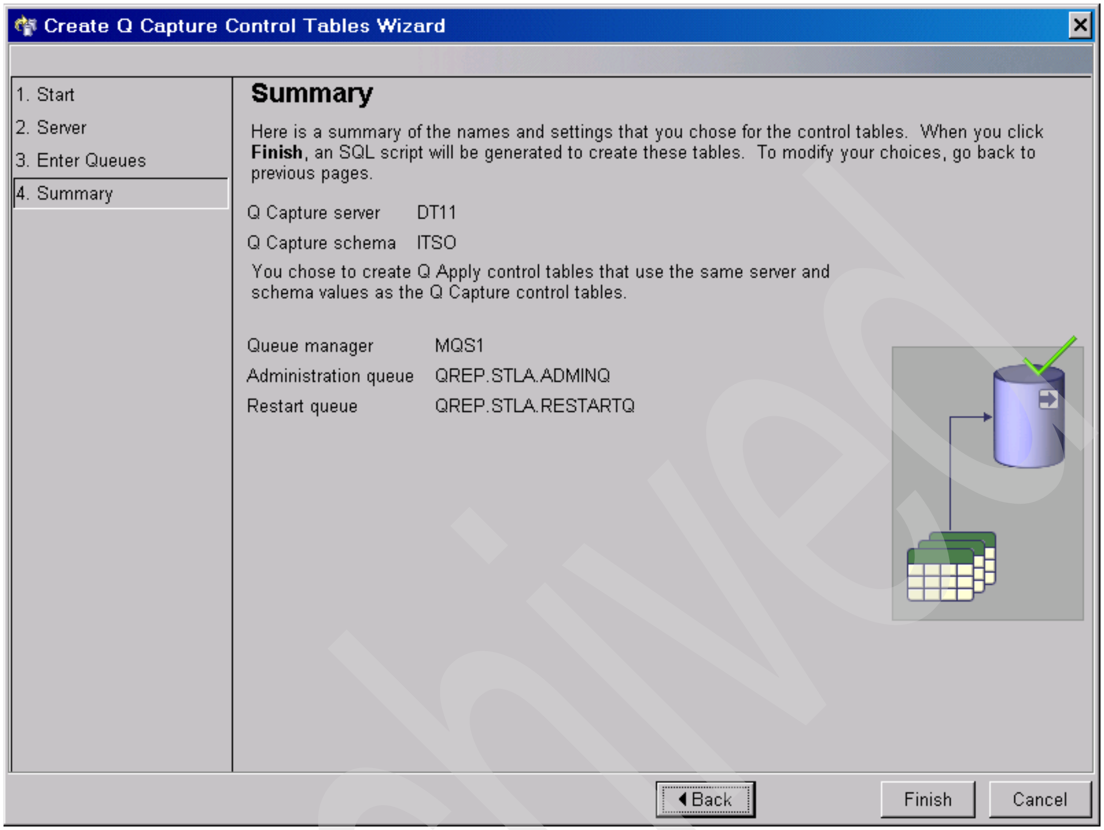


Figure 4-27 Q Capture server summary information

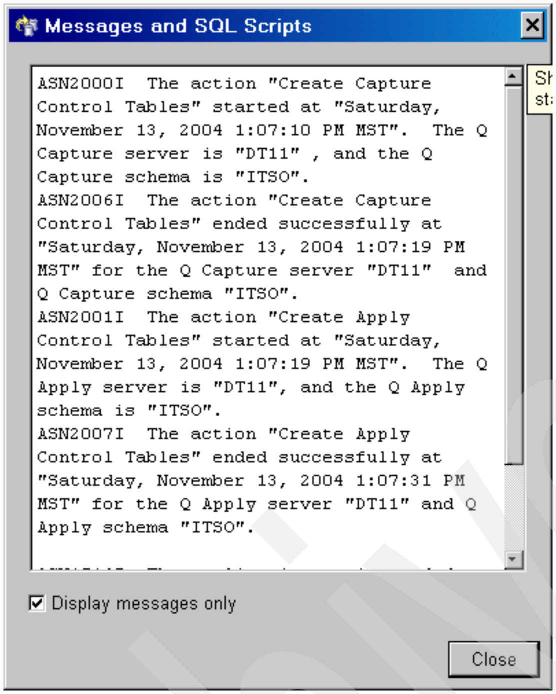


Figure 4-28 Messages and SQL Scripts screen for Q Capture server

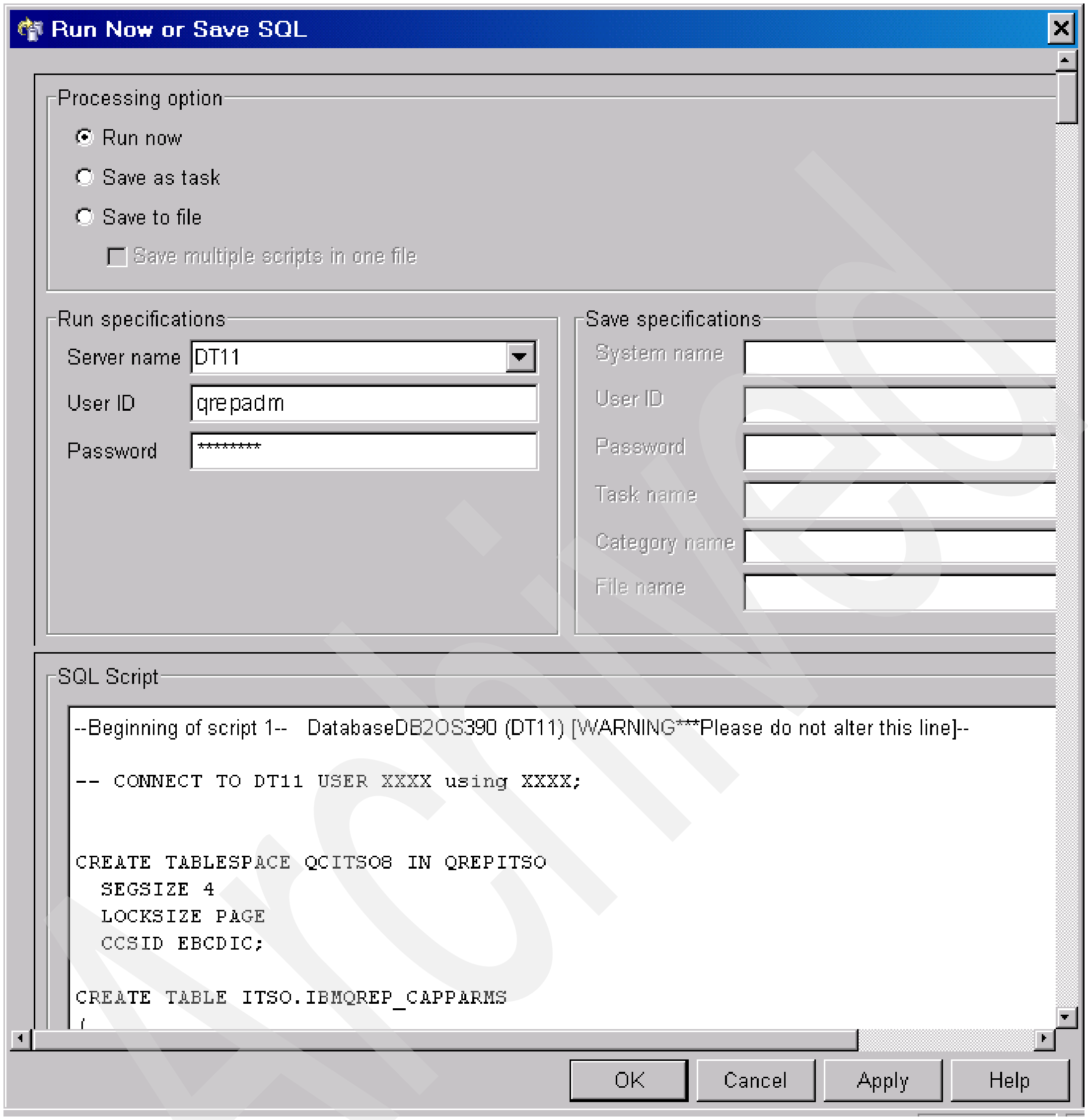


Figure 4-29 Run Now or Save SQL screen

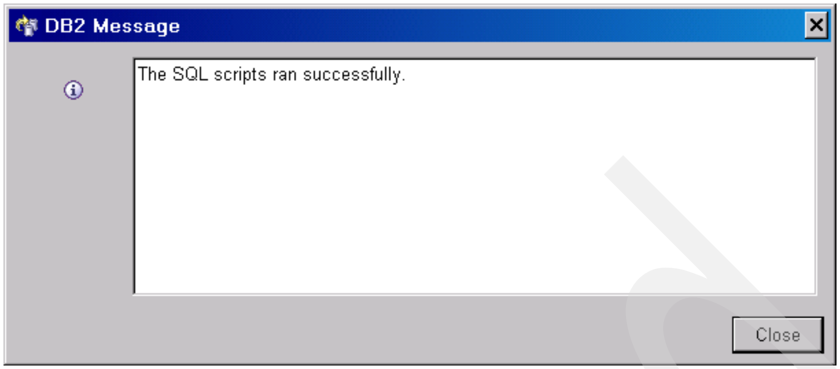


Figure 4-30 SQL scripts ran successfully message

Step 6e: Create Q Subscriptions and replication queue maps

After the Q Capture and Q Apply Control Tables have been created, the Q subscriptions and replication queue maps need to be created. Click the **Create a Q Subscription** option in Figure 4-31 to display Figure 4-32 to begin the process.

> **Note:** Since we created the Q Apply control tables earlier by selecting the **Create both Q Capture and Q Apply control tables on this server** in Figure 4-25 on page 328, we skipped the Create Q Apply Control Tables option in Figure 4-31.

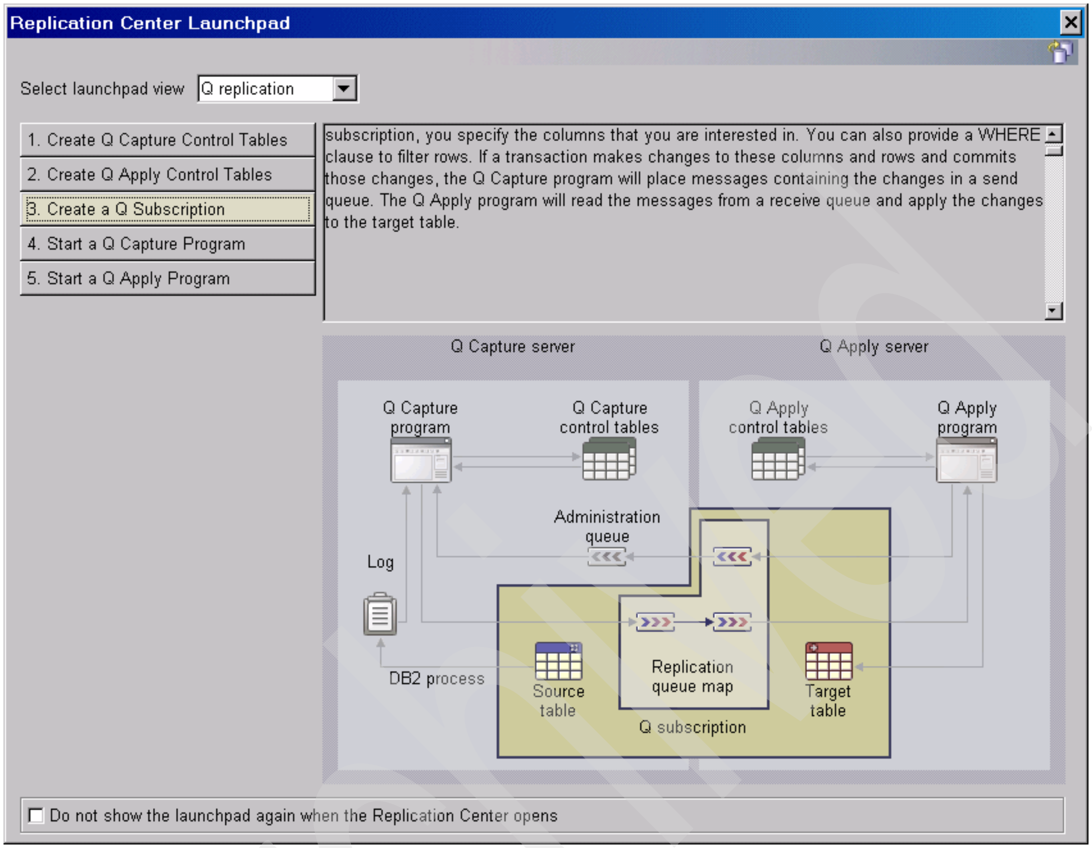


Figure 4-31 Five steps in setting up Q Replication infrastructure

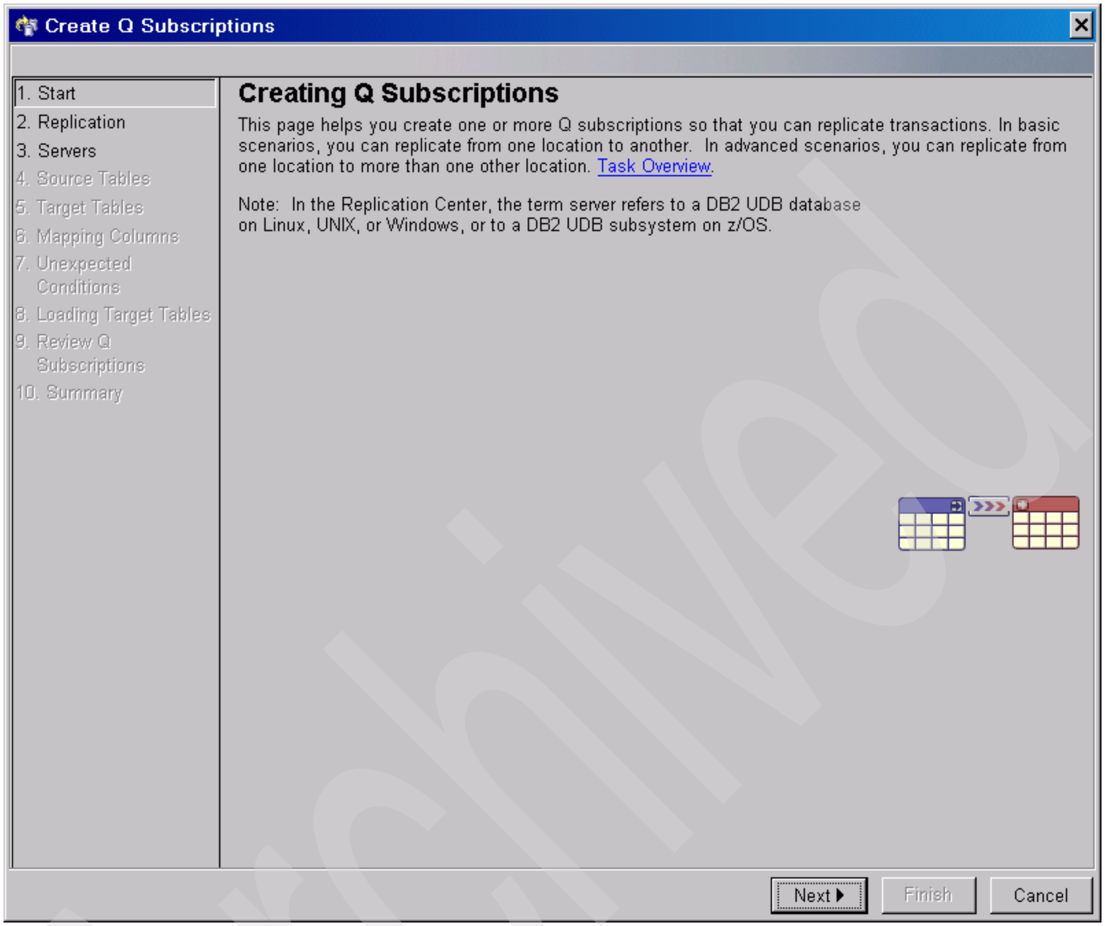


Figure 4-32 Create Q Subscriptions

Click **Next** to continue to Figure 4-33 to choose the type of replication topology desired.

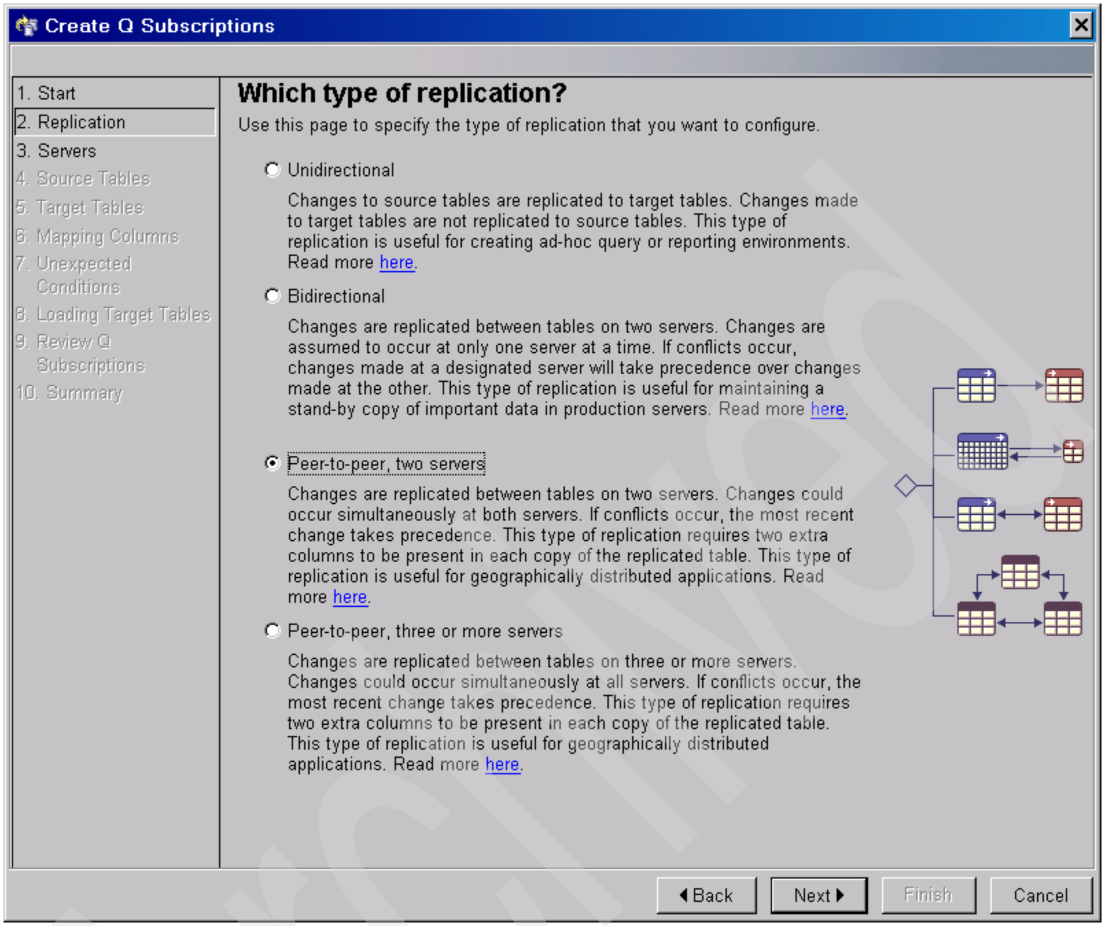


Figure 4-33 Which type of replication

Select the **Peer-to-peer, two servers** setting and click **Next** to provide details about the servers involved in the replication topology, as shown in Figure 4-34.

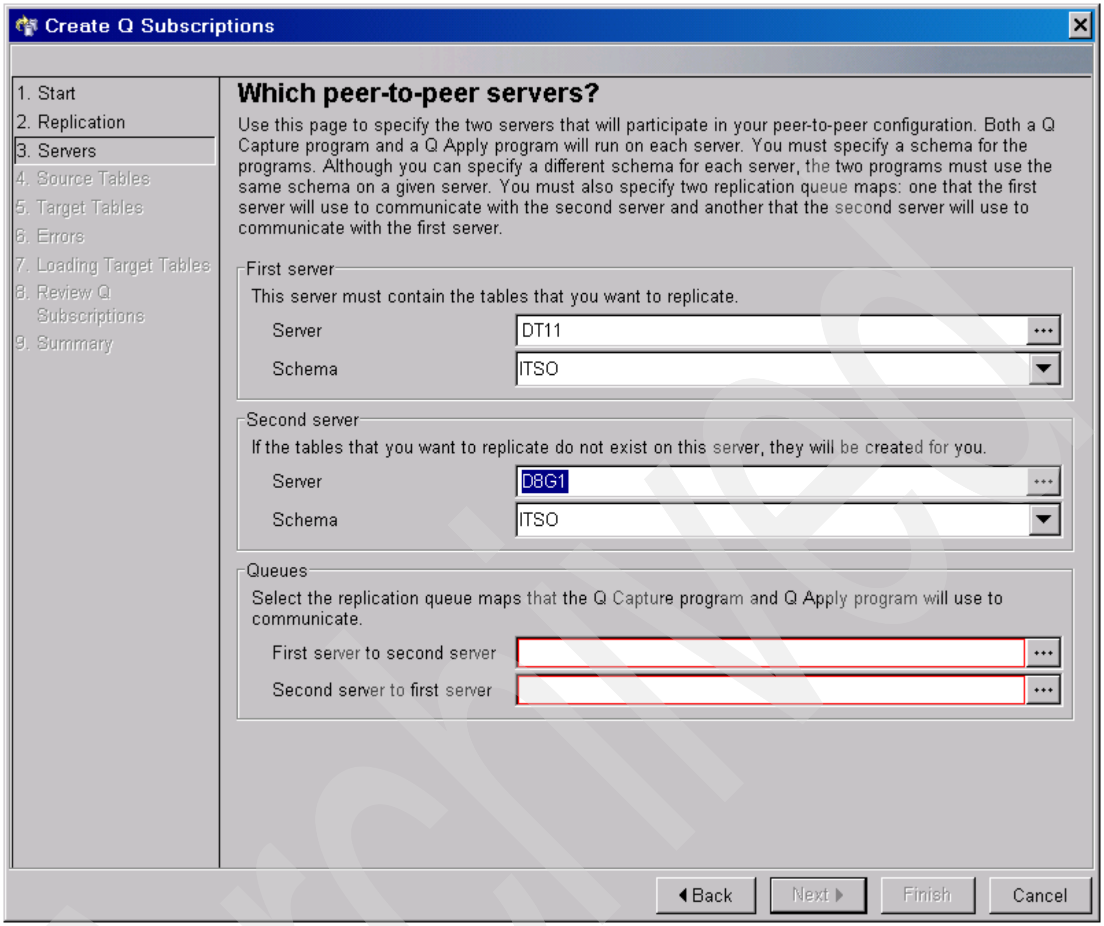


Figure 4-34 Which peer-to-peer servers

Supply the values for the First server (Server value "DT11" reference A.7a in Table 4-1 on page 301, Schema value "ITSO" reference A.22 in Table 4-3 on page 304, and Second server (Server value "D8G1" reference B.7a in Table 4-1 on page 301), Schema value "ITSO" reference B.22 in Table 4-3 on page 304.

Click the **...** button for the "First server to second server" to get a list of the available replication queue maps, as shown in Figure 4-35.

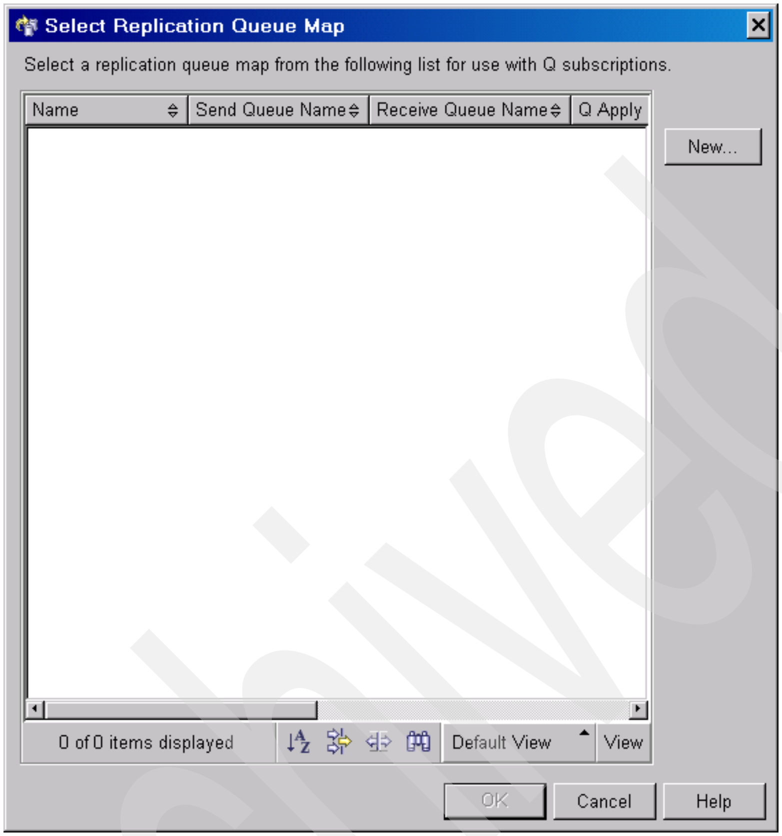


Figure 4-35   Select Replication Queue Map

Since no replication queue maps have been defined as yet, Figure 4-35 has no list of existing replication queue maps. A new replication queue map needs to be created by clicking the **New...** button. This displays Figure 4-36.

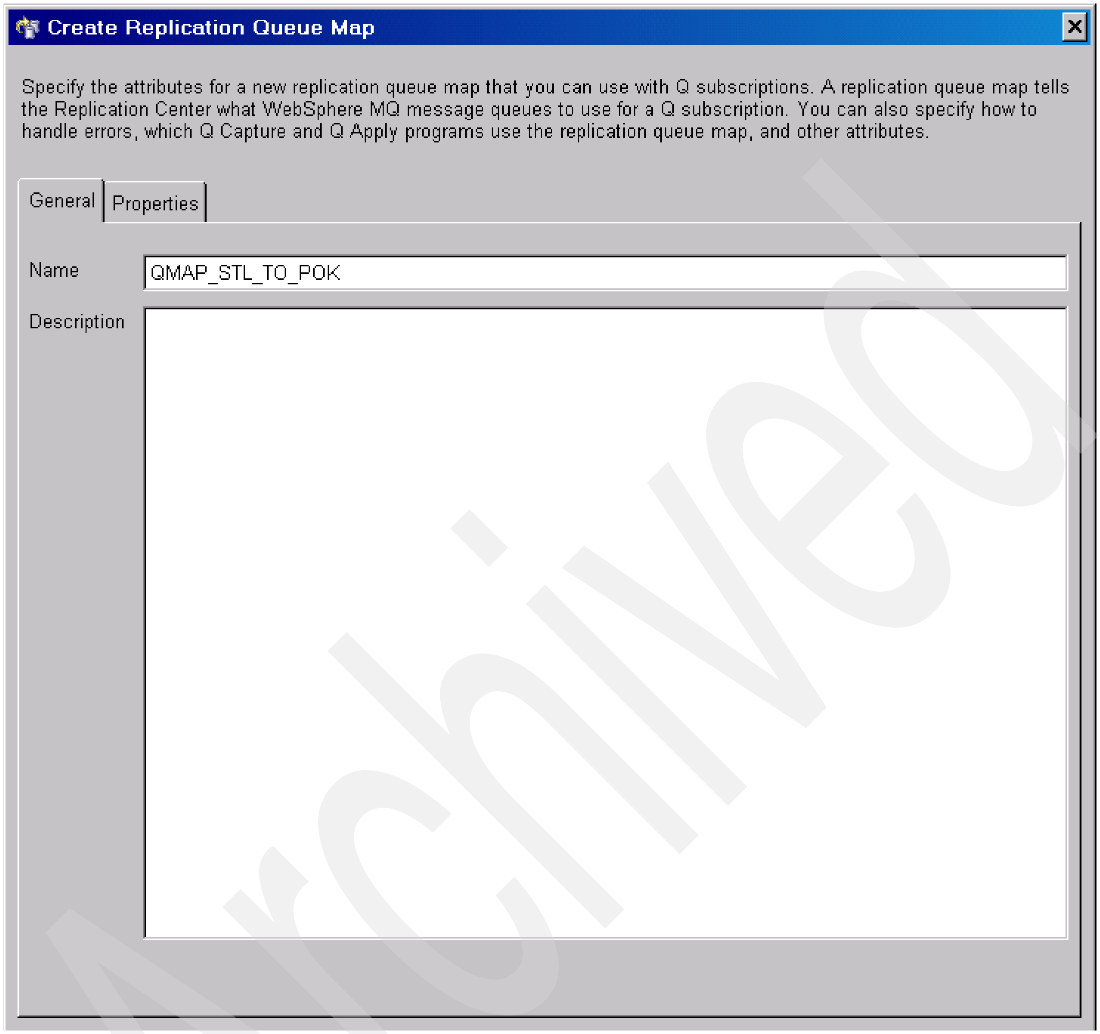


Figure 4-36 Create Replication Queue Map

Supply the name `QMAP_STL_TO_POK` (reference A.27 in Table 4-3 on page 304), and then choose the **Properties** tab to display Figure 4-37.

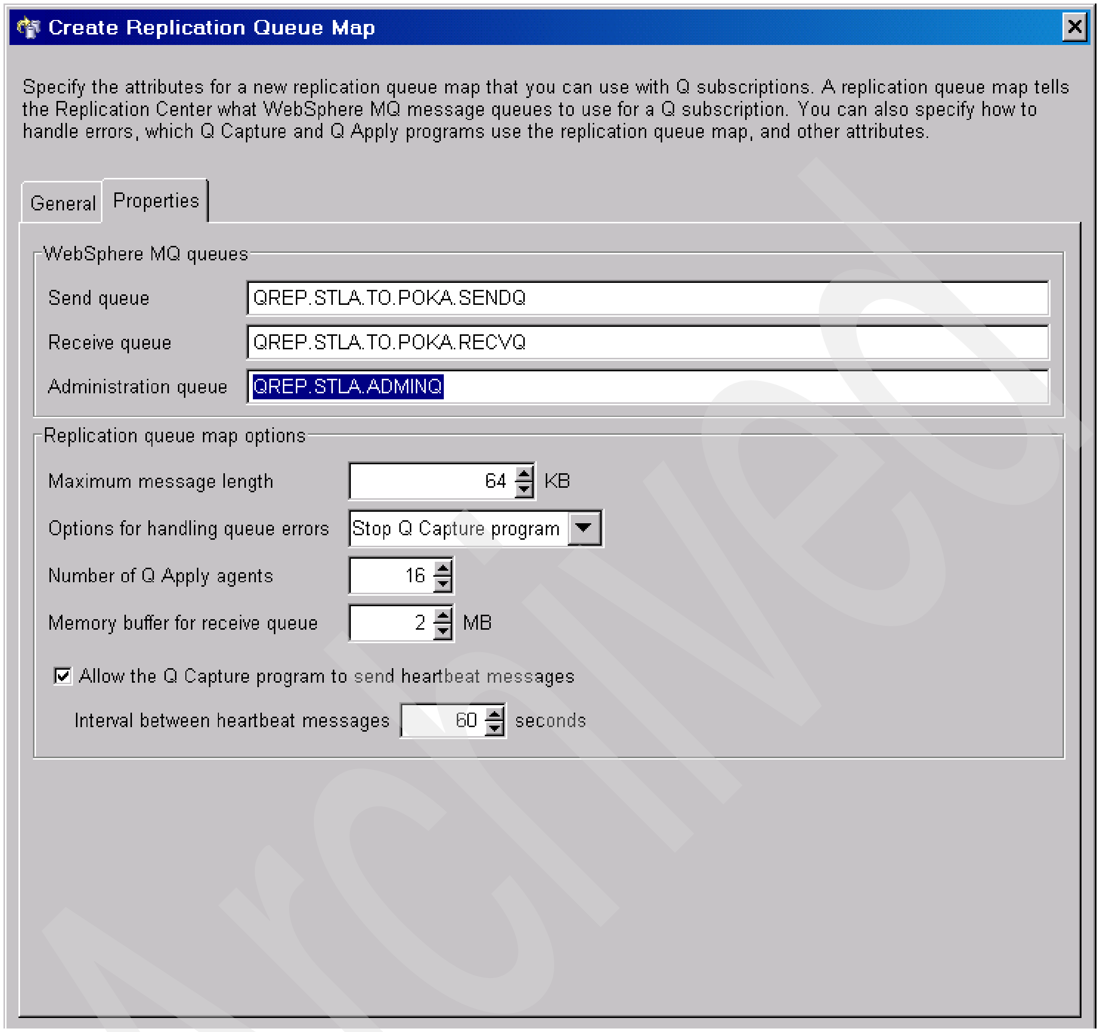


Figure 4-37 Create Replication Queue Map - Properties

Supply values for the Send queue (“QREP.STLA.TO.POKA.SENDQ” reference A.18 in Table 4-3 on page 304), Receive queue (“QREP.STLA.TO.POKA.RECVQ” reference B.19 in Table 4-3 on page 304), Administration queue (“QREP.STLA.ADMINQ” reference A.17 in Table 4-3 on page 304), and let other values default.

Click **OK** to complete the definition, and generate the scripts for creating the replication queue map objects on the servers, as shown in Figure 4-40.

Attention: In fixpak 9, which is due out in April 2005, the screens shown in Figure 4-36 and Figure 4-37 have been enhanced to appear as shown in Figure 4-38 and Figure 4-39, respectively. Please ignore the values shown therein, since the screens have been shown to only highlight the difference in the new screen layout.

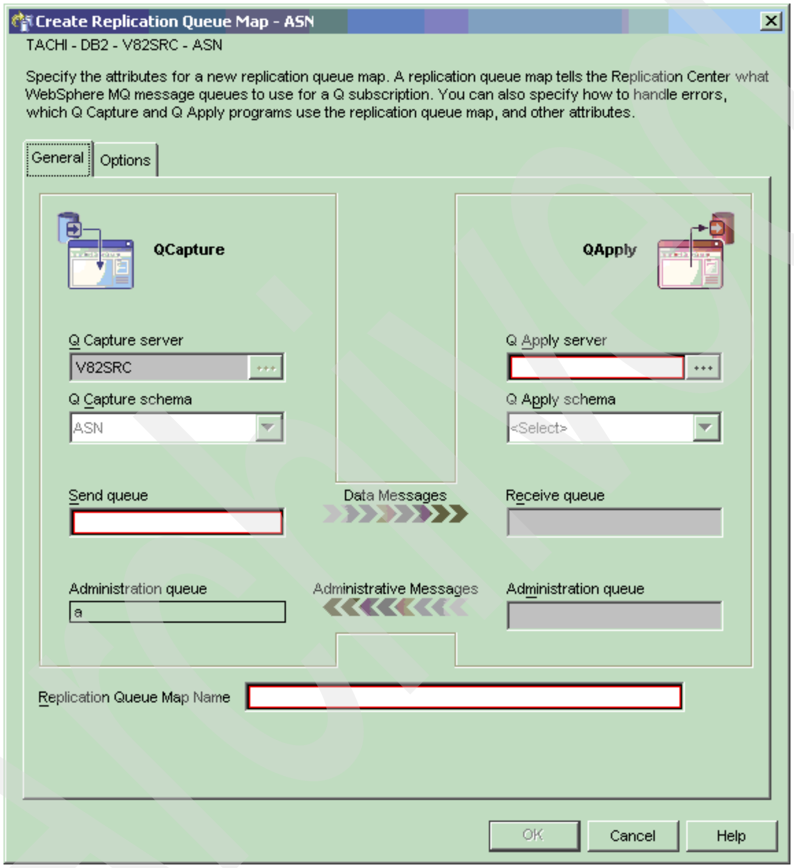


Figure 4-38 Fixpak 9 Create Replication Queue Map screen

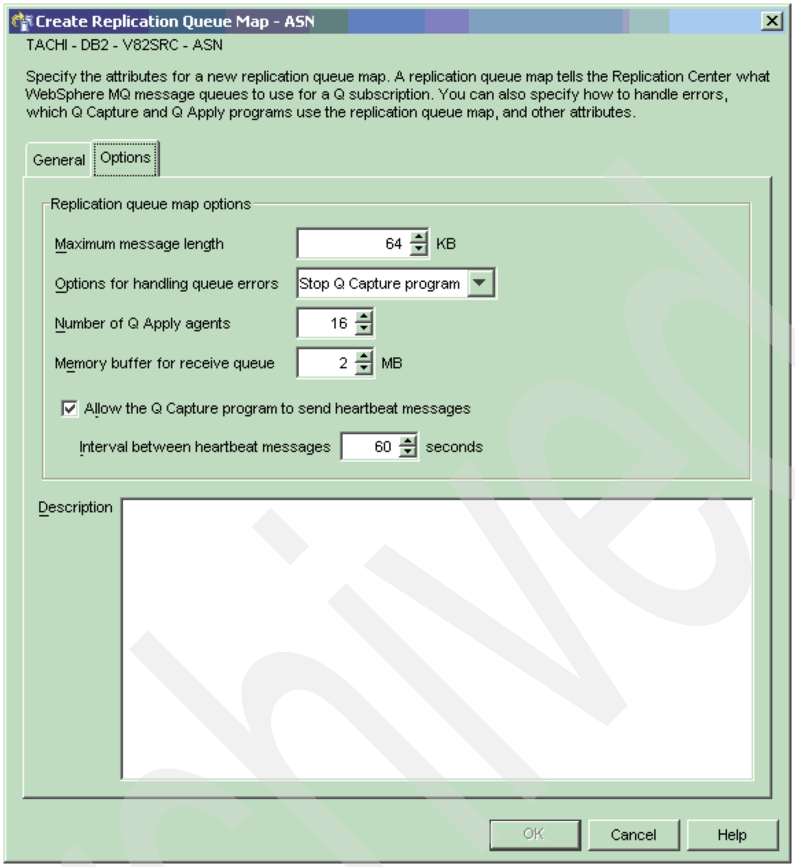


Figure 4-39 Fixpak 9 Create Replication Queue Map - Options screen

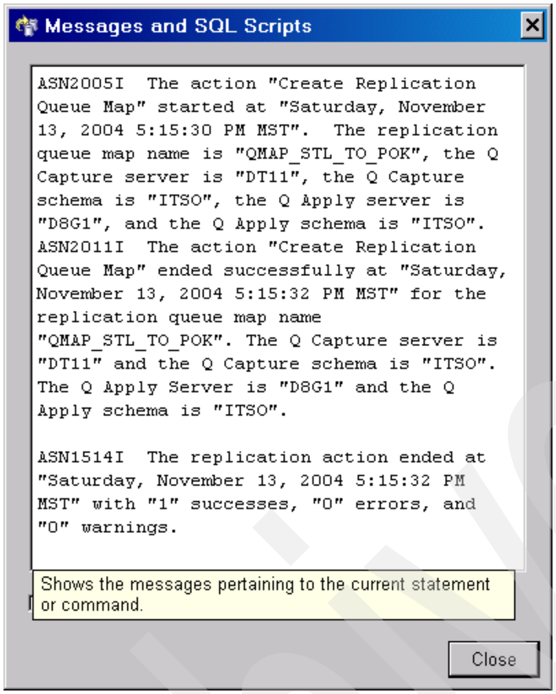


Figure 4-40 Messages and SQL Scripts

Click **Close** to bring up Figure 4-41 for choosing whether to execute the scripts right away or save the scripts for later execution.

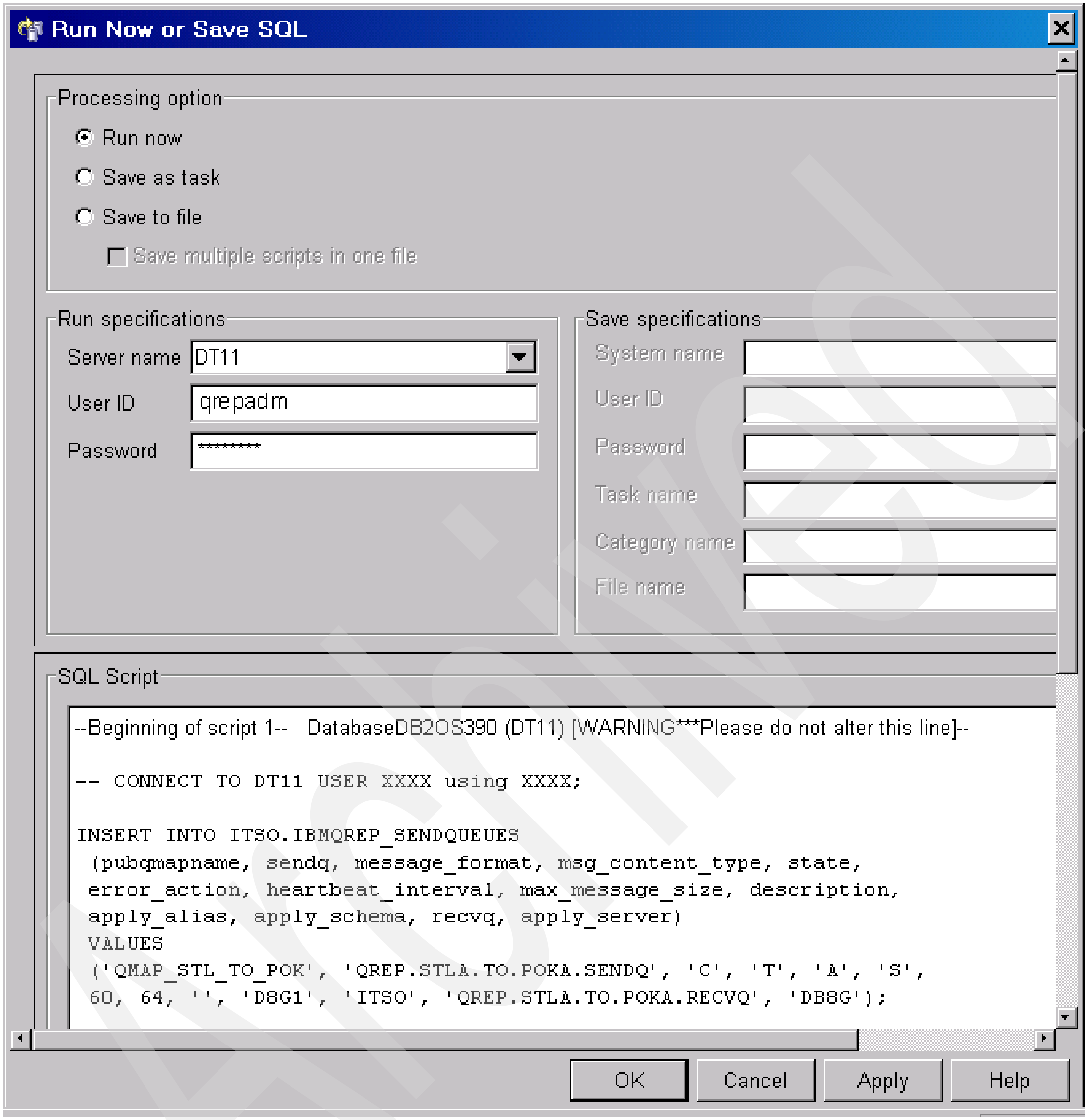


Figure 4-41 Run Now or Save SQL

Select the **Run now** option and click **OK** to execute the script immediately. Figure 4-42 is displayed when the scripts run successfully.

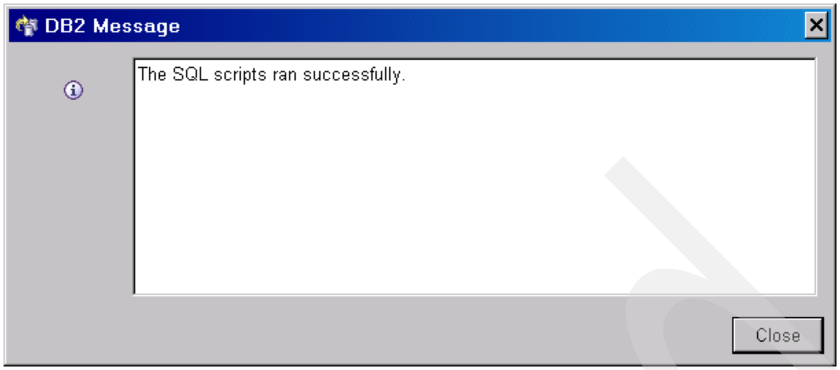


Figure 4-42   SQL scripts ran successfully message

Click **Close** to proceed to Figure 4-43, which displays the just created Replication Queue Map.

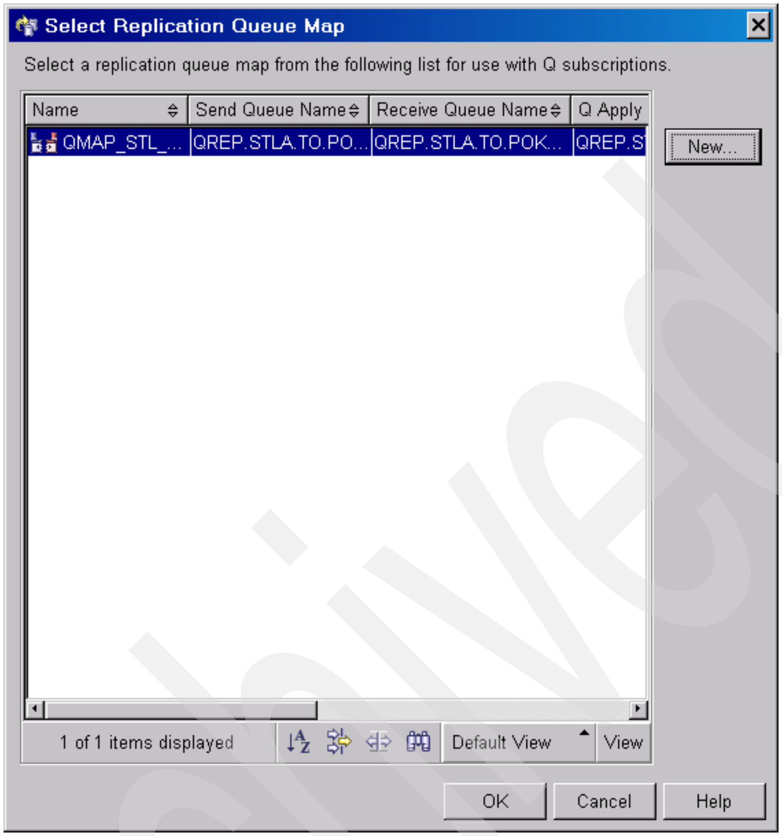


Figure 4-43 Select Replication Queue Map

Highlight the created replication queue map and click **OK** to proceed to Figure 4-44.

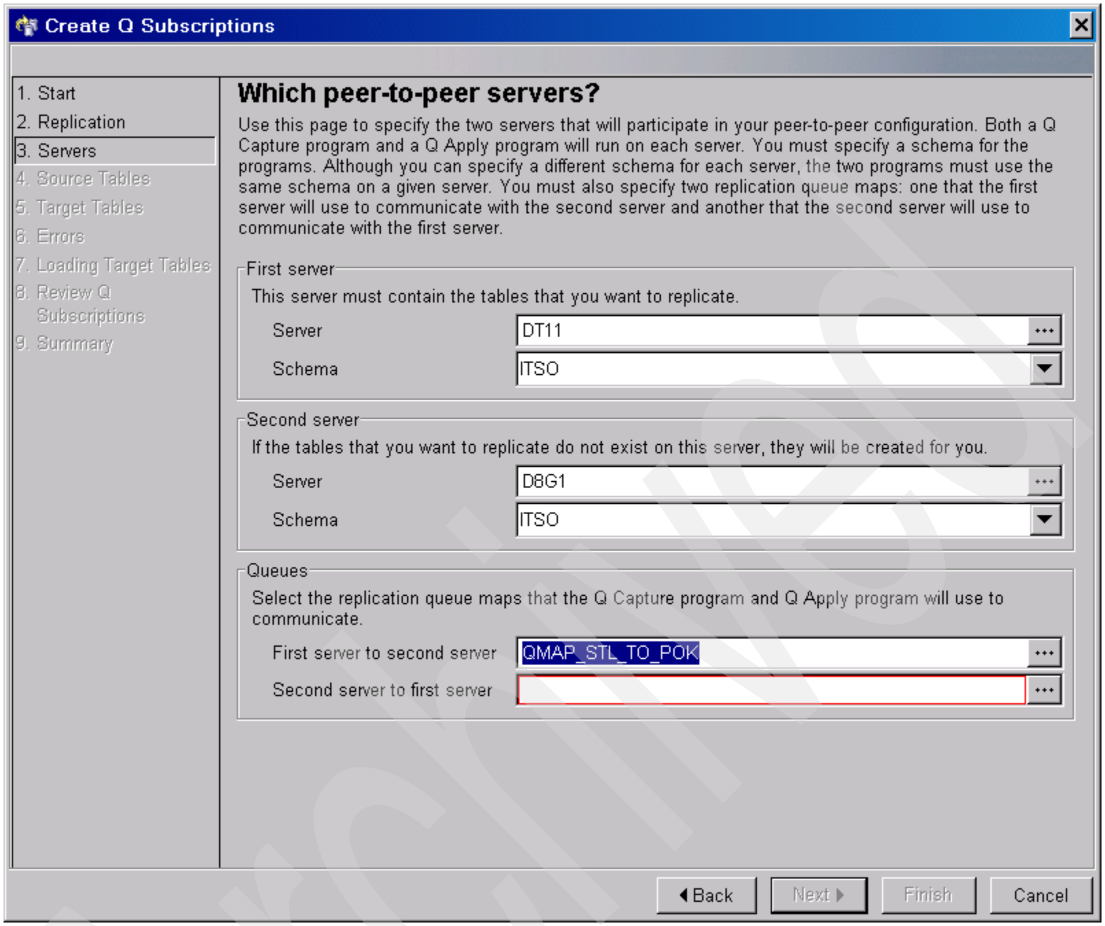


Figure 4-44 Which peer-to-peer servers

The next step is to select the replication queue map for communication between the “Second server to first server.”

Repeat the steps performed earlier for the replication queue map definition from the “First server to second server” (as described in Figure 4-34 on page 337 through Figure 4-43) using the appropriate template values corresponding to WTSC53 in Table 4-2 on page 302 and Table 4-3 on page 304. This is described in Figure 4-44 through Figure 4-51 on page 353.

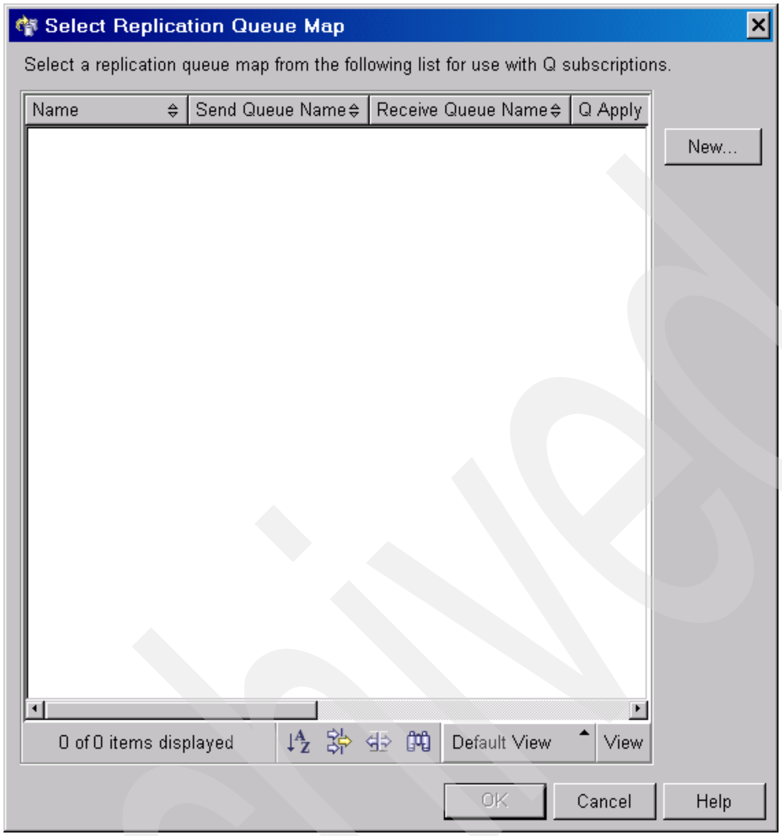


Figure 4-45 Select Replication Queue Map

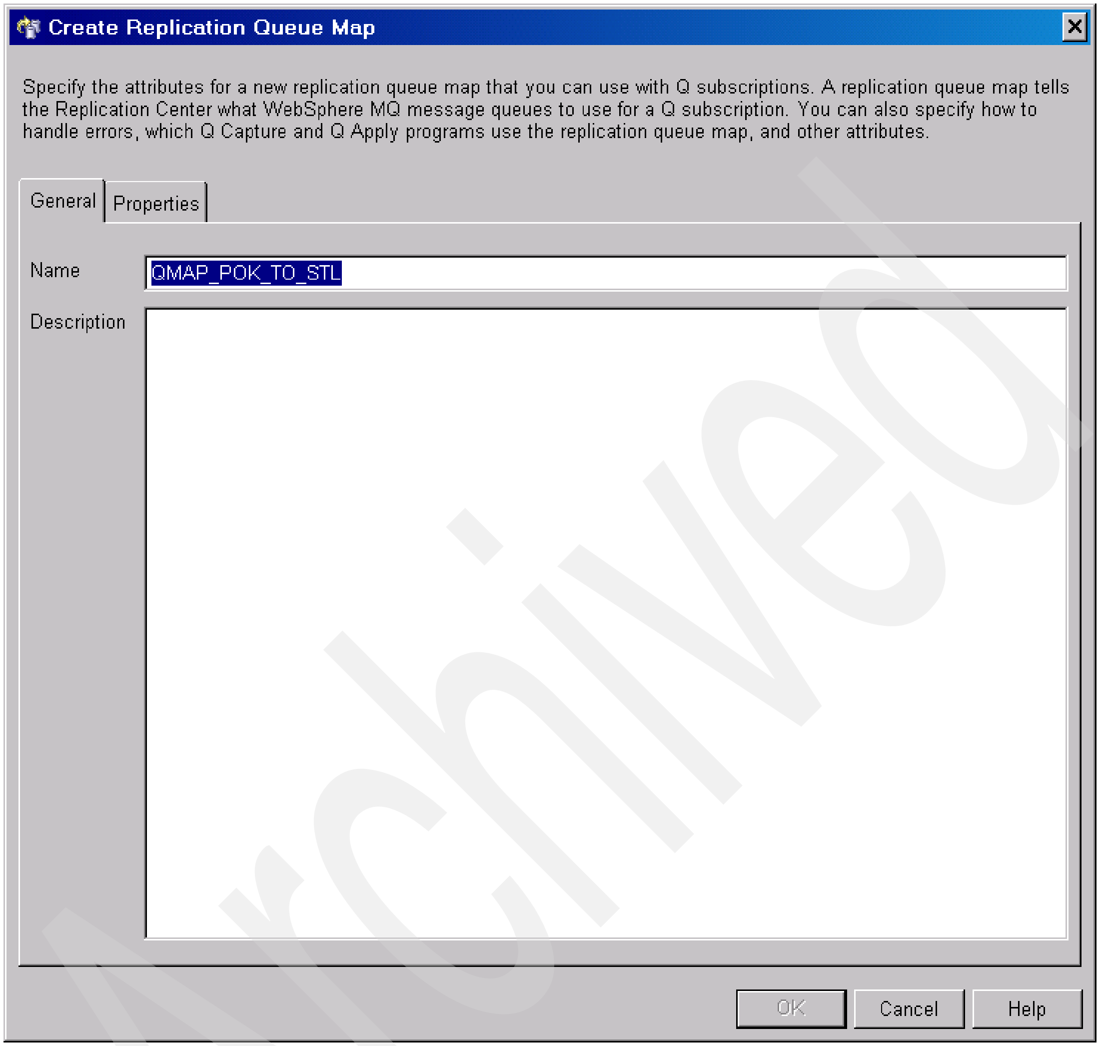


Figure 4-46 Create Replication Queue Map

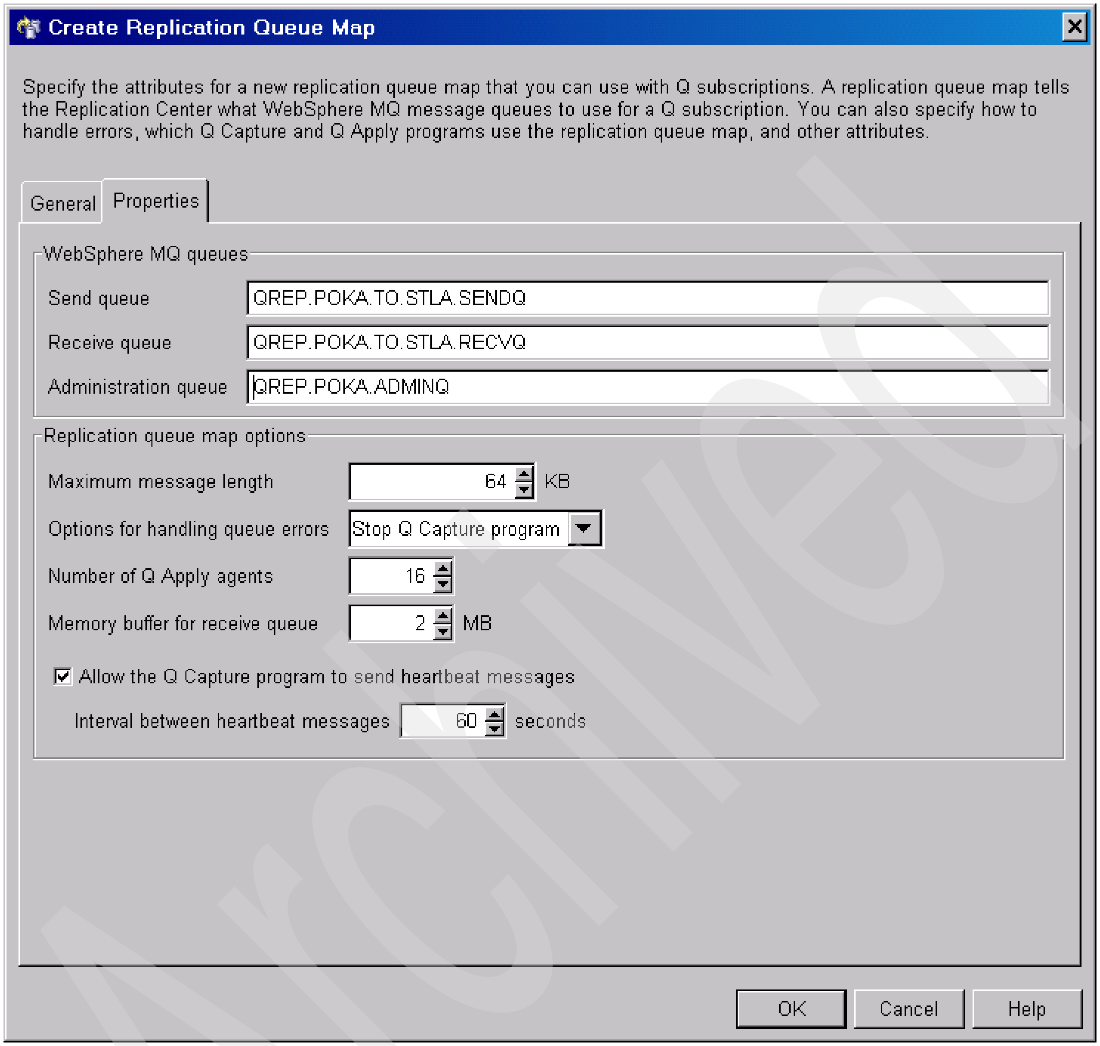


Figure 4-47 Create Replication Queue Map - Properties

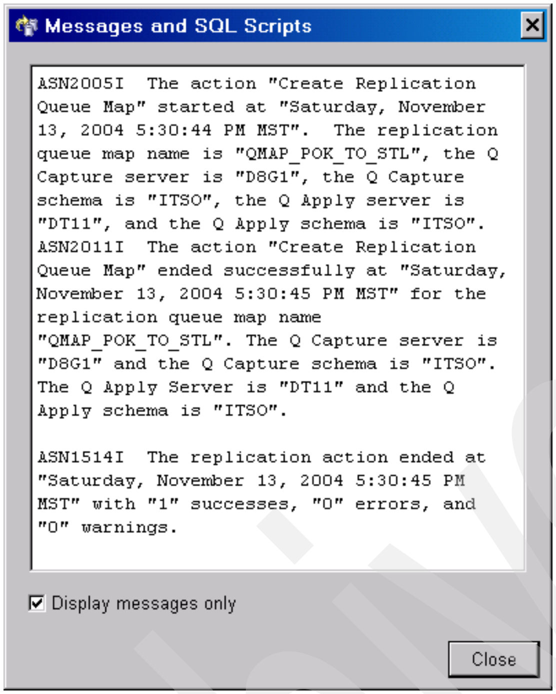


Figure 4-48 Messages and SQL Scrips

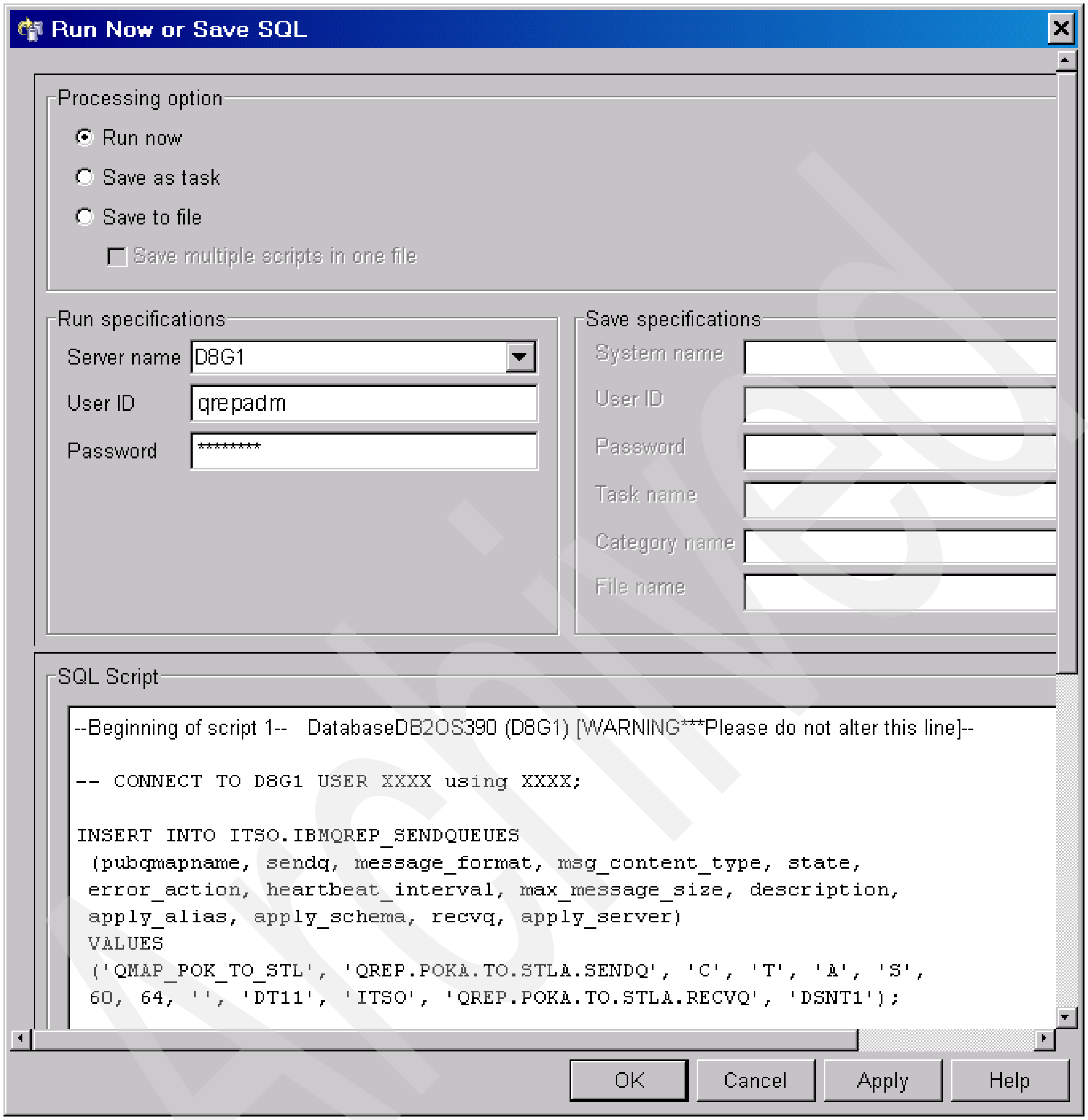


Figure 4-49 Run Now or Save SQL

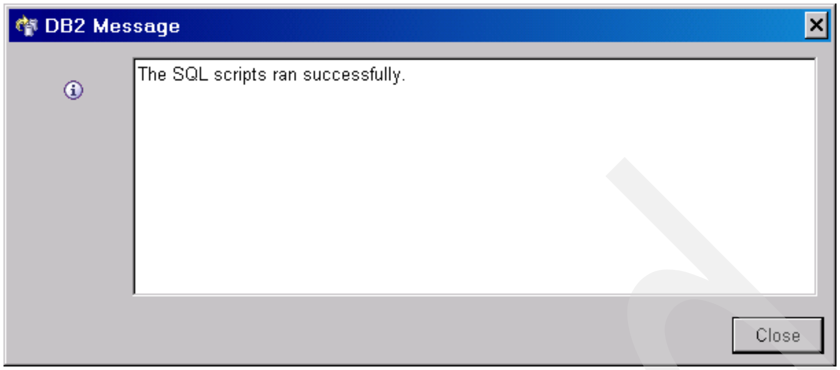


Figure 4-50 SQL scripts ran successfully message

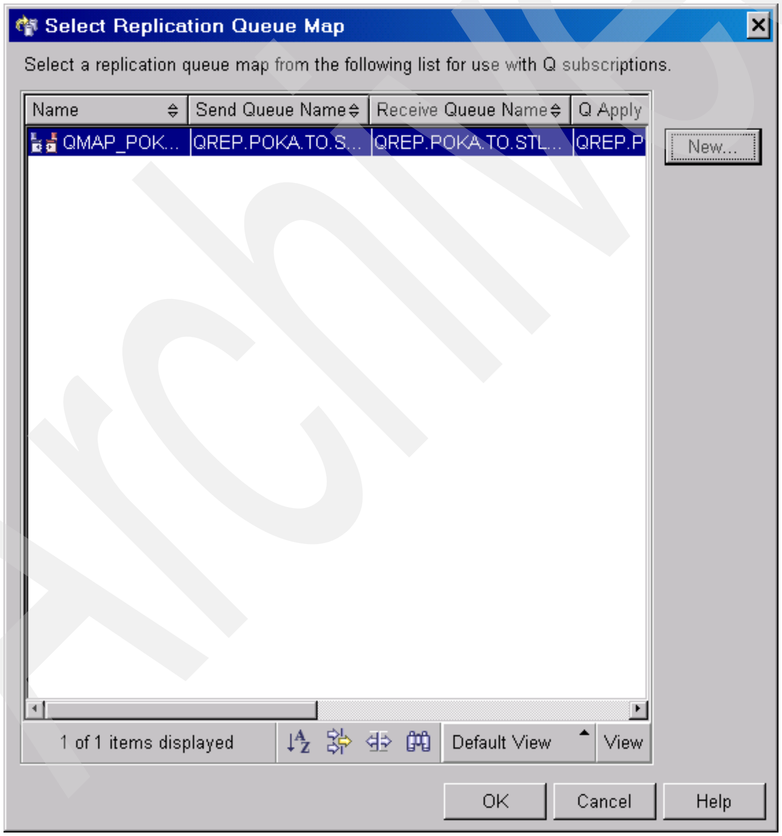


Figure 4-51 Select Replication Queue Map

At the completion of creating the replication queue maps for communication between the servers (see Figure 4-52), the individual subscriptions need to be defined.

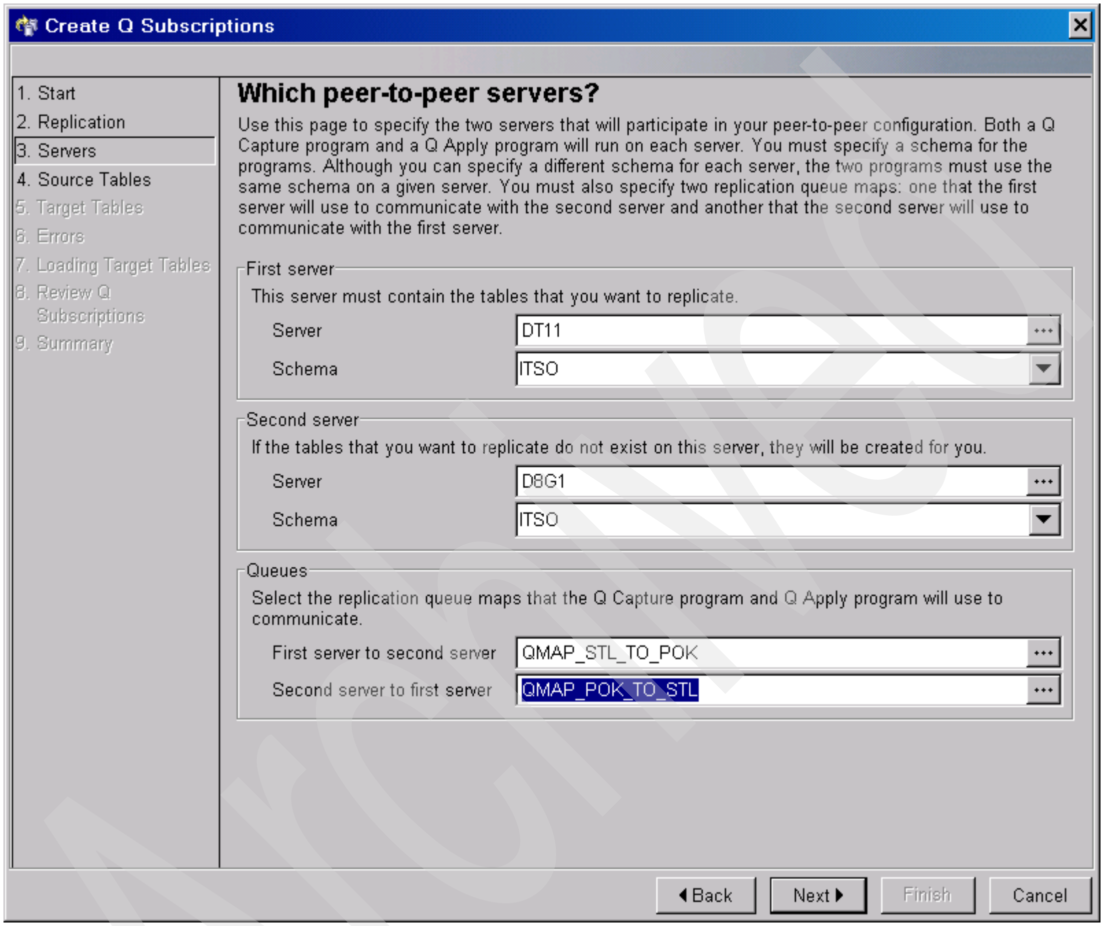


Figure 4-52 Completion of the replication queue maps selection

Click **Next** in Figure 4-52 to begin the process of defining the subscriptions, as shown in Figure 4-53.

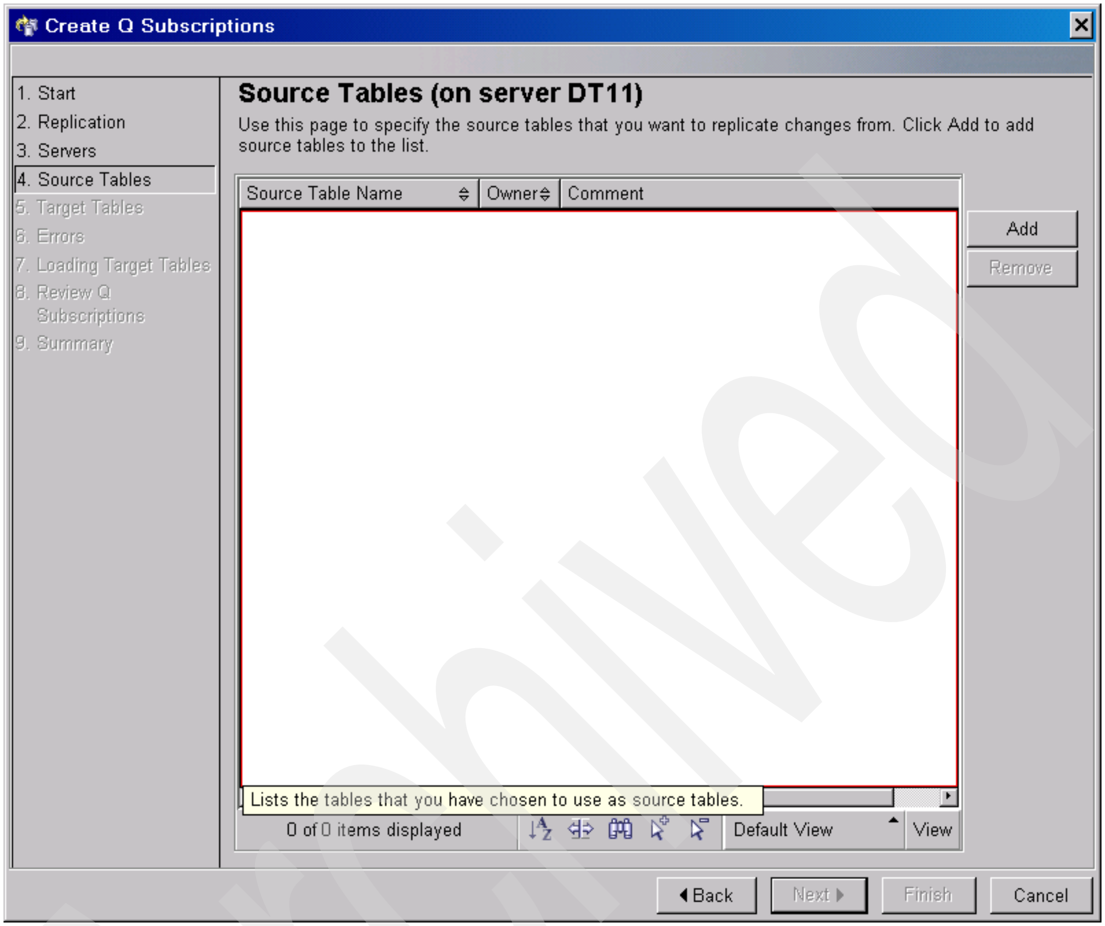


Figure 4-53 Create Q Subscriptions - Source Tables (on server DT11)

Since there are no tables to select from, click the **Add** button in Figure 4-53 to add to the list of tables available for subscriptions, as shown in Figure 4-54.

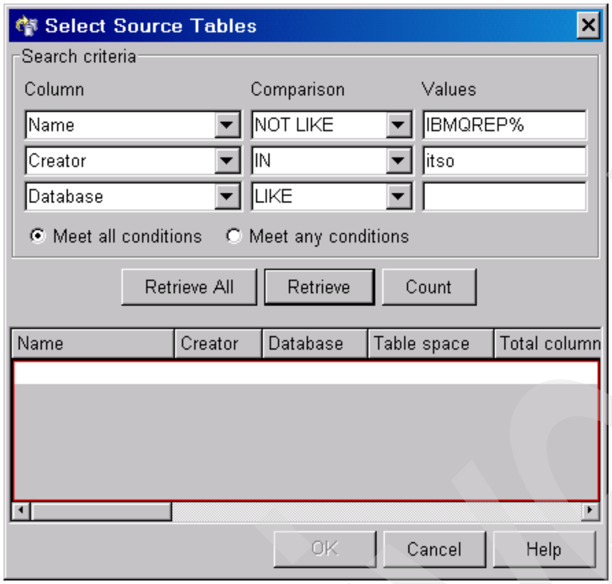


Figure 4-54 Select Source Tables

Exclude all tables that have a name of IBMQREP%, and include tables with a creator name of itso (the application tables for this configuration). Click **Retrieve** to list all the tables that qualify, as shown in Figure 4-55. Clicking **Retrieve All** ignores the criteria.

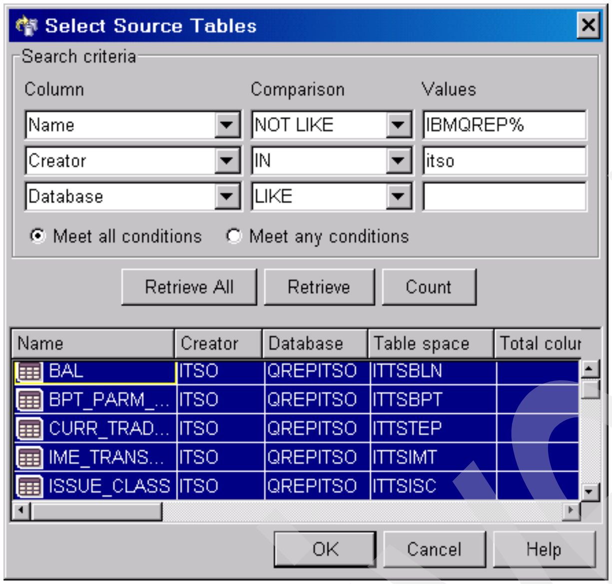


Figure 4-55 Select Source Tables

Go back and forth between Figure 4-55 and Figure 4-54 until all 23 tables in the application have been selected (23 tables in Table 4-3 on page 304). The complete list is shown in Figure 4-56.

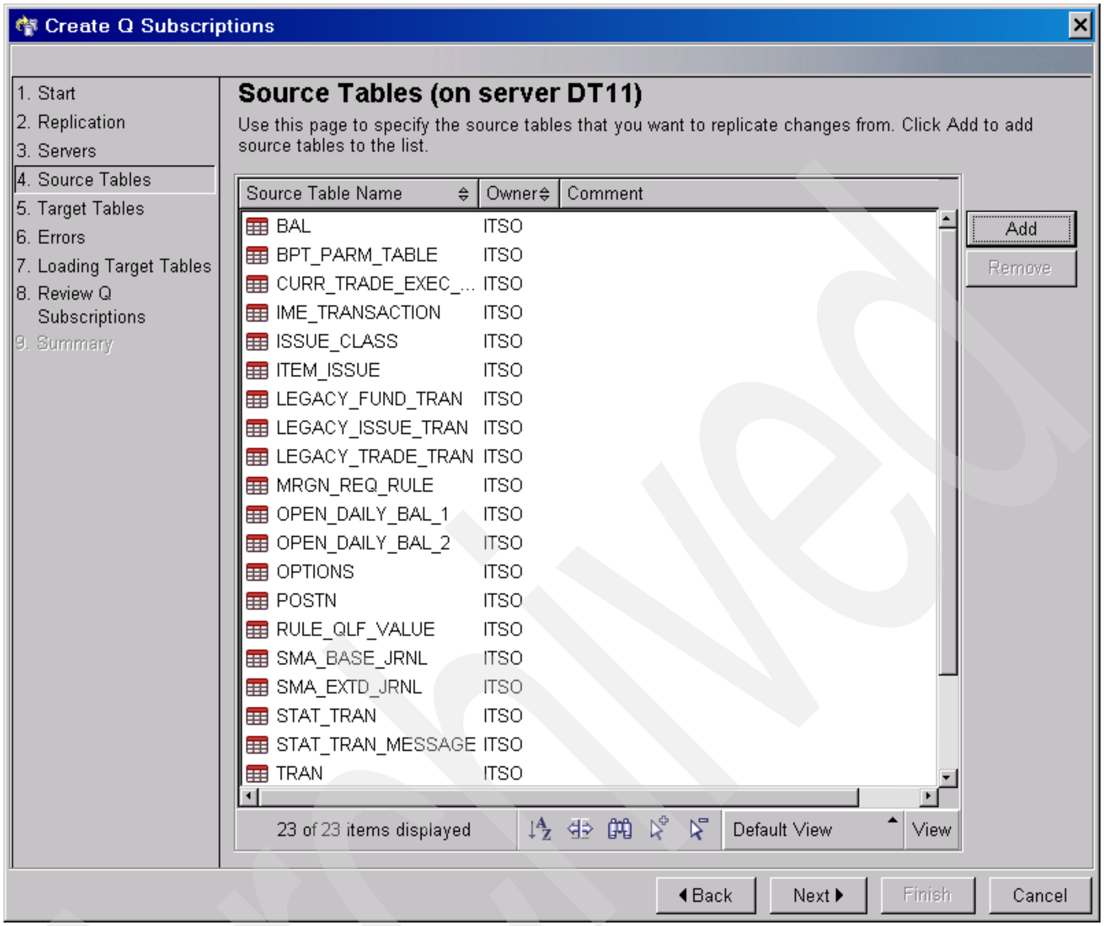


Figure 4-56 Source Tables

Click **Next** to set the profiles for the target tables, as shown in Figure 4-57.

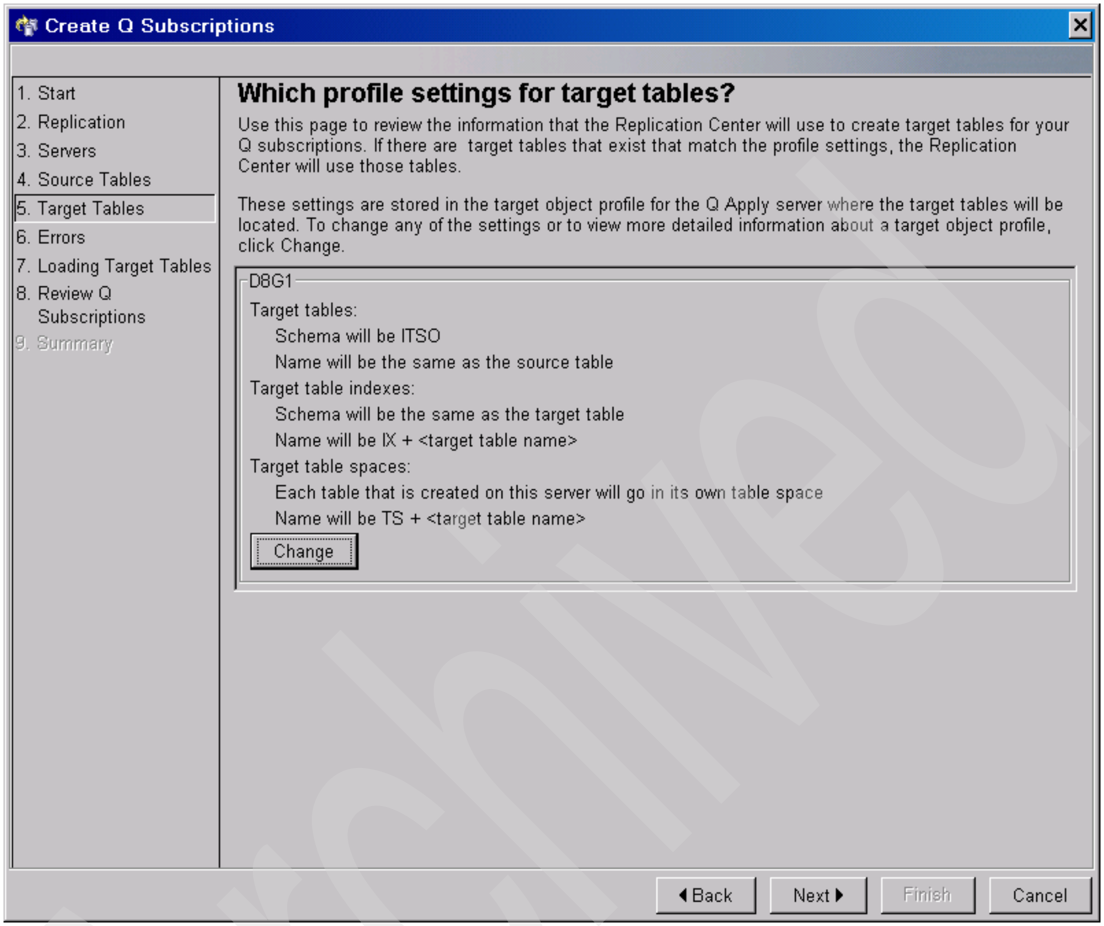


Figure 4-57 Create Q Subscriptions - Which profile settings for target tables

Figure 4-57 provides the option for changing the default settings for the target tables (reference A.28a, A.28b, and A.28c in Table 4-3 on page 304). We chose to go with the defaults. Click **Next** to specify the action to be taken in the event of errors for the peer-to-peer, as shown in Figure 4-58.

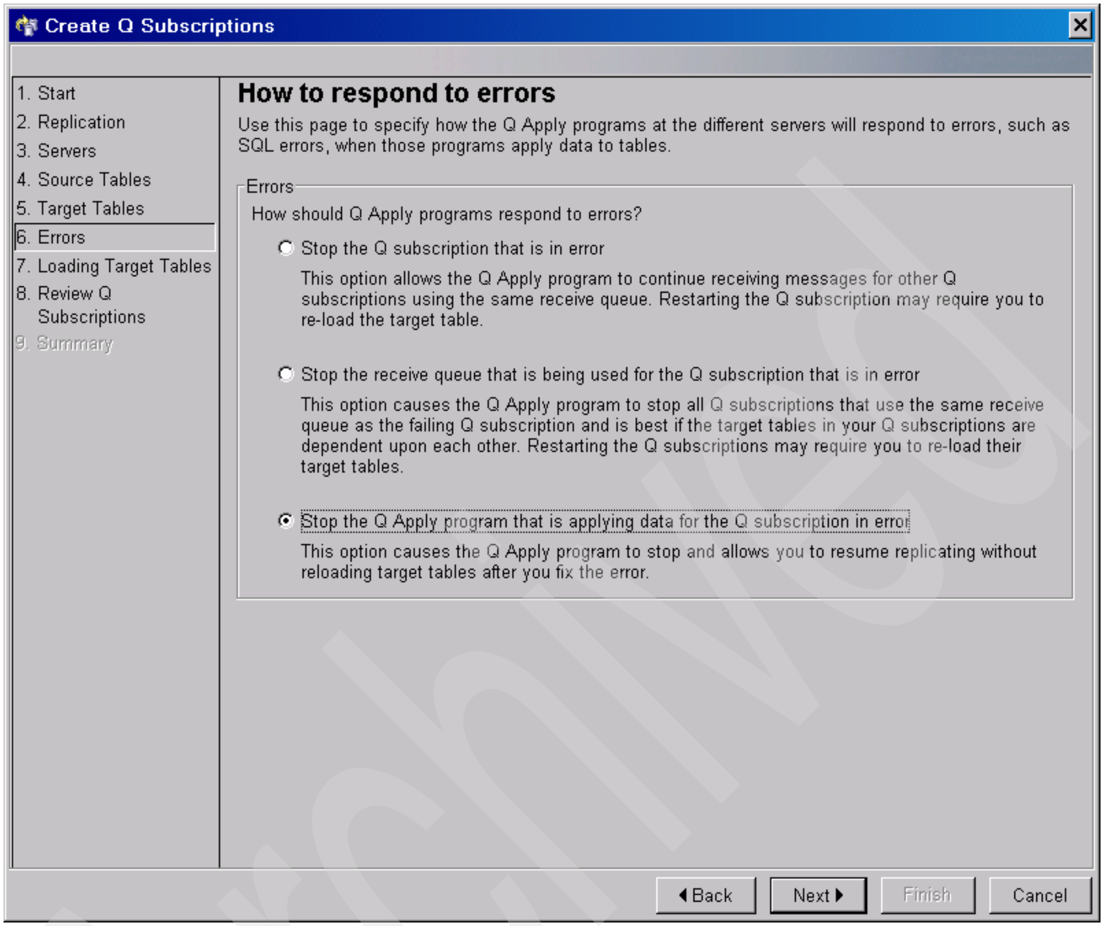


Figure 4-58 How to respond to errors

We selected the **Stop the Q Apply program that is applying data for the Q subscription in error** setting (reference A.28f in Table 4-3 on page 304).

We chose this option for the following reason:

- "Stop the Q subscription that is in error" would mean that Q Capture would continue through the DB2 log and ignore further changes to the table affected. This would force the reload/resynch of the table before the subscription can be reactivated, which can be very time consuming.
- "Stop the receive queue that is being used for the Q subscription that is in error" would stop all subscriptions that use this particular qmap. This would force the reload/resynch of *all* the tables in the replication queue map, which is undesirable.

- “Stop the Q Apply program that is applying data for the Q subscription in error” affects latency by introducing a delay; however, it permits correction without having to reload/resynch.

Click **Next** to specify how the target tables should be loaded in Figure 4-59.

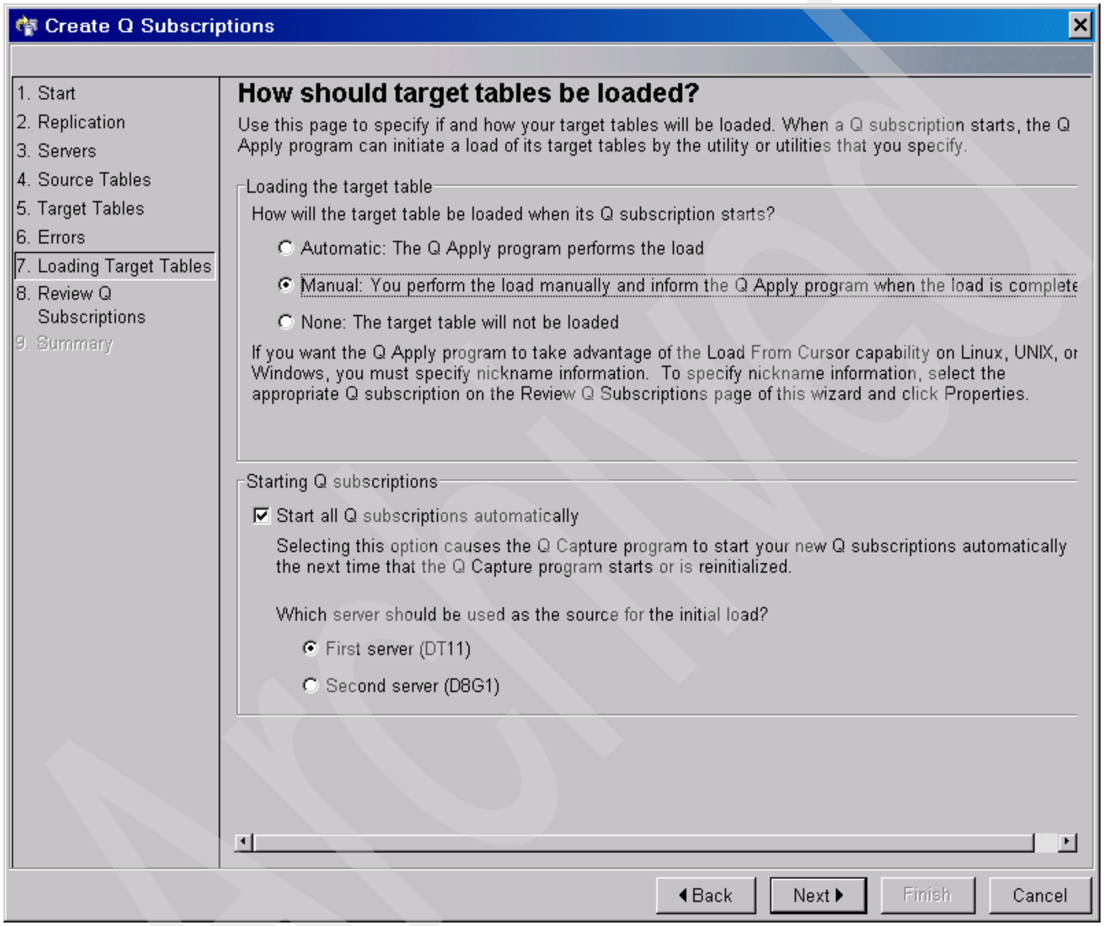


Figure 4-59 How should target tables be loaded

We chose the **Manual** option for this set of tables, indicating that we would be responsible for synchronizing the data in the target table with that of the source tables and sending a LOADDONE signal to the source Q Capture program when the load was completed. We selected the **Start all Q subscriptions automatically** option target tables. We also selected DT11 to be the source for the initial load, as shown in Figure 4-59.

Click **Next** to review the subscriptions defined.

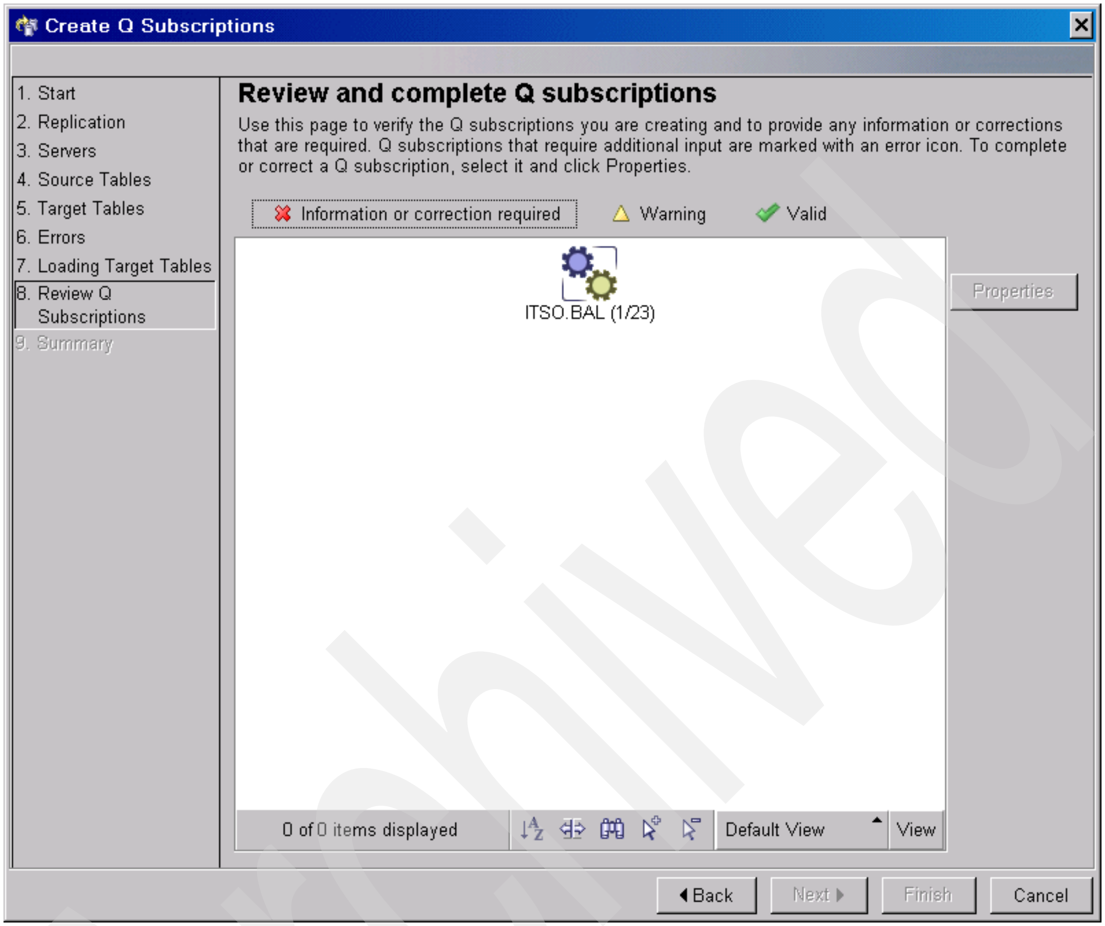


Figure 4-60 Review and complete Q subscriptions

Figure 4-60 is displayed while both z/OS databases are checked for accuracy of the replication request, and verified that the tables can be replicated. At the end of this Figure 4-61 is displayed showing the 46 subscriptions—two for each of the 23 tables.

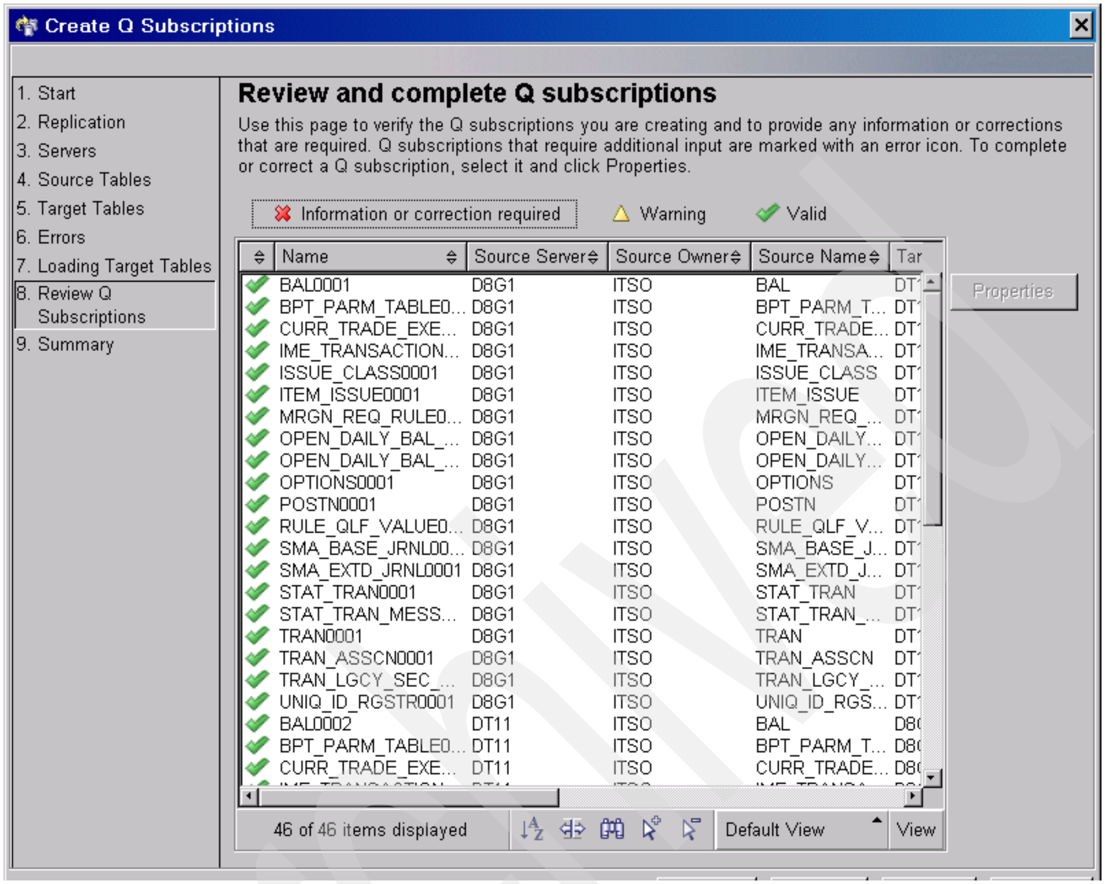


Figure 4-61 Review and complete Q subscriptions

Figure 4-61 shows 46 subscriptions in all—two for each of the 23 tables on each server. Click **Next** to complete the definition of the subscriptions and view the summary, as shown in Figure 4-62.

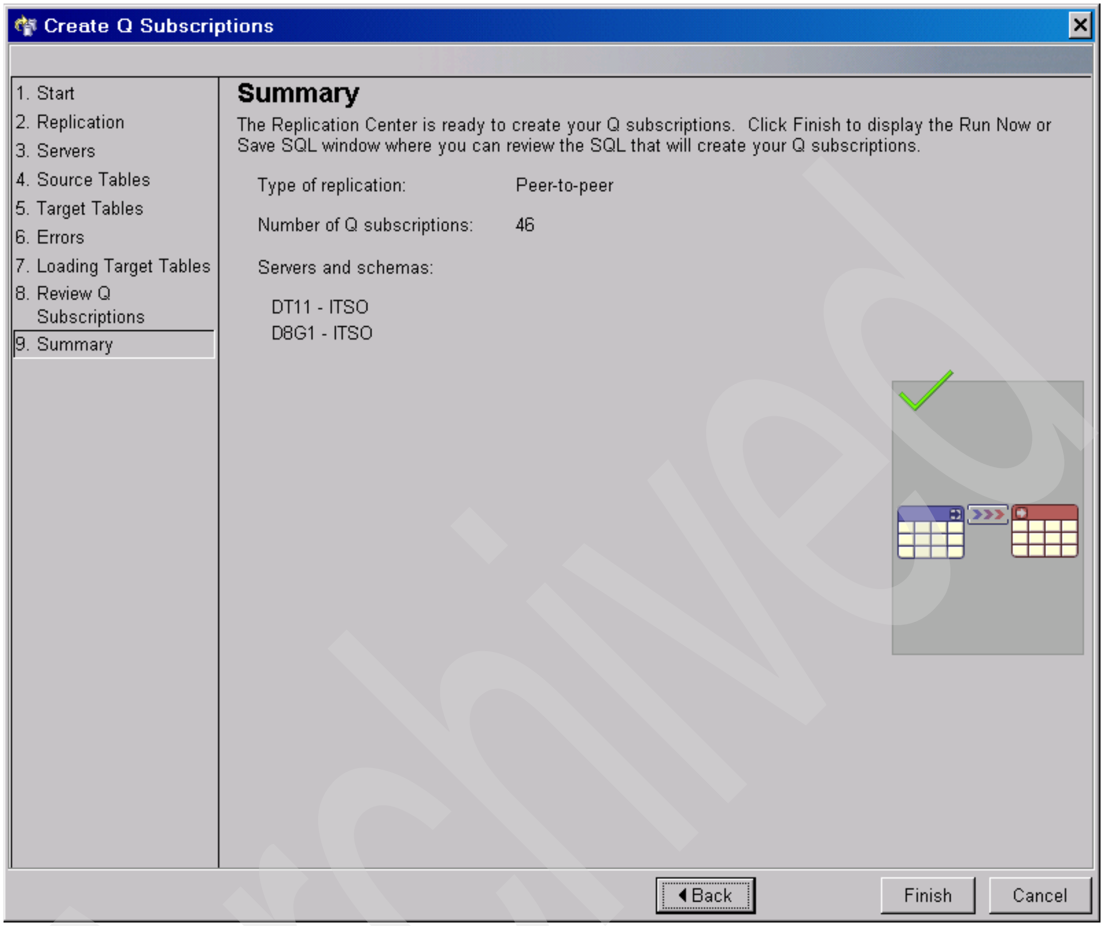


Figure 4-62 Create Q Subscriptions - Summary

Review the summary in Figure 4-62 and click **Finish** to generate the scripts for creating the various objects on the two servers, as shown in Figure 4-63.

Click the **Back** button to go back and correct any errors.

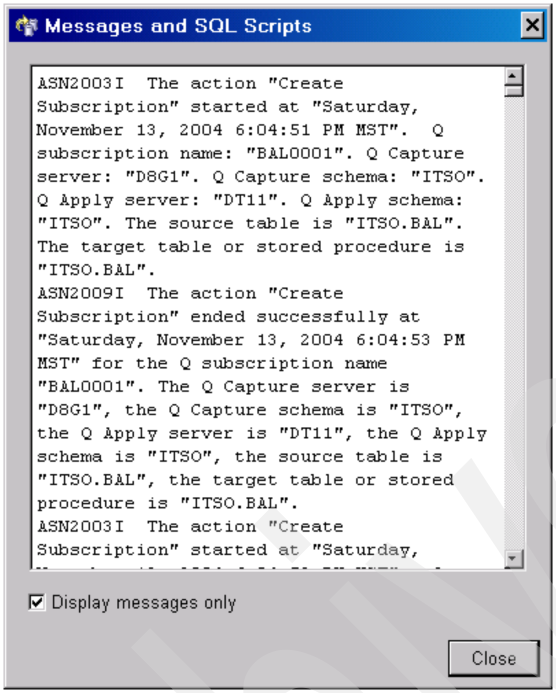


Figure 4-63 Messages and SQL Scripts

Click **Close** to decide whether to run the scripts right away or save it for later execution, as shown in Figure 4-64.

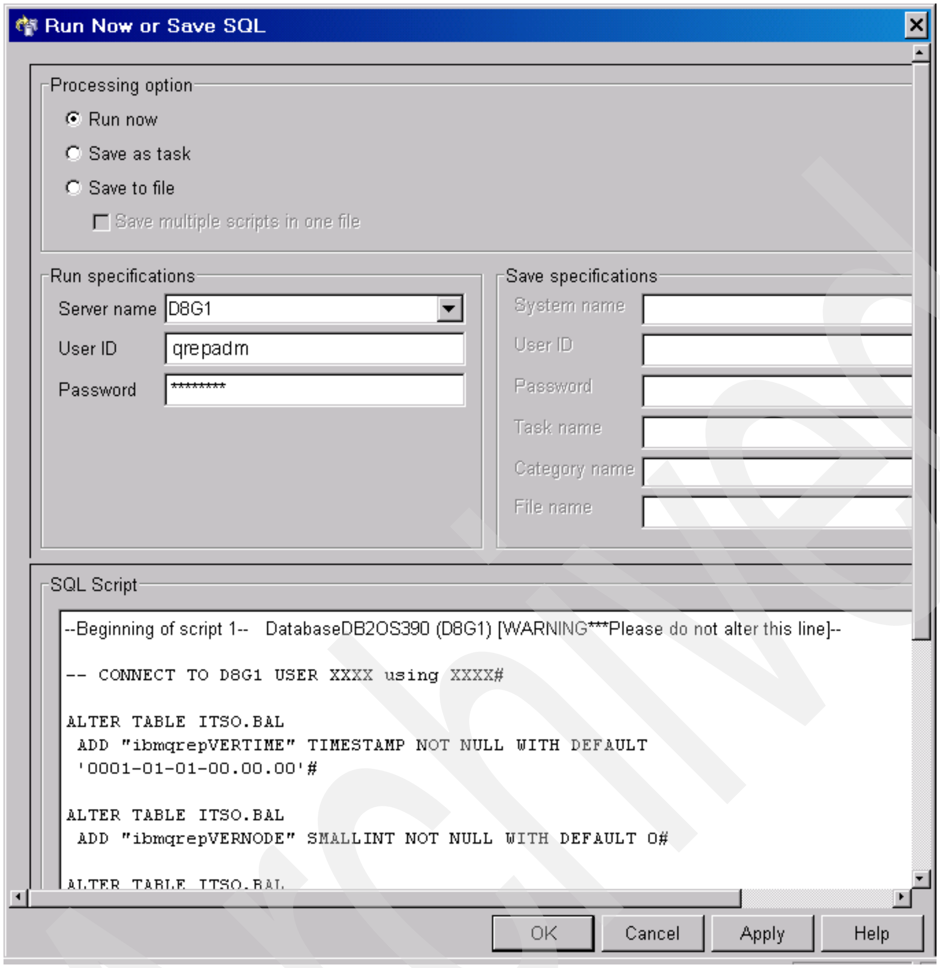


Figure 4-64 Run Now or Save SQL

We selected **Run now** and clicked **OK** to execute the scripts immediately. Figure 4-65 shows that the scripts ran successfully.

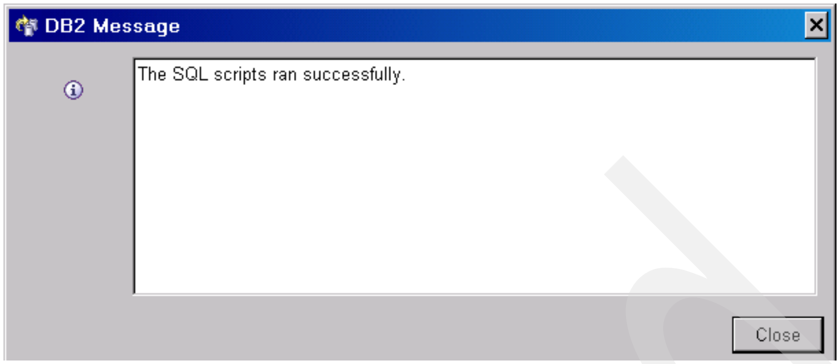


Figure 4-65 SQL scripts ran successfully message

With the completion of the subscriptions, the Q Capture program needs to be started on both servers.

> **Attention:** Q Capture should be started before Q Apply to ensure that all messages needed for activating subscriptions are processed correctly.

Step 6f: Start the Q Capture on both servers

This process is identical to “Step 6f: Start the Q Capture on both servers” on page 173.

The Q Apply program next needs to be started on both servers.

Step 6g: Start the Q Apply on both servers

This process is identical to “Step 6g: Start the Q Apply on both servers” on page 176.

Step 6h: Verify status of Q Capture and Q Apply processes

Our peer-to-peer configuration requested that a manual load (see Figure 4-59 on page 361, and reference A.28g in Table 4-3 on page 304) be performed when the subscriptions were activated from the DT11 source (reference A.28h in Table 4-3 on page 304).

When the DT11 capture task is started, it sees that there are new subscriptions to be activated, and sends Q subscription activation information to the apply task running on the D8G1 server. The D8G1 apply task then waits for the load to be completed (waiting on the receipt of the LOADDONE signal). Updates occurring during the load interval are received at D8G1 and spilled over into SPILLQ queues; these updates get applied subsequently when the LOADDONE signal is received.

This process is highlighted in the following screens.

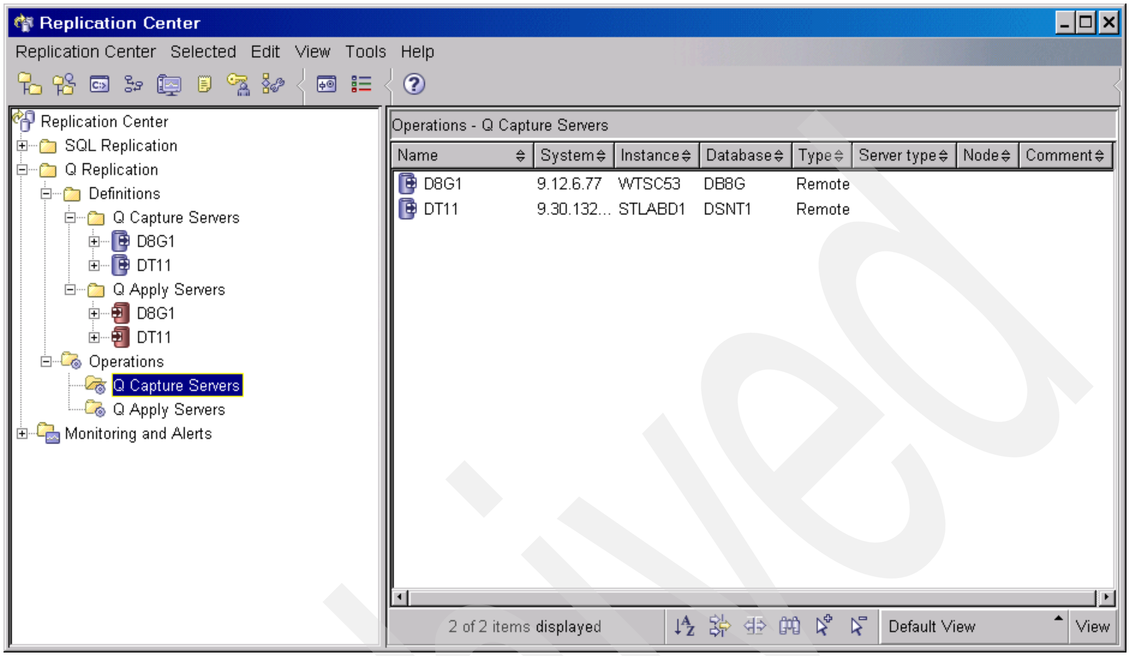


Figure 4-66 Replication Center menu

Expand the **Operations** folder in Figure 4-66 and highlight **Q Capture Servers** to display the Q Capture servers in the right-hand pane.

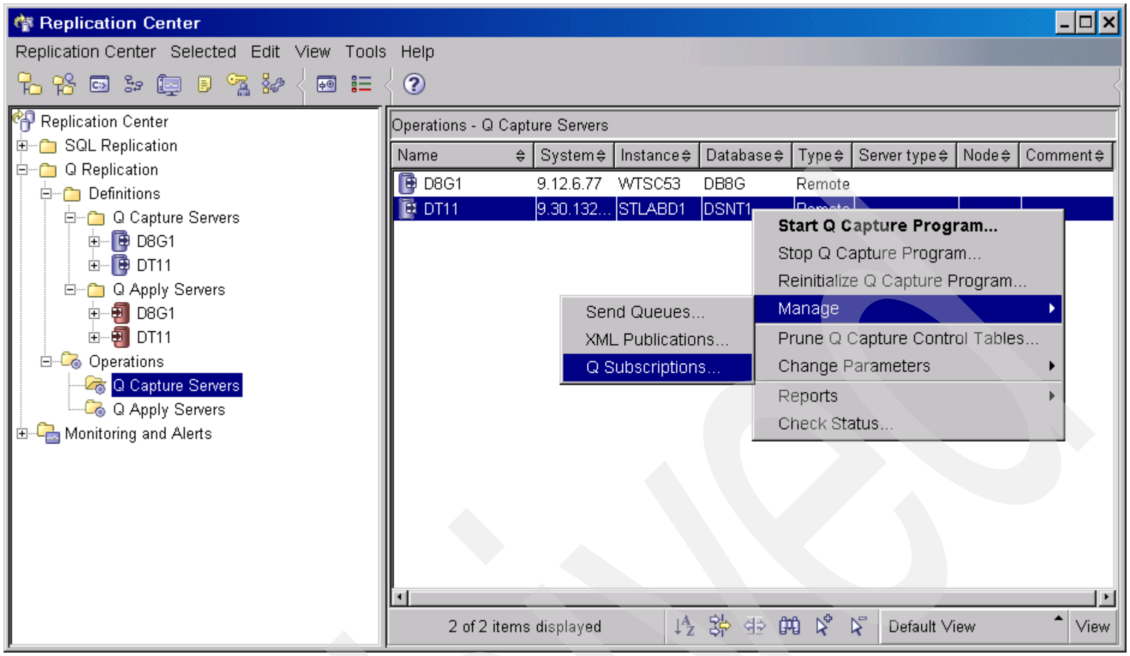


Figure 4-67 Replication Center - Manage Q Subscriptions

In Figure 4-67 highlight the **DT11** server and right-click it, and select **Manage** and **Q Subscriptions** to display Figure 4-68.

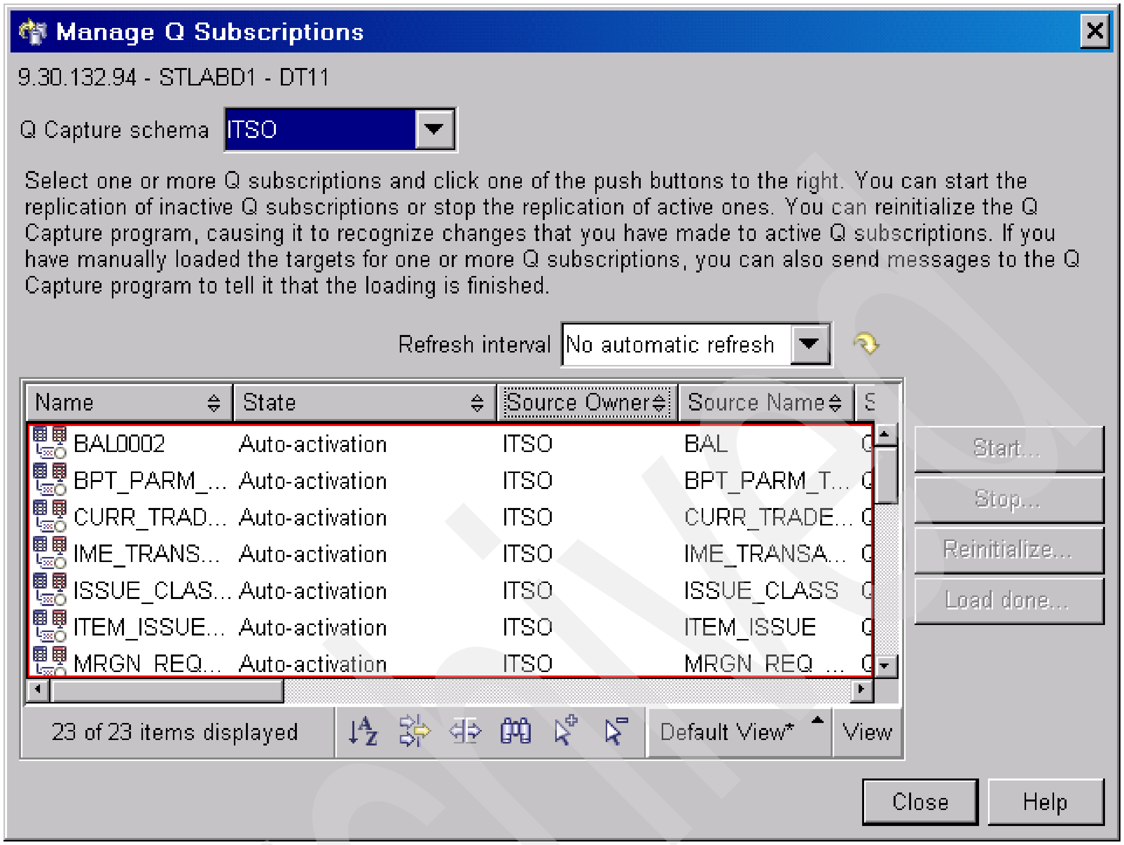


Figure 4-68 List of subscriptions (DT11 server)

Figure 4-68 lists the states of the various subscriptions. The Refresh interval field and the yellow arrow enable one to obtain the updated status of the subscription. Click **Close** to return to the previous screen.

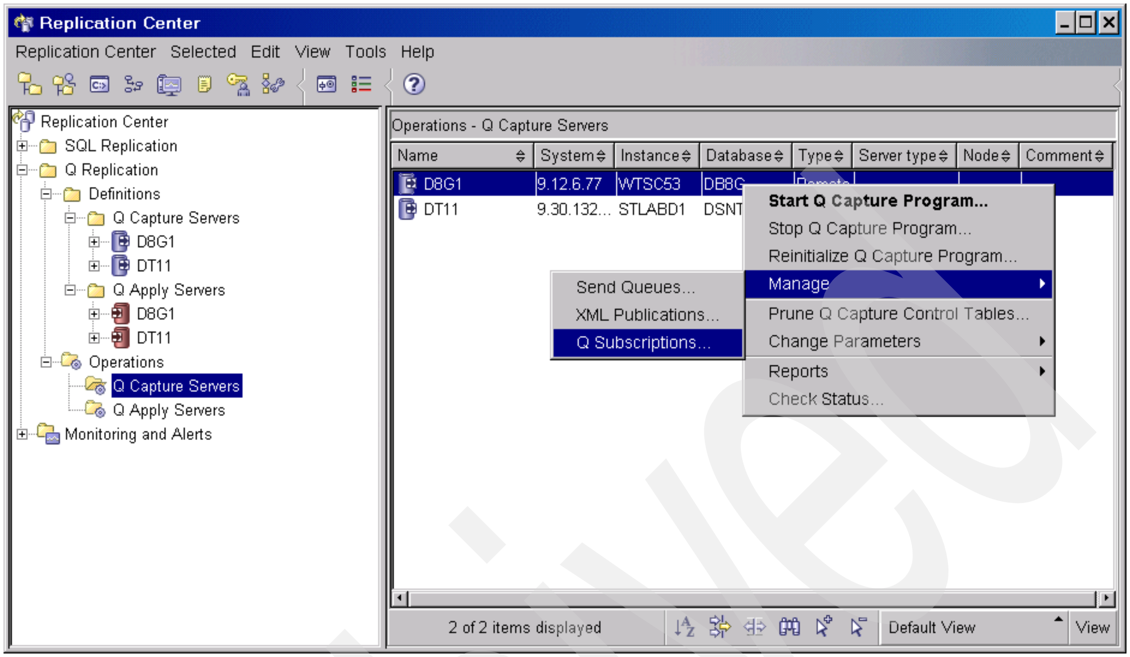


Figure 4-69 Replication Center - Manage Q Subscriptions

In Figure 4-69 highlight the **D8G1** server and right-click it, and select **Manage** and **Q Subscriptions** to display Figure 4-68.

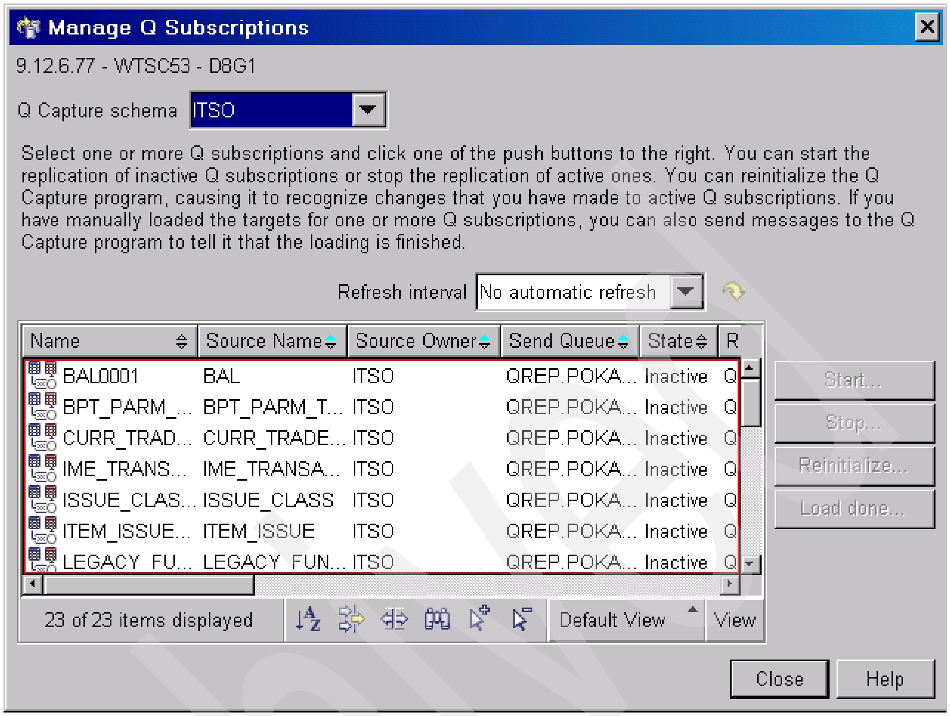


Figure 4-70 List of subscriptions (D8G1 server)

In Figure 4-70, the inactive state of the subscriptions in the target server D8G1 remains until the tables are loaded and any messages in the spill queue are emptied. Click **Close** and bring up the Manage Q Subscriptions selection for the source server DT11, as shown in Figure 4-71.

Next start the Q Capture and Q Apply tasks on the two servers DT11 and D8G1.

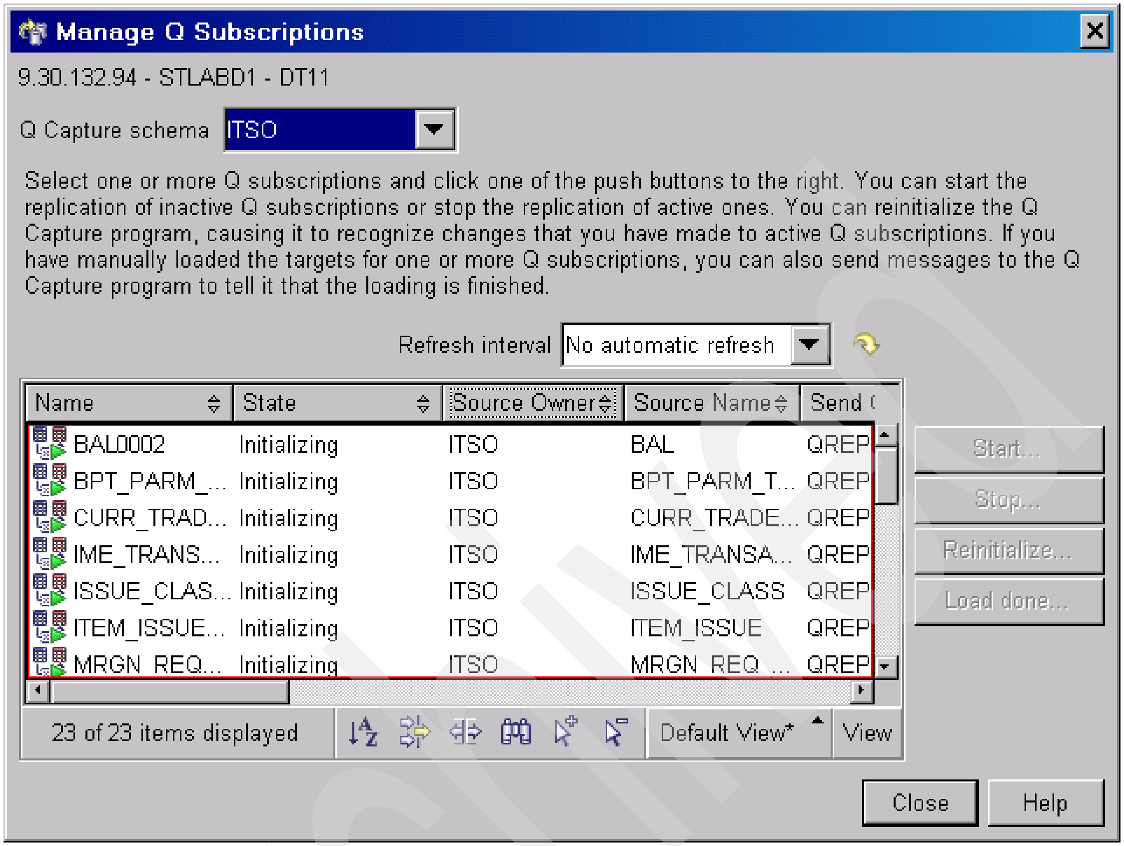


Figure 4-71 List of subscriptions (DT11 server)

In Figure 4-71, the initializing state for the subscriptions is indicated. Once the initialization process has started, the apply task sees the manual load request and signals the source of the load (DT11 server) that a manual load is required.

Figure 4-72 displays the screen indicating that a manual load is required.

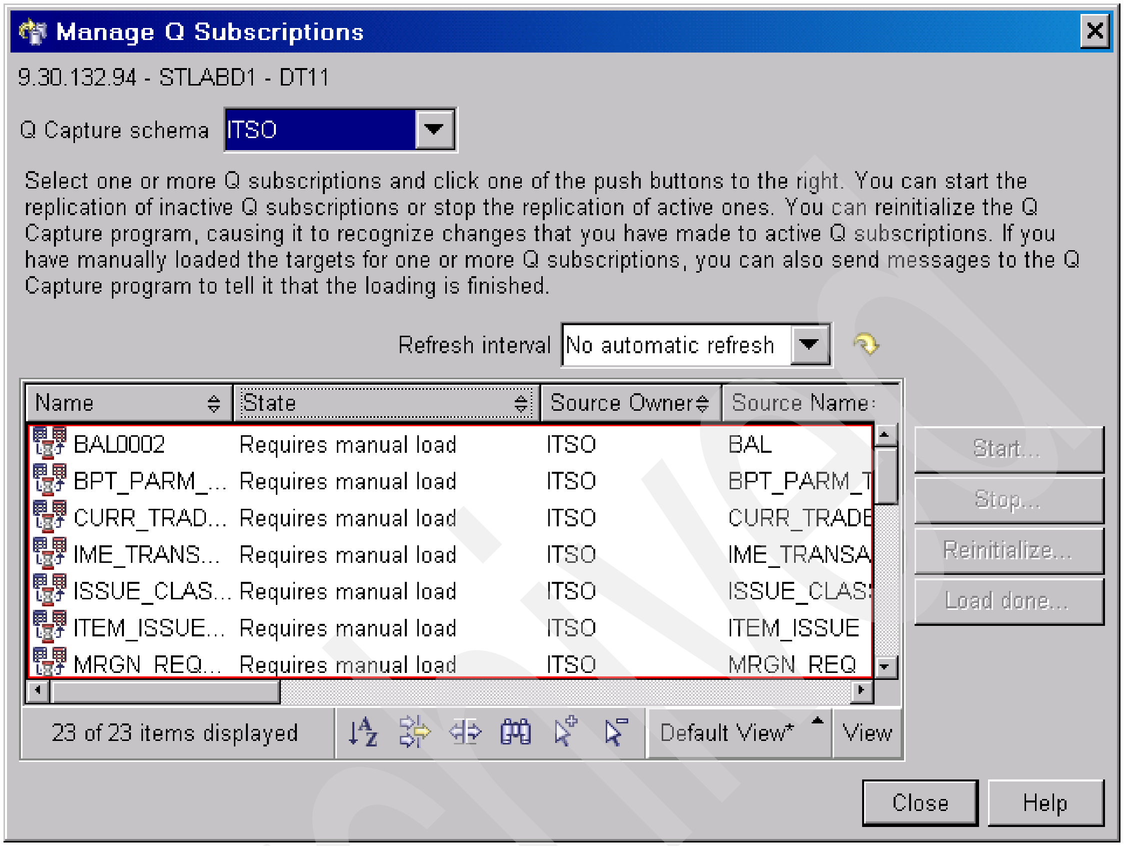


Figure 4-72 Requires manual load screen

The manual load can now begin. During this load phase, any updates to tables that are loading are placed in spill queues on the target server (D8G1). The messages in the spill queue are applied after that target server (D8G1) receives the LOADDONE signal.

The capture task and its log file receives the message shown in Example 4-1 when a subscription's load can begin.

Example 4-1 Capture log contents

```
11.03.26 STC01344  ASN7017I  "Q Capture" : "ITSO" : "WorkerThread" : The target
                   table "ITSO.BAL" is ready to be loaded from source table
                   "ITSO.BAL" for XML publication or Q subscription "BAL0002".
```

Figure 4-73 displays the contents of spill queues via MVS ISPF panels.

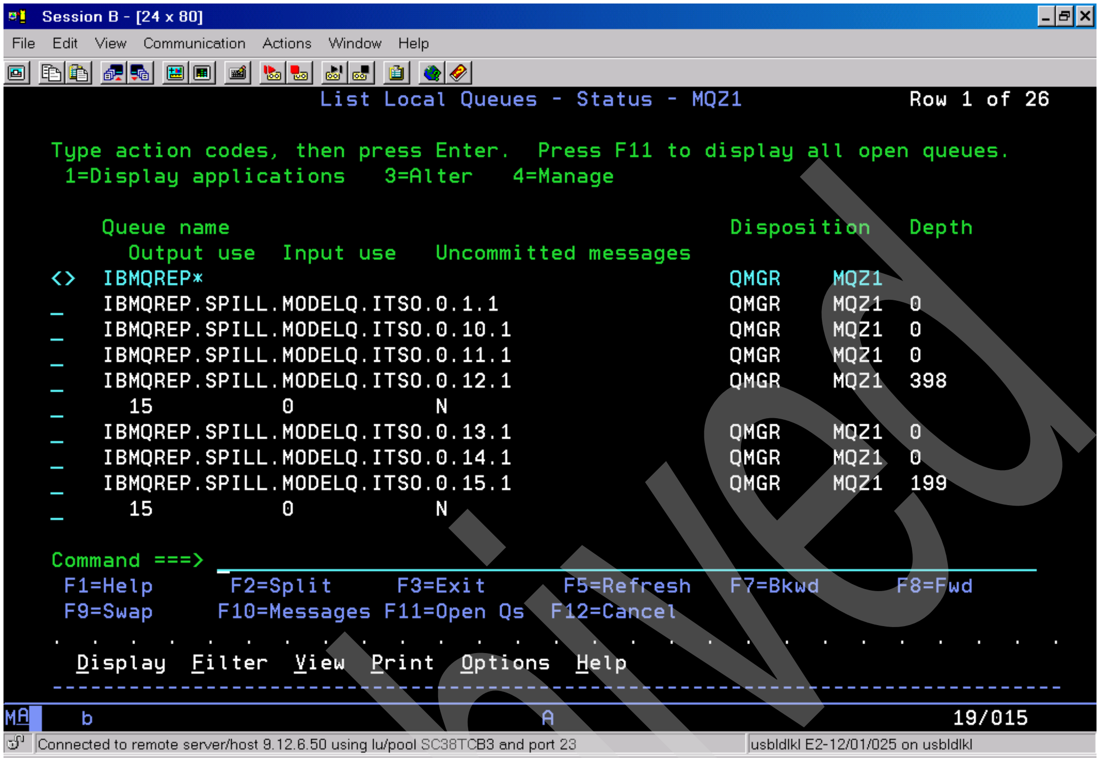


Figure 4-73 Display of spill queues from MVS ISPF panel

When the loads are completed, the user must insert LOADDONE signals at the source of the load site. Example 4-2 shows the SQL for inserting a LOADDONE signal.

Example 4-2 Insert LOADDONE signal at DT11 server

```
INSERT INTO ITSO.IBMQREP_SIGNAL
(SIGNAL_TYPE,SIGNAL_SUBTYPE,SIGNAL_STATE,SIGNAL_INPUT_IN)
 VALUES ('CMD','LOADDONE','P','BAL0002'              );
```

When these LOADDONE signals are received at the target server D8G1, the spill queues are drained at the target, and the subscription becomes active on both sides for peer-to-peer replication, as shown in Figure 4-74.

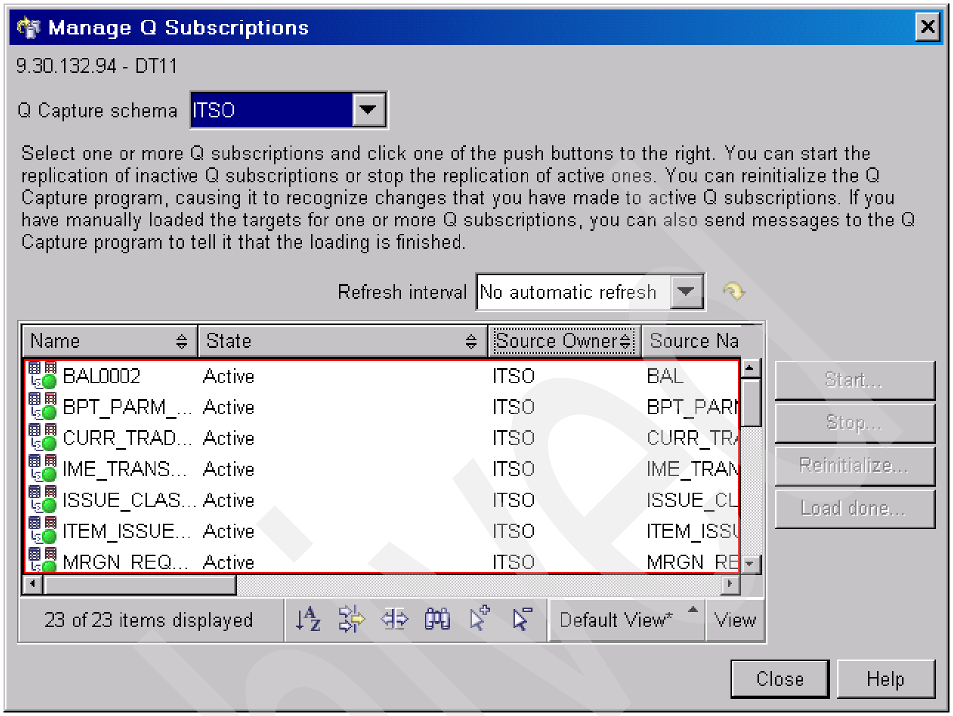


Figure 4-74 List of subscriptions - Active state

4.5.7 Step 6: Configure and activate Q replication using commands

WebSphere II Q replication configuration may be performed using GUI as described in 4.5.6, "Step 6: Configure and activate Q replication using GUI" on page 306, or by using ASNCLP commands to generate SQL scripts.

As mentioned earlier, ASNCLP does not run directly in a z/OS environment; therefore, the SQL scripts cannot be executed from ASNCLP as they are generated. ASNCLP may be executed on UNIX or Windows to generate SQL that can then be ported to z/OS and executed under SPUFI or DSNTEP2.

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 on the workstation, the user environment must be configured to provide support for Java, as shown in Example 4-3.

Example 4-3 Set up Java environment on the workstation

```
set CP=%CP%;c:\progra~1\ibm\sqllib\java\Common.jar;
set CP=%CP%;c:\progra~1\ibm\sqllib\tools\db2cmn.jar;
set CP=%CP%;c:\progra~1\ibm\sqllib\tools\db2replapis.jar;
set CP=%CP%;c:\progra~1\ibm\sqllib\tools\db2qreplapis.jar;
set CP=%CP%;c:\progra~1\ibm\sqllib\tools\jt400.jar;
set CLASSPATH=%CLASSPATH%;%CP%;
```

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

Example 4-4 asnclp setup configuration test from a workstation C: prompt

```
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 4-5.

Example 4-5 Execute commands from a file

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

The ASNCLP commands in the following sections are categorized according to the following tasks:

1. Create the Q Capture and Q Apply control tables.
2. Create the replication queue maps.
3. Create the subscriptions.

Create the Q Capture and Q Apply control tables

Example 4-6 lists the ASNCLP commands for creating the Q Capture and Q Apply control tables on logical database QREPITSO in subsystem DT11 on STLABD1, while Example 4-7 does the same on logical database QREPITSO in subsystem D8G1 on WTSC53. These commands should be saved in a file that is then executed using the command described in Example 4-5 on page 377.

Example 4-6 Create the Q Capture and Q Apply control tables on DT11

```
####################################################
```

```
# The following commands create the control tables
####################################################
ASNCLP SESSION SET TO Q REPLICATION;
SET OUTPUT MULTIDIR;
#############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 4-1 on page 301, Table 4-2 on page 302 and Table 4-3 on page 304
############################################################
##DT11 from reference A.7a in Table 4-1 on page 301
##QREPADM from reference A.9 in Table 4-1 on page 301
##QREPITSO from reference A.10 in Table 4-1 on page 301
##MQS1 from reference A.11 in Table 4-2 on page 302
##Capture schema ITSO from reference A.22 in Table 4-3 on page 304
##QREP.STLA.RESTARTQ from reference A.16 in Table 4-2 on page 302
##//DB2PROD from reference A.24 in Table 4-3 on page 304
##QREP.STLA.ADMINQ from reference A.17 in Table 4-2 on page 302
#############################################################
SET SERVER CAPTURE TO DB DT11 ID QREPADM PASSWORD "xxxxxx";
SET QMANAGER "MQS1" FOR CAPTURE SCHEMA;
SET CAPTURE SCHEMA SOURCE ITSO;
SET RUN SCRIPT LATER;
CREATE CONTROL TABLES FOR CAPTURE SERVER USING RESTARTQ "QREP.STLA.RESTARTQ"
ADMINQ "QREP.STLA.ADMINQ" STARTMODE WARMSI CAPTURE PATH "//DB2PROD" IN ZOS PAGE
LOCK DB "QREPITSO" NAMING PREFIX QC CREATE ROW LOCK DB "QREPITSO" NAMING PREFIX
QC CREATE;
############################################################
##DT11 from reference A.7a in Table 4-1 on page 301
##QREPADM from reference A.9 in Table 4-1 on page 301
##QREPITSO from reference A.10 in Table 4-1 on page 301
##MQS1 from reference A.11 in Table 4-2 on page 302
##//DB2PROD from reference A.25 in Table 4-3 on page 304
##Apply schema ITSO from reference A.23 in Table 4-3 on page 304
#############################################################
SET SERVER TARGET TO DB DT11 ID QREPADM PASSWORD "xxxxxx";
SET QMANAGER "MQS1" FOR APPLY SCHEMA;
SET APPLY SCHEMA ITSO;
SET RUN SCRIPT LATER;
CREATE CONTROL TABLES FOR APPLY SERVER USING APPLY PATH "//DB2PROD"  IN ZOS
PAGE LOCK DB "QREPITSO" NAMING PREFIX QA CREATE ROW LOCK DB "QREPITSO" NAMING
PREFIX QA CREATE;
```

The generated SQL[2] shown in Example 4-7 can be ported to z/OS for execution under SPUFI or DSNTEP2.

[2] When ASNCLP specifies SET RUN SCRIPT LATER, the SQL that is generated by ASNCLP will be stored in a file on the workstation, in the same directory that ASNCLP was executed from. It will have the name of the DB from the SET SERVER Statement with an .sql extension, for example, DT11.sql. If the ASNCLP generates sql for two servers there will be two files generated.

Example 4-7 Generated SQL for DT11 control tables

```
-- DatabaseDB2OS390 (DT11)

-- CONNECT TO DT11 USER XXXX using XXXX;



CREATE TABLESPACE QCITSOCP IN QREPITSO
  SEGSIZE 4
  LOCKSIZE PAGE
  CLOSE NO;



CREATE TABLESPACE QCITSOCR IN QREPITSO
  SEGSIZE 4
  LOCKSIZE ROW
  CLOSE NO;


CREATE TABLE ITSO.IBMQREP_CAPPARMS
(
 QMGR VARCHAR(48) NOT NULL,
 REMOTE_SRC_SERVER VARCHAR(18),
 RESTARTQ VARCHAR(48) NOT NULL,
 ADMINQ VARCHAR(48) NOT NULL,
 STARTMODE VARCHAR(6) NOT NULL WITH DEFAULT 'WARMSI',
 MEMORY_LIMIT INTEGER NOT NULL WITH DEFAULT 32,
 COMMIT_INTERVAL INTEGER NOT NULL WITH DEFAULT 500,
 AUTOSTOP CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 MONITOR_INTERVAL INTEGER NOT NULL WITH DEFAULT 300,
 MONITOR_LIMIT INTEGER NOT NULL WITH DEFAULT 10080,
 TRACE_LIMIT INTEGER NOT NULL WITH DEFAULT 10080,
 SIGNAL_LIMIT INTEGER NOT NULL WITH DEFAULT 10080,
 PRUNE_INTERVAL INTEGER NOT NULL WITH DEFAULT 300,
 SLEEP_INTERVAL INTEGER NOT NULL WITH DEFAULT 5000,
 LOGREUSE CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 LOGSTDOUT CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 TERM CHARACTER(1) NOT NULL WITH DEFAULT 'Y',
 CAPTURE_PATH VARCHAR(1040) WITH DEFAULT NULL,
 ARCH_LEVEL CHARACTER(4) NOT NULL WITH DEFAULT '0802',
 CONSTRAINT CC_STARTMODE CHECK(STARTMODE IN ('COLD','WARMSI','WARMSA'
,'WARMNS')),
 CONSTRAINT CC_MEMORY_LIMIT CHECK(MEMORY_LIMIT >= 1 AND MEMORY_LIMIT
 <= 1000),
 CONSTRAINT CC_COMMIT_INTERVAL CHECK(COMMIT_INTERVAL  >= 100 AND
 COMMIT_INTERVAL <= 600000),
 CONSTRAINT CC_AUTOSTOP CHECK(AUTOSTOP IN ('Y','N')),
```

```
 CONSTRAINT CC_MON_INTERVAL CHECK(MONITOR_INTERVAL >= 1 AND
 MONITOR_INTERVAL <= 2147483647),
 CONSTRAINT CC_MON_LIMIT CHECK(MONITOR_LIMIT >= 1 AND MONITOR_LIMIT
 <= 2147483647),
 CONSTRAINT CC_TRACE_LIMT CHECK(TRACE_LIMIT >= 1 AND TRACE_LIMIT <=
 2147483647),
 CONSTRAINT CC_SIGNAL_LIMIT CHECK(SIGNAL_LIMIT >= 1 AND SIGNAL_LIMIT
 <= 2147483647),
 CONSTRAINT CC_PRUNE_INTERVAL CHECK(PRUNE_INTERVAL >= 1 AND
 PRUNE_INTERVAL <= 2147483647),
 CONSTRAINT CC_LOGREUSE CHECK(LOGREUSE IN ('Y','N')),
 CONSTRAINT CC_LOGSTDOUT CHECK(LOGSTDOUT IN ('Y','N')),
 CONSTRAINT CC_TERM CHECK(TERM IN ('Y','N')),
 CONSTRAINT CC_SLEEP_INTERVAL CHECK(SLEEP_INTERVAL >= 1 AND
 SLEEP_INTERVAL <= 2147483647)
)
 IN QREPITSO.QCITSOCP;


CREATE UNIQUE INDEX ITSO.IX1CQMGRCOL ON ITSO.IBMQREP_CAPPARMS
(
 QMGR ASC
);


CREATE TABLE ITSO.IBMQREP_SENDQUEUES
(
 PUBQMAPNAME VARCHAR(128) NOT NULL,
 SENDQ VARCHAR(48) NOT NULL,
 RECVQ VARCHAR(48),
 MESSAGE_FORMAT CHARACTER(1) NOT NULL WITH DEFAULT 'C',
 MSG_CONTENT_TYPE CHARACTER(1) NOT NULL WITH DEFAULT 'T',
 STATE CHARACTER(1) NOT NULL WITH DEFAULT 'A',
 STATE_TIME TIMESTAMP NOT NULL WITH DEFAULT,
 STATE_INFO CHARACTER(8),
 ERROR_ACTION CHARACTER(1) NOT NULL WITH DEFAULT 'S',
 HEARTBEAT_INTERVAL INTEGER NOT NULL WITH DEFAULT 60,
 MAX_MESSAGE_SIZE INTEGER NOT NULL WITH DEFAULT 64,
 APPLY_SERVER VARCHAR(18),
 APPLY_ALIAS VARCHAR(8),
 APPLY_SCHEMA VARCHAR(128),
 DESCRIPTION VARCHAR(254),
 PRIMARY KEY(SENDQ),
 CONSTRAINT CC_MSG_FORMAT CHECK(MESSAGE_FORMAT IN ('X','C', 'J')),
 CONSTRAINT CC_MSG_CONT_TYPE CHECK(MSG_CONTENT_TYPE IN ('T','R')),
 CONSTRAINT CC_SENDQ_STATE CHECK(STATE IN ('A','I')),
 CONSTRAINT CC_QERRORACTION CHECK(ERROR_ACTION IN ('I','S')),
 CONSTRAINT CC_HTBEAT_INTERVAL CHECK(HEARTBEAT_INTERVAL >= 0 AND
 HEARTBEAT_INTERVAL <= 32767)
```

```
)
 IN QREPITSO.QCITSOCR;


CREATE UNIQUE INDEX ITSO.PKIBMQREP_SENDQUEUES ON
 ITSO.IBMQREP_SENDQUEUES
(
 SENDQ
);


CREATE UNIQUE INDEX ITSO.IX1PUBMAPCOL ON ITSO.IBMQREP_SENDQUEUES
(
 PUBQMAPNAME ASC
);


CREATE TABLE ITSO.IBMQREP_SUBS
(
 SUBNAME VARCHAR(132) NOT NULL,
 SOURCE_OWNER VARCHAR(128) NOT NULL,
 SOURCE_NAME VARCHAR(128) NOT NULL,
 TARGET_SERVER VARCHAR(18),
 TARGET_ALIAS VARCHAR(8),
 TARGET_OWNER VARCHAR(128),
 TARGET_NAME VARCHAR(128),
 TARGET_TYPE INTEGER,
 APPLY_SCHEMA VARCHAR(128),
 SENDQ VARCHAR(48) NOT NULL,
 SEARCH_CONDITION VARCHAR(2048) WITH DEFAULT NULL,
 SUB_ID INTEGER WITH DEFAULT NULL,
 SUBTYPE CHARACTER(1) NOT NULL WITH DEFAULT 'U',
 ALL_CHANGED_ROWS CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 BEFORE_VALUES CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 CHANGED_COLS_ONLY CHARACTER(1) NOT NULL WITH DEFAULT 'Y',
 HAS_LOADPHASE CHARACTER(1) NOT NULL WITH DEFAULT 'I',
 STATE CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 STATE_TIME TIMESTAMP NOT NULL WITH DEFAULT,
 STATE_INFO CHARACTER(8),
 STATE_TRANSITION VARCHAR(256) FOR BIT DATA,
 SUBGROUP VARCHAR(30) WITH DEFAULT NULL,
 SOURCE_NODE SMALLINT NOT NULL WITH DEFAULT 0,
 TARGET_NODE SMALLINT NOT NULL WITH DEFAULT 0,
 GROUP_MEMBERS CHARACTER(254) FOR BIT DATA WITH DEFAULT NULL,
 OPTIONS_FLAG CHARACTER(4) NOT NULL WITH DEFAULT 'NNNN',
 SUPPRESS_DELETES CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 DESCRIPTION VARCHAR(200),
 TOPIC VARCHAR(256),
 PRIMARY KEY(SUBNAME),
```

```
 CONSTRAINT FKSENDQ FOREIGN KEY(SENDQ) REFERENCES
 ITSO.IBMQREP_SENDQUEUES(SENDQ),
 CONSTRAINT CC_SUBTYPE CHECK(SUBTYPE IN ('U','B','P')),
 CONSTRAINT CC_ALL_CHGD_ROWS CHECK(ALL_CHANGED_ROWS IN ('Y','N')),
 CONSTRAINT CC_BEFORE_VALUES CHECK(BEFORE_VALUES IN ('Y','N')),
 CONSTRAINT CC_CHGD_COLS_ONLY CHECK(CHANGED_COLS_ONLY IN ('Y','N')),
 CONSTRAINT CC_HAS_LOADPHASE CHECK(HAS_LOADPHASE IN ('N','I','E')),
 CONSTRAINT CC_SUBS_STATE CHECK(STATE IN ('L','A','I','T','G','N')),
 CONSTRAINT CC_SUPPRESS_DELS CHECK(SUPPRESS_DELETES IN ('Y','N'))
)
 IN QREPITSO.QCITSOCR;


CREATE UNIQUE INDEX ITSO.PKIBMQREP_SUBS ON ITSO.IBMQREP_SUBS
(
 SUBNAME
);


CREATE TABLE ITSO.IBMQREP_SRC_COLS
(
 SUBNAME VARCHAR(132) NOT NULL,
 SRC_COLNAME VARCHAR(128) NOT NULL,
 IS_KEY SMALLINT NOT NULL WITH DEFAULT 0,
 PRIMARY KEY(SUBNAME, SRC_COLNAME),
 CONSTRAINT FKSUBS FOREIGN KEY(SUBNAME) REFERENCES ITSO.IBMQREP_SUBS
(SUBNAME)
)
 IN QREPITSO.QCITSOCP;


CREATE UNIQUE INDEX ITSO.PKIBMQREP_SRC_COLS ON ITSO.IBMQREP_SRC_COLS
(
 SUBNAME,
 SRC_COLNAME
);


CREATE TABLE ITSO.IBMQREP_SRCH_COND
(
 ASNQREQD INTEGER
)
 IN QREPITSO.QCITSOCP;


CREATE TABLE ITSO.IBMQREP_SIGNAL
(
 SIGNAL_TIME TIMESTAMP NOT NULL WITH DEFAULT,
 SIGNAL_TYPE VARCHAR(30) NOT NULL,
```

```
 SIGNAL_SUBTYPE VARCHAR(30),
 SIGNAL_INPUT_IN VARCHAR(500),
 SIGNAL_STATE CHARACTER(1) NOT NULL WITH DEFAULT 'P',
 SIGNAL_LSN CHARACTER(10) FOR BIT DATA,
 PRIMARY KEY(SIGNAL_TIME),
 CONSTRAINT CC_SIGNAL_TYPE CHECK(SIGNAL_TYPE IN ('CMD','USER')),
 CONSTRAINT CC_SIGNAL_STATE CHECK(SIGNAL_STATE IN ('P','R','C','F'))
)
 IN QREPITSO.QCITSOCR
 DATA CAPTURE CHANGES;


CREATE UNIQUE INDEX ITSO.PKIBMQREP_SIGNAL ON ITSO.IBMQREP_SIGNAL
(
 SIGNAL_TIME
);


CREATE TABLE ITSO.IBMQREP_CAPTRACE
(
 OPERATION CHARACTER(8) NOT NULL,
 TRACE_TIME TIMESTAMP NOT NULL,
 DESCRIPTION VARCHAR(1024) NOT NULL
)
 IN QREPITSO.QCITSOCR;


CREATE TABLE ITSO.IBMQREP_CAPMON
(
 MONITOR_TIME TIMESTAMP NOT NULL,
 CURRENT_LOG_TIME TIMESTAMP NOT NULL,
 CAPTURE_IDLE INTEGER NOT NULL,
 CURRENT_MEMORY INTEGER NOT NULL,
 ROWS_PROCESSED INTEGER NOT NULL,
 TRANS_SKIPPED INTEGER NOT NULL,
 TRANS_PROCESSED INTEGER NOT NULL,
 TRANS_SPILLED INTEGER NOT NULL,
 MAX_TRANS_SIZE INTEGER NOT NULL,
 QUEUES_IN_ERROR INTEGER NOT NULL,
 RESTART_SEQ CHARACTER(10) FOR BIT DATA NOT NULL,
 CURRENT_SEQ CHARACTER(10) FOR BIT DATA NOT NULL,
 PRIMARY KEY(MONITOR_TIME)
)
 IN QREPITSO.QCITSOCP;


CREATE UNIQUE INDEX ITSO.PKIBMQREP_CAPMON ON ITSO.IBMQREP_CAPMON
(
 MONITOR_TIME
```

```
);


CREATE TABLE ITSO.IBMQREP_CAPQMON
(
 MONITOR_TIME TIMESTAMP NOT NULL,
 SENDQ VARCHAR(48) NOT NULL,
 ROWS_PUBLISHED INTEGER NOT NULL,
 TRANS_PUBLISHED INTEGER NOT NULL,
 CHG_ROWS_SKIPPED INTEGER NOT NULL,
 DELROWS_SUPPRESSED INTEGER NOT NULL,
 ROWS_SKIPPED INTEGER NOT NULL,
 PRIMARY KEY(MONITOR_TIME, SENDQ)
)
 IN QREPITSO.QCITSOCP;


CREATE UNIQUE INDEX ITSO.PKIBMQREP_CAPQMON ON ITSO.IBMQREP_CAPQMON
(
 MONITOR_TIME,
 SENDQ
);


CREATE TABLE ITSO.IBMQREP_CAPENQ
(
 LOCKNAME INTEGER
)
 IN QREPITSO.QCITSOCR;


CREATE TABLE ITSO.IBMQREP_ADMINMSG
(
 MQMSGID CHARACTER(24) FOR BIT DATA NOT NULL,
 MSG_TIME TIMESTAMP NOT NULL WITH DEFAULT,
 PRIMARY KEY(MQMSGID)
)
 IN QREPITSO.QCITSOCP;


CREATE UNIQUE INDEX ITSO.PKIBMQREP_ADMINMSG ON ITSO.IBMQREP_ADMINMSG
(
 MQMSGID
);


INSERT INTO ITSO.IBMQREP_CAPPARMS
 (qmgr, restartq, adminq, startmode, memory_limit, commit_interval,
 autostop, monitor_interval, monitor_limit, trace_limit, signal_limit,
```

```
 prune_interval, sleep_interval, logreuse, logstdout, term,
 capture_path, arch_level)
 VALUES
 ('MQS1', 'QREP.STLA.RESTARTQ', 'QREP.STLA.ADMINQ', 'WARMSI', 32, 500
, 'N', 300, 10080, 10080, 10080, 300, 5000, 'N', 'N', 'Y', '//DB2PROD'
, '0802');


-- COMMIT;


-- DatabaseDB2OS390 (DT11)

-- CONNECT TO DT11 USER XXXX using XXXX;



CREATE TABLESPACE QAASNAP IN QREPITSO
  SEGSIZE 4
  LOCKSIZE PAGE
  CLOSE NO;



CREATE TABLESPACE QAASNAR IN QREPITSO
  SEGSIZE 4
  LOCKSIZE ROW
  CLOSE NO;


CREATE TABLE ITSO.IBMQREP_APPLYPARMS
(
 QMGR VARCHAR(48) NOT NULL,
 MONITOR_LIMIT INTEGER NOT NULL WITH DEFAULT 10080,
 TRACE_LIMIT INTEGER NOT NULL WITH DEFAULT 10080,
 MONITOR_INTERVAL INTEGER NOT NULL WITH DEFAULT 300,
 PRUNE_INTERVAL INTEGER NOT NULL WITH DEFAULT 300,
 AUTOSTOP CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 LOGREUSE CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 LOGSTDOUT CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 APPLY_PATH VARCHAR(1040) WITH DEFAULT NULL,
 ARCH_LEVEL CHARACTER(4) NOT NULL WITH DEFAULT '0802',
 TERM CHARACTER(1) NOT NULL WITH DEFAULT 'Y',
 PWDFILE VARCHAR(48) WITH DEFAULT NULL,
 DEADLOCK_RETRIES INTEGER NOT NULL WITH DEFAULT 3,
 CONSTRAINT CA_MON_LIMIT CHECK(MONITOR_LIMIT >= 1 AND MONITOR_LIMIT
 <= 2147483647),
 CONSTRAINT CA_TRACE_LIMT CHECK(TRACE_LIMIT >= 1 AND TRACE_LIMIT <=
 2147483647),
```

```
 CONSTRAINT CA_MON_INTERVAL CHECK(MONITOR_INTERVAL >= 1 AND
 MONITOR_INTERVAL <= 2147483647),
 CONSTRAINT CA_PRUNE_INTERVAL CHECK(PRUNE_INTERVAL >= 1 AND
 PRUNE_INTERVAL <= 2147483647),
 CONSTRAINT CA_AUTOSTOP CHECK(AUTOSTOP IN ('Y','N')),
 CONSTRAINT CA_LOGREUSE CHECK(LOGREUSE IN ('Y','N')),
 CONSTRAINT CA_LOGSTDOUT CHECK(LOGSTDOUT IN ('Y','N')),
 CONSTRAINT CA_TERM CHECK(TERM IN ('Y','N')),
 CONSTRAINT CA_RETRIES CHECK(DEADLOCK_RETRIES  >= 3 AND
 DEADLOCK_RETRIES <= 2147483647)
)
 IN QREPITSO.QAASNAP;


CREATE UNIQUE INDEX ITSO.IX1AQMGRCOL ON ITSO.IBMQREP_APPLYPARMS
(
 QMGR ASC
);


CREATE TABLE ITSO.IBMQREP_RECVQUEUES
(
 REPQMAPNAME VARCHAR(128) NOT NULL,
 RECVQ VARCHAR(48) NOT NULL,
 SENDQ VARCHAR(48) WITH DEFAULT NULL,
 ADMINQ VARCHAR(48) NOT NULL,
 NUM_APPLY_AGENTS INTEGER NOT NULL WITH DEFAULT 16,
 MEMORY_LIMIT INTEGER NOT NULL WITH DEFAULT 32,
 CAPTURE_SERVER VARCHAR(18) NOT NULL,
 CAPTURE_ALIAS VARCHAR(8) NOT NULL,
 CAPTURE_SCHEMA VARCHAR(30) NOT NULL WITH DEFAULT 'ASN',
 STATE CHARACTER(1) NOT NULL WITH DEFAULT 'A',
 STATE_TIME TIMESTAMP NOT NULL WITH DEFAULT,
 STATE_INFO CHARACTER(8),
 DESCRIPTION VARCHAR(254),
 PRIMARY KEY(RECVQ),
 CONSTRAINT CC_SENDQ_STATE CHECK(STATE IN ('A','I'))
)
 IN QREPITSO.QAASNAR;


CREATE UNIQUE INDEX ITSO.PKIBMQREP_RECVQUEUES ON
 ITSO.IBMQREP_RECVQUEUES
(
 RECVQ
);


CREATE UNIQUE INDEX ITSO.IX1REPMAPCOL ON ITSO.IBMQREP_RECVQUEUES
```

```
(
 REPQMAPNAME ASC
);


CREATE TABLE ITSO.IBMQREP_TARGETS
(
 SUBNAME VARCHAR(132) NOT NULL,
 RECVQ VARCHAR(48) NOT NULL,
 SUB_ID INTEGER WITH DEFAULT NULL,
 SOURCE_SERVER VARCHAR(18) NOT NULL,
 SOURCE_ALIAS VARCHAR(8) NOT NULL,
 SOURCE_OWNER VARCHAR(128) NOT NULL,
 SOURCE_NAME VARCHAR(128) NOT NULL,
 SRC_NICKNAME_OWNER VARCHAR(128),
 SRC_NICKNAME VARCHAR(128),
 TARGET_OWNER VARCHAR(128) NOT NULL,
 TARGET_NAME VARCHAR(128) NOT NULL,
 TARGET_TYPE INTEGER NOT NULL WITH DEFAULT 1,
 FEDERATED_TGT_SRVR VARCHAR(18) WITH DEFAULT NULL,
 STATE CHARACTER(1) NOT NULL WITH DEFAULT 'I',
 STATE_TIME TIMESTAMP NOT NULL WITH DEFAULT,
 STATE_INFO CHARACTER(8),
 SUBTYPE CHARACTER(1) NOT NULL WITH DEFAULT 'U',
 CONFLICT_RULE CHARACTER(1) NOT NULL WITH DEFAULT 'K',
 CONFLICT_ACTION CHARACTER(1) NOT NULL WITH DEFAULT 'I',
 ERROR_ACTION CHARACTER(1) NOT NULL WITH DEFAULT 'Q',
 SPILLQ VARCHAR(48) WITH DEFAULT NULL,
 OKSQLSTATES VARCHAR(128) WITH DEFAULT NULL,
 SUBGROUP VARCHAR(30) WITH DEFAULT NULL,
 SOURCE_NODE SMALLINT NOT NULL WITH DEFAULT 0,
 TARGET_NODE SMALLINT NOT NULL WITH DEFAULT 0,
 GROUP_INIT_ROLE CHARACTER(1) WITH DEFAULT NULL,
 HAS_LOADPHASE CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 LOAD_TYPE SMALLINT NOT NULL WITH DEFAULT 0,
 DESCRIPTION VARCHAR(254),
 SEARCH_CONDITION VARCHAR(2048) WITH DEFAULT NULL,
 CONSTRAINT CA_TARGTBL_STATE CHECK(STATE IN ('L','A','I','E','D','F'
,'T')),
 CONSTRAINT CA_UPDATEANY CHECK(SUBTYPE IN ('U','B','P')),
 CONSTRAINT CA_CONFLICTACTION CHECK(CONFLICT_ACTION IN ('F','I','D'
,'S','Q')),
 CONSTRAINT CA_ERRORACTION CHECK(ERROR_ACTION IN ('D','S','Q')),
 CONSTRAINT CA_UPANY_SOURCE CHECK(SOURCE_NODE <= 32767 AND
 SOURCE_NODE >= 0 ),
 CONSTRAINT CA_UPANY_TARGET CHECK(TARGET_NODE <= 32767 AND
 TARGET_NODE >= 0 ),
 CONSTRAINT CA_TARGET_TYPE CHECK(TARGET_TYPE >= 1 AND TARGET_TYPE <=
 5 ),
```

```
 CONSTRAINT CA_GROUP_INIT_ROLE CHECK(GROUP_INIT_ROLE IN ('I','M','N')
),
 CONSTRAINT CA_LOAD_TYPE CHECK(LOAD_TYPE >= 0 AND LOAD_TYPE <= 3 )
)
 IN QREPITSO.QAASNAR;


CREATE UNIQUE INDEX ITSO.IX1TARGETS ON ITSO.IBMQREP_TARGETS
(
 SUBNAME ASC,
 RECVQ ASC
);


CREATE UNIQUE INDEX ITSO.IX2TARGETS ON ITSO.IBMQREP_TARGETS
(
 TARGET_OWNER ASC,
 TARGET_NAME ASC,
 RECVQ ASC,
 SOURCE_OWNER ASC,
 SOURCE_NAME ASC
);


CREATE INDEX ITSO.IX3TARGETS ON ITSO.IBMQREP_TARGETS
(
 RECVQ ASC,
 SUB_ID ASC
);


CREATE TABLE ITSO.IBMQREP_TRG_COLS
(
 RECVQ VARCHAR(48) NOT NULL,
 SUBNAME VARCHAR(132) NOT NULL,
 SOURCE_COLNAME VARCHAR(128) NOT NULL,
 TARGET_COLNAME VARCHAR(128) NOT NULL,
 TARGET_COLNO INTEGER WITH DEFAULT NULL,
 MSG_COL_CODEPAGE INTEGER WITH DEFAULT NULL,
 MSG_COL_NUMBER SMALLINT WITH DEFAULT NULL,
 MSG_COL_TYPE SMALLINT WITH DEFAULT NULL,
 MSG_COL_LENGTH INTEGER WITH DEFAULT NULL,
 IS_KEY CHARACTER(1) NOT NULL,
 CONSTRAINT CA_IS_KEY CHECK(IS_KEY IN ('Y','N'))
)
 IN QREPITSO.QAASNAP;


CREATE UNIQUE INDEX ITSO.IX1TRGCOL ON ITSO.IBMQREP_TRG_COLS
```

```
(
 RECVQ ASC,
 SUBNAME ASC,
 TARGET_COLNAME ASC
);


CREATE TABLE ITSO.IBMQREP_SPILLQS
(
 SPILLQ VARCHAR(48) NOT NULL,
 SUBNAME VARCHAR(132) NOT NULL,
 RECVQ VARCHAR(48) NOT NULL,
 PRIMARY KEY(SPILLQ)
)
 IN QREPITSO.QAASNAP;


CREATE UNIQUE INDEX ITSO.PKIBMQREP_SPILLQS ON ITSO.IBMQREP_SPILLQS
(
 SPILLQ
);



CREATE LOB TABLESPACE LXTIBMQR IN QREPITSO
  LOG NO;


CREATE TABLE ITSO.IBMQREP_EXCEPTIONS
(
 EXCEPTION_TIME TIMESTAMP NOT NULL WITH DEFAULT,
 RECVQ VARCHAR(48) NOT NULL,
 SRC_COMMIT_LSN CHARACTER(10) FOR BIT DATA NOT NULL,
 SRC_TRANS_TIME TIMESTAMP NOT NULL,
 SUBNAME VARCHAR(132) NOT NULL,
 REASON CHARACTER(12) NOT NULL,
 SQLCODE INTEGER,
 SQLSTATE CHARACTER(5),
 SQLERRMC VARCHAR(70) FOR BIT DATA,
 OPERATION VARCHAR(18) NOT NULL,
 TEXT CLOB(32768) NOT NULL,
 IS_APPLIED CHARACTER(1) NOT NULL,
 CONFLICT_RULE CHARACTER(1),
 REPLROWID ROWID NOT NULL GENERATED BY DEFAULT,
 CONSTRAINT CA_IS_APPLIED CHECK(IS_APPLIED IN ('Y','N'))
)
 IN QREPITSO.QAASNAP;
```

```
CREATE AUXILIARY TABLE ITSO.XTIBMQREP_EXCEPTIONSO
 IN QREPITSO.LXTIBMQR
 STORES ITSO.IBMQREP_EXCEPTIONS COLUMN TEXT;


CREATE INDEX ITSO.XIXTIBMQREP_EXCEPTIONSO ON
 ITSO.XTIBMQREP_EXCEPTIONSO;


CREATE UNIQUE INDEX ITSO.RIIBMQREP_EXCEPTIONS ON
 ITSO.IBMQREP_EXCEPTIONS
(
 REPLROWID
);


CREATE TABLE ITSO.IBMQREP_APPLYTRACE
(
 OPERATION CHARACTER(8) NOT NULL,
 TRACE_TIME TIMESTAMP NOT NULL,
 DESCRIPTION VARCHAR(1024) NOT NULL
)
 IN QREPITSO.QAASNAR;


CREATE INDEX ITSO.IX1TRCTMCOL ON ITSO.IBMQREP_APPLYTRACE
(
 TRACE_TIME ASC
);


CREATE TABLE ITSO.IBMQREP_APPLYMON
(
 MONITOR_TIME TIMESTAMP NOT NULL,
 RECVQ VARCHAR(48) NOT NULL,
 QSTART_TIME TIMESTAMP NOT NULL,
 CURRENT_MEMORY INTEGER NOT NULL,
 QDEPTH INTEGER NOT NULL,
 END2END_LATENCY INTEGER NOT NULL,
 QLATENCY INTEGER NOT NULL,
 APPLY_LATENCY INTEGER NOT NULL,
 TRANS_APPLIED INTEGER NOT NULL,
 ROWS_APPLIED INTEGER NOT NULL,
 TRANS_SERIALIZED INTEGER NOT NULL,
 RI_DEPENDENCIES INTEGER NOT NULL,
 RI_RETRIES INTEGER NOT NULL,
 DEADLOCK_RETRIES INTEGER NOT NULL,
 ROWS_NOT_APPLIED INTEGER NOT NULL,
```

```
 MONSTER_TRANS INTEGER NOT NULL,
 MEM_FULL_TIME INTEGER NOT NULL,
 APPLY_SLEEP_TIME INTEGER NOT NULL,
 SPILLED_ROWS INTEGER NOT NULL,
 SPILLEDROWSAPPLIED INTEGER NOT NULL,
 OLDEST_TRANS TIMESTAMP NOT NULL,
 OKSQLSTATE_ERRORS INTEGER NOT NULL,
 HEARTBEAT_LATENCY INTEGER NOT NULL,
 KEY_DEPENDENCIES INTEGER NOT NULL,
 UNIQ_DEPENDENCIES INTEGER NOT NULL,
 UNIQ_RETRIES INTEGER NOT NULL,
 PRIMARY KEY(MONITOR_TIME, RECVQ)
)
 IN QREPITSO.QAASNAP;


CREATE UNIQUE INDEX ITSO.PKIBMQREP_APPLYMON ON ITSO.IBMQREP_APPLYMON
(
 MONITOR_TIME,
 RECVQ
);


CREATE TABLE ITSO.IBMQREP_DONEMSG
(
 RECVQ VARCHAR(48) NOT NULL,
 MQMSGID CHARACTER(24) FOR BIT DATA NOT NULL,
 PRIMARY KEY(RECVQ, MQMSGID)
)
 IN QREPITSO.QAASNAP;


CREATE UNIQUE INDEX ITSO.PKIBMQREP_DONEMSG ON ITSO.IBMQREP_DONEMSG
(
 RECVQ,
 MQMSGID
);


CREATE TABLE ITSO.IBMQREP_SPILLEDROW
(
 SPILLQ VARCHAR(48) NOT NULL,
 MQMSGID CHARACTER(24) FOR BIT DATA NOT NULL,
 PRIMARY KEY(SPILLQ, MQMSGID)
)
 IN QREPITSO.QAASNAP;


CREATE UNIQUE INDEX ITSO.PKIBMQREP_SPILLEDROW ON
```

```
 ITSO.IBMQREP_SPILLEDROW
(
 SPILLQ,
 MQMSGID
);


CREATE TABLE ITSO.IBMQREP_SAVERI
(
 SUBNAME VARCHAR(132) NOT NULL,
 RECVQ VARCHAR(48) NOT NULL,
 CONSTNAME VARCHAR(18) NOT NULL,
 TABSCHEMA VARCHAR(128) NOT NULL,
 TABNAME VARCHAR(128) NOT NULL,
 REFTABSCHEMA VARCHAR(128) NOT NULL,
 REFTABNAME VARCHAR(128) NOT NULL,
 ALTER_RI_DDL VARCHAR(1680) NOT NULL,
 TYPE_OF_LOAD CHARACTER(1) NOT NULL,
 CONSTRAINT CA_TYPE_OF_LOAD CHECK(TYPE_OF_LOAD IN ('I','E'))
)
 IN QREPITSO.QAASNAP;


CREATE UNIQUE INDEX ITSO.IX1SAVERI ON ITSO.IBMQREP_SAVERI
(
 SUBNAME ASC,
 RECVQ ASC,
 CONSTNAME ASC
);


CREATE TABLE ITSO.IBMQREP_APPLYENQ
(
 LOCKNAME INTEGER
)
 IN QREPITSO.QAASNAR;


INSERT INTO ITSO.IBMQREP_APPLYPARMS
 (qmgr, monitor_limit, trace_limit, monitor_interval, prune_interval,
 autostop, logreuse, logstdout, apply_path, arch_level, term,
 deadlock_retries)
 VALUES
 ('MQS1', 10080, 10080, 300, 300, 'N', 'N', 'N', '//DB2PROD', '0802',
 'Y', 3);


-- COMMIT;
```

Example 4-8 Create the Q Capture and Q Apply control tables on D8G1

```
######################################################
# The following commands create the control tables
#####################################################
ASNCLP SESSION SET TO Q REPLICATION;
SET OUTPUT MULTIDIR;
#############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 4-1 on page 301, Table 4-2 on page 302 and Table 4-3 on page 304
############################################################
##D8G1 from reference B.7a in Table 4-1 on page 301
##QREPADM from reference B.9 in Table 4-1 on page 301
##QREPITSO from reference B.10 in Table 4-1 on page 301
##MQZ1 from reference B.11 in Table 4-2 on page 302
##Capture schema ITSO from reference B.22 in Table 4-3 on page 304
##QREP.POKA.RESTARTQ from reference B.16 in Table 4-2 on page 302
##//DB2PROD from reference B.24 in Table 4-3 on page 304
##QREP.POKA.ADMINQ from reference B.17 in Table 4-2 on page 302
#############################################################
SET SERVER CAPTURE TO DB D8G1 ID QREPADM PASSWORD "xxxxxx";
SET QMANAGER "MQZ1" FOR CAPTURE SCHEMA;
SET CAPTURE SCHEMA SOURCE ITSO;
SET RUN SCRIPT LATER;
CREATE CONTROL TABLES FOR CAPTURE SERVER USING RESTARTQ "QREP.POKA.RESTARTQ"
ADMINQ "QREP.POKA.ADMINQ" STARTMODE WARMSI CAPTURE PATH "//DB2PROD" IN ZOS PAGE
LOCK DB "QREPITSO" NAMING PREFIX QC CREATE ROW LOCK DB "QREPITSO" NAMING PREFIX
QC CREATE;
############################################################
##D8G1 from reference B.7a in Table 4-1 on page 301
##QREPADM from reference B.9 in Table 4-1 on page 301
##QREPITSO from reference B.10 in Table 4-1 on page 301
##MQZ1 from reference B.11 in Table 4-2 on page 302
##//DB2PROD from reference B.25 in Table 4-3 on page 304
##Apply schema ITSO from reference B.23 in Table 4-3 on page 304
#############################################################
SET SERVER TARGET TO DB D8G1 ID QREPADM PASSWORD "xxxxxx";
SET QMANAGER "MQZ1" FOR APPLY SCHEMA;
SET APPLY SCHEMA ITSO;
SET RUN SCRIPT LATER;
CREATE CONTROL TABLES FOR APPLY SERVER USING APPLY PATH "//DB2PROD"  IN ZOS PAGE
LOCK DB "QREPITSO" NAMING PREFIX QA CREATE ROW LOCK DB "QREPITSO" NAMING PREFIX
QA CREATE;
```

The generated SQL shown in Example 4-9 can be ported to z/OS for execution under SPUFI or DSNTEP2.

Example 4-9 Generated SQL for D8G1 control tables

```
-- DatabaseDB2OS390 (D8G1)

-- CONNECT TO D8G1 USER XXXX using XXXX;



CREATE TABLESPACE QCITSOCP IN QREPITSO
  SEGSIZE 4
  LOCKSIZE PAGE
  CLOSE NO;



CREATE TABLESPACE QCITSOCR IN QREPITSO
  SEGSIZE 4
  LOCKSIZE ROW
  CLOSE NO;


CREATE TABLE ITSO.IBMQREP_CAPPARMS
(
 QMGR VARCHAR(48) NOT NULL,
 REMOTE_SRC_SERVER VARCHAR(18),
 RESTARTQ VARCHAR(48) NOT NULL,
 ADMINQ VARCHAR(48) NOT NULL,
 STARTMODE VARCHAR(6) NOT NULL WITH DEFAULT 'WARMSI',
 MEMORY_LIMIT INTEGER NOT NULL WITH DEFAULT 32,
 COMMIT_INTERVAL INTEGER NOT NULL WITH DEFAULT 500,
 AUTOSTOP CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 MONITOR_INTERVAL INTEGER NOT NULL WITH DEFAULT 300,
 MONITOR_LIMIT INTEGER NOT NULL WITH DEFAULT 10080,
 TRACE_LIMIT INTEGER NOT NULL WITH DEFAULT 10080,
 SIGNAL_LIMIT INTEGER NOT NULL WITH DEFAULT 10080,
 PRUNE_INTERVAL INTEGER NOT NULL WITH DEFAULT 300,
 SLEEP_INTERVAL INTEGER NOT NULL WITH DEFAULT 5000,
 LOGREUSE CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 LOGSTDOUT CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 TERM CHARACTER(1) NOT NULL WITH DEFAULT 'Y',
 CAPTURE_PATH VARCHAR(1040) WITH DEFAULT NULL,
 ARCH_LEVEL CHARACTER(4) NOT NULL WITH DEFAULT '0802',
 CONSTRAINT CC_STARTMODE CHECK(STARTMODE IN ('COLD','WARMSI','WARMSA'
,'WARMNS')),
 CONSTRAINT CC_MEMORY_LIMIT CHECK(MEMORY_LIMIT >= 1 AND MEMORY_LIMIT
 <= 1000),
 CONSTRAINT CC_COMMIT_INTERVAL CHECK(COMMIT_INTERVAL  >= 100 AND
 COMMIT_INTERVAL <= 600000),
 CONSTRAINT CC_AUTOSTOP CHECK(AUTOSTOP IN ('Y','N')),
```

```
 CONSTRAINT CC_MON_INTERVAL CHECK(MONITOR_INTERVAL >= 1 AND
 MONITOR_INTERVAL <= 2147483647),
 CONSTRAINT CC_MON_LIMIT CHECK(MONITOR_LIMIT >= 1 AND MONITOR_LIMIT
 <= 2147483647),
 CONSTRAINT CC_TRACE_LIMT CHECK(TRACE_LIMIT >= 1 AND TRACE_LIMIT <=
 2147483647),
 CONSTRAINT CC_SIGNAL_LIMIT CHECK(SIGNAL_LIMIT >= 1 AND SIGNAL_LIMIT
 <= 2147483647),
 CONSTRAINT CC_PRUNE_INTERVAL CHECK(PRUNE_INTERVAL >= 1 AND
 PRUNE_INTERVAL <= 2147483647),
 CONSTRAINT CC_LOGREUSE CHECK(LOGREUSE IN ('Y','N')),
 CONSTRAINT CC_LOGSTDOUT CHECK(LOGSTDOUT IN ('Y','N')),
 CONSTRAINT CC_TERM CHECK(TERM IN ('Y','N')),
 CONSTRAINT CC_SLEEP_INTERVAL CHECK(SLEEP_INTERVAL >= 1 AND
 SLEEP_INTERVAL <= 2147483647)
)
 IN QREPITSO.QCITSOCP;


CREATE UNIQUE INDEX ITSO.IX1CQMGRCOL ON ITSO.IBMQREP_CAPPARMS
(
 QMGR ASC
);


CREATE TABLE ITSO.IBMQREP_SENDQUEUES
(
 PUBQMAPNAME VARCHAR(128) NOT NULL,
 SENDQ VARCHAR(48) NOT NULL,
 RECVQ VARCHAR(48),
 MESSAGE_FORMAT CHARACTER(1) NOT NULL WITH DEFAULT 'C',
 MSG_CONTENT_TYPE CHARACTER(1) NOT NULL WITH DEFAULT 'T',
 STATE CHARACTER(1) NOT NULL WITH DEFAULT 'A',
 STATE_TIME TIMESTAMP NOT NULL WITH DEFAULT,
 STATE_INFO CHARACTER(8),
 ERROR_ACTION CHARACTER(1) NOT NULL WITH DEFAULT 'S',
 HEARTBEAT_INTERVAL INTEGER NOT NULL WITH DEFAULT 60,
 MAX_MESSAGE_SIZE INTEGER NOT NULL WITH DEFAULT 64,
 APPLY_SERVER VARCHAR(18),
 APPLY_ALIAS VARCHAR(8),
 APPLY_SCHEMA VARCHAR(128),
 DESCRIPTION VARCHAR(254),
 PRIMARY KEY(SENDQ),
 CONSTRAINT CC_MSG_FORMAT CHECK(MESSAGE_FORMAT IN ('X','C', 'J')),
 CONSTRAINT CC_MSG_CONT_TYPE CHECK(MSG_CONTENT_TYPE IN ('T','R')),
 CONSTRAINT CC_SENDQ_STATE CHECK(STATE IN ('A','I')),
 CONSTRAINT CC_QERRORACTION CHECK(ERROR_ACTION IN ('I','S')),
 CONSTRAINT CC_HTBEAT_INTERVAL CHECK(HEARTBEAT_INTERVAL >= 0 AND
 HEARTBEAT_INTERVAL <= 32767)
```

```
)
 IN QREPITSO.QCITSOCR;


CREATE UNIQUE INDEX ITSO.PKIBMQREP_SENDQUEUES ON
 ITSO.IBMQREP_SENDQUEUES
(
 SENDQ
);


CREATE UNIQUE INDEX ITSO.IX1PUBMAPCOL ON ITSO.IBMQREP_SENDQUEUES
(
 PUBQMAPNAME ASC
);


CREATE TABLE ITSO.IBMQREP_SUBS
(
 SUBNAME VARCHAR(132) NOT NULL,
 SOURCE_OWNER VARCHAR(128) NOT NULL,
 SOURCE_NAME VARCHAR(128) NOT NULL,
 TARGET_SERVER VARCHAR(18),
 TARGET_ALIAS VARCHAR(8),
 TARGET_OWNER VARCHAR(128),
 TARGET_NAME VARCHAR(128),
 TARGET_TYPE INTEGER,
 APPLY_SCHEMA VARCHAR(128),
 SENDQ VARCHAR(48) NOT NULL,
 SEARCH_CONDITION VARCHAR(2048) WITH DEFAULT NULL,
 SUB_ID INTEGER WITH DEFAULT NULL,
 SUBTYPE CHARACTER(1) NOT NULL WITH DEFAULT 'U',
 ALL_CHANGED_ROWS CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 BEFORE_VALUES CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 CHANGED_COLS_ONLY CHARACTER(1) NOT NULL WITH DEFAULT 'Y',
 HAS_LOADPHASE CHARACTER(1) NOT NULL WITH DEFAULT 'I',
 STATE CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 STATE_TIME TIMESTAMP NOT NULL WITH DEFAULT,
 STATE_INFO CHARACTER(8),
 STATE_TRANSITION VARCHAR(256) FOR BIT DATA,
 SUBGROUP VARCHAR(30) WITH DEFAULT NULL,
 SOURCE_NODE SMALLINT NOT NULL WITH DEFAULT 0,
 TARGET_NODE SMALLINT NOT NULL WITH DEFAULT 0,
 GROUP_MEMBERS CHARACTER(254) FOR BIT DATA WITH DEFAULT NULL,
 OPTIONS_FLAG CHARACTER(4) NOT NULL WITH DEFAULT 'NNNN',
 SUPPRESS_DELETES CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 DESCRIPTION VARCHAR(200),
 TOPIC VARCHAR(256),
 PRIMARY KEY(SUBNAME),
```

```
 CONSTRAINT FKSENDQ FOREIGN KEY(SENDQ) REFERENCES
 ITSO.IBMQREP_SENDQUEUES(SENDQ),
 CONSTRAINT CC_SUBTYPE CHECK(SUBTYPE IN ('U','B','P')),
 CONSTRAINT CC_ALL_CHGD_ROWS CHECK(ALL_CHANGED_ROWS IN ('Y','N')),
 CONSTRAINT CC_BEFORE_VALUES CHECK(BEFORE_VALUES IN ('Y','N')),
 CONSTRAINT CC_CHGD_COLS_ONLY CHECK(CHANGED_COLS_ONLY IN ('Y','N')),
 CONSTRAINT CC_HAS_LOADPHASE CHECK(HAS_LOADPHASE IN ('N','I','E')),
 CONSTRAINT CC_SUBS_STATE CHECK(STATE IN ('L','A','I','T','G','N')),
 CONSTRAINT CC_SUPPRESS_DELS CHECK(SUPPRESS_DELETES IN ('Y','N'))
)
 IN QREPITSO.QCITSOCR;


CREATE UNIQUE INDEX ITSO.PKIBMQREP_SUBS ON ITSO.IBMQREP_SUBS
(
 SUBNAME
);


CREATE TABLE ITSO.IBMQREP_SRC_COLS
(
 SUBNAME VARCHAR(132) NOT NULL,
 SRC_COLNAME VARCHAR(128) NOT NULL,
 IS_KEY SMALLINT NOT NULL WITH DEFAULT 0,
 PRIMARY KEY(SUBNAME, SRC_COLNAME),
 CONSTRAINT FKSUBS FOREIGN KEY(SUBNAME) REFERENCES ITSO.IBMQREP_SUBS
(SUBNAME)
)
 IN QREPITSO.QCITSOCP;


CREATE UNIQUE INDEX ITSO.PKIBMQREP_SRC_COLS ON ITSO.IBMQREP_SRC_COLS
(
 SUBNAME,
 SRC_COLNAME
);


CREATE TABLE ITSO.IBMQREP_SRCH_COND
(
 ASNQREQD INTEGER
)
 IN QREPITSO.QCITSOCP;


CREATE TABLE ITSO.IBMQREP_SIGNAL
(
 SIGNAL_TIME TIMESTAMP NOT NULL WITH DEFAULT,
 SIGNAL_TYPE VARCHAR(30) NOT NULL,
```

```
 SIGNAL_SUBTYPE VARCHAR(30),
 SIGNAL_INPUT_IN VARCHAR(500),
 SIGNAL_STATE CHARACTER(1) NOT NULL WITH DEFAULT 'P',
 SIGNAL_LSN CHARACTER(10) FOR BIT DATA,
 PRIMARY KEY(SIGNAL_TIME),
 CONSTRAINT CC_SIGNAL_TYPE CHECK(SIGNAL_TYPE IN ('CMD','USER')),
 CONSTRAINT CC_SIGNAL_STATE CHECK(SIGNAL_STATE IN ('P','R','C','F'))
)
 IN QREPITSO.QCITSOCR
 DATA CAPTURE CHANGES;


CREATE UNIQUE INDEX ITSO.PKIBMQREP_SIGNAL ON ITSO.IBMQREP_SIGNAL
(
 SIGNAL_TIME
);


CREATE TABLE ITSO.IBMQREP_CAPTRACE
(
 OPERATION CHARACTER(8) NOT NULL,
 TRACE_TIME TIMESTAMP NOT NULL,
 DESCRIPTION VARCHAR(1024) NOT NULL
)
 IN QREPITSO.QCITSOCR;


CREATE TABLE ITSO.IBMQREP_CAPMON
(
 MONITOR_TIME TIMESTAMP NOT NULL,
 CURRENT_LOG_TIME TIMESTAMP NOT NULL,
 CAPTURE_IDLE INTEGER NOT NULL,
 CURRENT_MEMORY INTEGER NOT NULL,
 ROWS_PROCESSED INTEGER NOT NULL,
 TRANS_SKIPPED INTEGER NOT NULL,
 TRANS_PROCESSED INTEGER NOT NULL,
 TRANS_SPILLED INTEGER NOT NULL,
 MAX_TRANS_SIZE INTEGER NOT NULL,
 QUEUES_IN_ERROR INTEGER NOT NULL,
 RESTART_SEQ CHARACTER(10) FOR BIT DATA NOT NULL,
 CURRENT_SEQ CHARACTER(10) FOR BIT DATA NOT NULL,
 PRIMARY KEY(MONITOR_TIME)
)
 IN QREPITSO.QCITSOCP;


CREATE UNIQUE INDEX ITSO.PKIBMQREP_CAPMON ON ITSO.IBMQREP_CAPMON
(
 MONITOR_TIME
```

```
);


CREATE TABLE ITSO.IBMQREP_CAPQMON
(
 MONITOR_TIME TIMESTAMP NOT NULL,
 SENDQ VARCHAR(48) NOT NULL,
 ROWS_PUBLISHED INTEGER NOT NULL,
 TRANS_PUBLISHED INTEGER NOT NULL,
 CHG_ROWS_SKIPPED INTEGER NOT NULL,
 DELROWS_SUPPRESSED INTEGER NOT NULL,
 ROWS_SKIPPED INTEGER NOT NULL,
 PRIMARY KEY(MONITOR_TIME, SENDQ)
)
 IN QREPITSO.QCITSOCP;


CREATE UNIQUE INDEX ITSO.PKIBMQREP_CAPQMON ON ITSO.IBMQREP_CAPQMON
(
 MONITOR_TIME,
 SENDQ
);


CREATE TABLE ITSO.IBMQREP_CAPENQ
(
 LOCKNAME INTEGER
)
 IN QREPITSO.QCITSOCR;


CREATE TABLE ITSO.IBMQREP_ADMINMSG
(
 MQMSGID CHARACTER(24) FOR BIT DATA NOT NULL,
 MSG_TIME TIMESTAMP NOT NULL WITH DEFAULT,
 PRIMARY KEY(MQMSGID)
)
 IN QREPITSO.QCITSOCP;


CREATE UNIQUE INDEX ITSO.PKIBMQREP_ADMINMSG ON ITSO.IBMQREP_ADMINMSG
(
 MQMSGID
);


INSERT INTO ITSO.IBMQREP_CAPPARMS
 (qmgr, restartq, adminq, startmode, memory_limit, commit_interval,
 autostop, monitor_interval, monitor_limit, trace_limit, signal_limit,
```

```
 prune_interval, sleep_interval, logreuse, logstdout, term,
 capture_path, arch_level)
 VALUES
 ('MQZ1', 'QREP.POKA.RESTARTQ', 'QREP.POKA.ADMINQ', 'WARMSI', 32, 500
, 'N', 300, 10080, 10080, 10080, 300, 5000, 'N', 'N', 'Y', '//DKELSEY'
, '0802');


-- COMMIT;


-- DatabaseDB2OS390 (D8G1)

-- CONNECT TO D8G1 USER XXXX using XXXX;



CREATE TABLESPACE QAASNAP IN QREPITSO
  SEGSIZE 4
  LOCKSIZE PAGE
  CLOSE NO;



CREATE TABLESPACE QAASNAR IN QREPITSO
  SEGSIZE 4
  LOCKSIZE ROW
  CLOSE NO;


CREATE TABLE ITSO.IBMQREP_APPLYPARMS
(
 QMGR VARCHAR(48) NOT NULL,
 MONITOR_LIMIT INTEGER NOT NULL WITH DEFAULT 10080,
 TRACE_LIMIT INTEGER NOT NULL WITH DEFAULT 10080,
 MONITOR_INTERVAL INTEGER NOT NULL WITH DEFAULT 300,
 PRUNE_INTERVAL INTEGER NOT NULL WITH DEFAULT 300,
 AUTOSTOP CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 LOGREUSE CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 LOGSTDOUT CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 APPLY_PATH VARCHAR(1040) WITH DEFAULT NULL,
 ARCH_LEVEL CHARACTER(4) NOT NULL WITH DEFAULT '0802',
 TERM CHARACTER(1) NOT NULL WITH DEFAULT 'Y',
 PWDFILE VARCHAR(48) WITH DEFAULT NULL,
 DEADLOCK_RETRIES INTEGER NOT NULL WITH DEFAULT 3,
 CONSTRAINT CA_MON_LIMIT CHECK(MONITOR_LIMIT >= 1 AND MONITOR_LIMIT
 <= 2147483647),
 CONSTRAINT CA_TRACE_LIMT CHECK(TRACE_LIMIT >= 1 AND TRACE_LIMIT <=
 2147483647),
```

```
 CONSTRAINT CA_MON_INTERVAL CHECK(MONITOR_INTERVAL >= 1 AND
 MONITOR_INTERVAL <= 2147483647),
 CONSTRAINT CA_PRUNE_INTERVAL CHECK(PRUNE_INTERVAL >= 1 AND
 PRUNE_INTERVAL <= 2147483647),
 CONSTRAINT CA_AUTOSTOP CHECK(AUTOSTOP IN ('Y','N')),
 CONSTRAINT CA_LOGREUSE CHECK(LOGREUSE IN ('Y','N')),
 CONSTRAINT CA_LOGSTDOUT CHECK(LOGSTDOUT IN ('Y','N')),
 CONSTRAINT CA_TERM CHECK(TERM IN ('Y','N')),
 CONSTRAINT CA_RETRIES CHECK(DEADLOCK_RETRIES  >= 3 AND
 DEADLOCK_RETRIES <= 2147483647)
)
 IN QREPITSO.QAASNAP;


CREATE UNIQUE INDEX ITSO.IX1AQMGRCOL ON ITSO.IBMQREP_APPLYPARMS
(
 QMGR ASC
);


CREATE TABLE ITSO.IBMQREP_RECVQUEUES
(
 REPQMAPNAME VARCHAR(128) NOT NULL,
 RECVQ VARCHAR(48) NOT NULL,
 SENDQ VARCHAR(48) WITH DEFAULT NULL,
 ADMINQ VARCHAR(48) NOT NULL,
 NUM_APPLY_AGENTS INTEGER NOT NULL WITH DEFAULT 16,
 MEMORY_LIMIT INTEGER NOT NULL WITH DEFAULT 32,
 CAPTURE_SERVER VARCHAR(18) NOT NULL,
 CAPTURE_ALIAS VARCHAR(8) NOT NULL,
 CAPTURE_SCHEMA VARCHAR(30) NOT NULL WITH DEFAULT 'ASN',
 STATE CHARACTER(1) NOT NULL WITH DEFAULT 'A',
 STATE_TIME TIMESTAMP NOT NULL WITH DEFAULT,
 STATE_INFO CHARACTER(8),
 DESCRIPTION VARCHAR(254),
 PRIMARY KEY(RECVQ),
 CONSTRAINT CC_SENDQ_STATE CHECK(STATE IN ('A','I'))
)
 IN QREPITSO.QAASNAR;


CREATE UNIQUE INDEX ITSO.PKIBMQREP_RECVQUEUES ON
 ITSO.IBMQREP_RECVQUEUES
(
 RECVQ
);


CREATE UNIQUE INDEX ITSO.IX1REPMAPCOL ON ITSO.IBMQREP_RECVQUEUES
```

```
(
 REPQMAPNAME ASC
);


CREATE TABLE ITSO.IBMQREP_TARGETS
(
 SUBNAME VARCHAR(132) NOT NULL,
 RECVQ VARCHAR(48) NOT NULL,
 SUB_ID INTEGER WITH DEFAULT NULL,
 SOURCE_SERVER VARCHAR(18) NOT NULL,
 SOURCE_ALIAS VARCHAR(8) NOT NULL,
 SOURCE_OWNER VARCHAR(128) NOT NULL,
 SOURCE_NAME VARCHAR(128) NOT NULL,
 SRC_NICKNAME_OWNER VARCHAR(128),
 SRC_NICKNAME VARCHAR(128),
 TARGET_OWNER VARCHAR(128) NOT NULL,
 TARGET_NAME VARCHAR(128) NOT NULL,
 TARGET_TYPE INTEGER NOT NULL WITH DEFAULT 1,
 FEDERATED_TGT_SRVR VARCHAR(18) WITH DEFAULT NULL,
 STATE CHARACTER(1) NOT NULL WITH DEFAULT 'I',
 STATE_TIME TIMESTAMP NOT NULL WITH DEFAULT,
 STATE_INFO CHARACTER(8),
 SUBTYPE CHARACTER(1) NOT NULL WITH DEFAULT 'U',
 CONFLICT_RULE CHARACTER(1) NOT NULL WITH DEFAULT 'K',
 CONFLICT_ACTION CHARACTER(1) NOT NULL WITH DEFAULT 'I',
 ERROR_ACTION CHARACTER(1) NOT NULL WITH DEFAULT 'Q',
 SPILLQ VARCHAR(48) WITH DEFAULT NULL,
 OKSQLSTATES VARCHAR(128) WITH DEFAULT NULL,
 SUBGROUP VARCHAR(30) WITH DEFAULT NULL,
 SOURCE_NODE SMALLINT NOT NULL WITH DEFAULT 0,
 TARGET_NODE SMALLINT NOT NULL WITH DEFAULT 0,
 GROUP_INIT_ROLE CHARACTER(1) WITH DEFAULT NULL,
 HAS_LOADPHASE CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 LOAD_TYPE SMALLINT NOT NULL WITH DEFAULT 0,
 DESCRIPTION VARCHAR(254),
 SEARCH_CONDITION VARCHAR(2048) WITH DEFAULT NULL,
 CONSTRAINT CA_TARGTBL_STATE CHECK(STATE IN ('L','A','I','E','D','F'
,'T')),
 CONSTRAINT CA_UPDATEANY CHECK(SUBTYPE IN ('U','B','P')),
 CONSTRAINT CA_CONFLICTACTION CHECK(CONFLICT_ACTION IN ('F','I','D'
,'S','Q')),
 CONSTRAINT CA_ERRORACTION CHECK(ERROR_ACTION IN ('D','S','Q')),
 CONSTRAINT CA_UPANY_SOURCE CHECK(SOURCE_NODE <= 32767 AND
 SOURCE_NODE >= 0 ),
 CONSTRAINT CA_UPANY_TARGET CHECK(TARGET_NODE <= 32767 AND
 TARGET_NODE >= 0 ),
 CONSTRAINT CA_TARGET_TYPE CHECK(TARGET_TYPE >= 1 AND TARGET_TYPE <=
 5 ),
```

```
 CONSTRAINT CA_GROUP_INIT_ROLE CHECK(GROUP_INIT_ROLE IN ('I','M','N')
),
 CONSTRAINT CA_LOAD_TYPE CHECK(LOAD_TYPE >= 0 AND LOAD_TYPE <= 3 )
)
 IN QREPITSO.QAASNAR;


CREATE UNIQUE INDEX ITSO.IX1TARGETS ON ITSO.IBMQREP_TARGETS
(
 SUBNAME ASC,
 RECVQ ASC
);


CREATE UNIQUE INDEX ITSO.IX2TARGETS ON ITSO.IBMQREP_TARGETS
(
 TARGET_OWNER ASC,
 TARGET_NAME ASC,
 RECVQ ASC,
 SOURCE_OWNER ASC,
 SOURCE_NAME ASC
);


CREATE INDEX ITSO.IX3TARGETS ON ITSO.IBMQREP_TARGETS
(
 RECVQ ASC,
 SUB_ID ASC
);


CREATE TABLE ITSO.IBMQREP_TRG_COLS
(
 RECVQ VARCHAR(48) NOT NULL,
 SUBNAME VARCHAR(132) NOT NULL,
 SOURCE_COLNAME VARCHAR(128) NOT NULL,
 TARGET_COLNAME VARCHAR(128) NOT NULL,
 TARGET_COLNO INTEGER WITH DEFAULT NULL,
 MSG_COL_CODEPAGE INTEGER WITH DEFAULT NULL,
 MSG_COL_NUMBER SMALLINT WITH DEFAULT NULL,
 MSG_COL_TYPE SMALLINT WITH DEFAULT NULL,
 MSG_COL_LENGTH INTEGER WITH DEFAULT NULL,
 IS_KEY CHARACTER(1) NOT NULL,
 CONSTRAINT CA_IS_KEY CHECK(IS_KEY IN ('Y','N'))
)
 IN QREPITSO.QAASNAP;


CREATE UNIQUE INDEX ITSO.IX1TRGCOL ON ITSO.IBMQREP_TRG_COLS
```

```
(
 RECVQ ASC,
 SUBNAME ASC,
 TARGET_COLNAME ASC
);


CREATE TABLE ITSO.IBMQREP_SPILLQS
(
 SPILLQ VARCHAR(48) NOT NULL,
 SUBNAME VARCHAR(132) NOT NULL,
 RECVQ VARCHAR(48) NOT NULL,
 PRIMARY KEY(SPILLQ)
)
 IN QREPITSO.QAASNAP;


CREATE UNIQUE INDEX ITSO.PKIBMQREP_SPILLQS ON ITSO.IBMQREP_SPILLQS
(
 SPILLQ
);



CREATE LOB TABLESPACE LXTIBMQR IN QREPITSO
  LOG NO;


CREATE TABLE ITSO.IBMQREP_EXCEPTIONS
(
 EXCEPTION_TIME TIMESTAMP NOT NULL WITH DEFAULT,
 RECVQ VARCHAR(48) NOT NULL,
 SRC_COMMIT_LSN CHARACTER(10) FOR BIT DATA NOT NULL,
 SRC_TRANS_TIME TIMESTAMP NOT NULL,
 SUBNAME VARCHAR(132) NOT NULL,
 REASON CHARACTER(12) NOT NULL,
 SQLCODE INTEGER,
 SQLSTATE CHARACTER(5),
 SQLERRMC VARCHAR(70) FOR BIT DATA,
 OPERATION VARCHAR(18) NOT NULL,
 TEXT CLOB(32768) NOT NULL,
 IS_APPLIED CHARACTER(1) NOT NULL,
 CONFLICT_RULE CHARACTER(1),
 REPLROWID ROWID NOT NULL GENERATED BY DEFAULT,
 CONSTRAINT CA_IS_APPLIED CHECK(IS_APPLIED IN ('Y','N'))
)
 IN QREPITSO.QAASNAP;
```

```
CREATE AUXILIARY TABLE ITSO.XTIBMQREP_EXCEPTIONSO
 IN QREPITSO.LXTIBMQR
 STORES ITSO.IBMQREP_EXCEPTIONS COLUMN TEXT;


CREATE INDEX ITSO.XIXTIBMQREP_EXCEPTIONSO ON
 ITSO.XTIBMQREP_EXCEPTIONSO;


CREATE UNIQUE INDEX ITSO.RIIBMQREP_EXCEPTIONS ON
 ITSO.IBMQREP_EXCEPTIONS
(
 REPLROWID
);


CREATE TABLE ITSO.IBMQREP_APPLYTRACE
(
 OPERATION CHARACTER(8) NOT NULL,
 TRACE_TIME TIMESTAMP NOT NULL,
 DESCRIPTION VARCHAR(1024) NOT NULL
)
 IN QREPITSO.QAASNAR;


CREATE INDEX ITSO.IX1TRCTMCOL ON ITSO.IBMQREP_APPLYTRACE
(
 TRACE_TIME ASC
);


CREATE TABLE ITSO.IBMQREP_APPLYMON
(
 MONITOR_TIME TIMESTAMP NOT NULL,
 RECVQ VARCHAR(48) NOT NULL,
 QSTART_TIME TIMESTAMP NOT NULL,
 CURRENT_MEMORY INTEGER NOT NULL,
 QDEPTH INTEGER NOT NULL,
 END2END_LATENCY INTEGER NOT NULL,
 QLATENCY INTEGER NOT NULL,
 APPLY_LATENCY INTEGER NOT NULL,
 TRANS_APPLIED INTEGER NOT NULL,
 ROWS_APPLIED INTEGER NOT NULL,
 TRANS_SERIALIZED INTEGER NOT NULL,
 RI_DEPENDENCIES INTEGER NOT NULL,
 RI_RETRIES INTEGER NOT NULL,
 DEADLOCK_RETRIES INTEGER NOT NULL,
 ROWS_NOT_APPLIED INTEGER NOT NULL,
```

```
 MONSTER_TRANS INTEGER NOT NULL,
 MEM_FULL_TIME INTEGER NOT NULL,
 APPLY_SLEEP_TIME INTEGER NOT NULL,
 SPILLED_ROWS INTEGER NOT NULL,
 SPILLEDROWSAPPLIED INTEGER NOT NULL,
 OLDEST_TRANS TIMESTAMP NOT NULL,
 OKSQLSTATE_ERRORS INTEGER NOT NULL,
 HEARTBEAT_LATENCY INTEGER NOT NULL,
 KEY_DEPENDENCIES INTEGER NOT NULL,
 UNIQ_DEPENDENCIES INTEGER NOT NULL,
 UNIQ_RETRIES INTEGER NOT NULL,
 PRIMARY KEY(MONITOR_TIME, RECVQ)
)
 IN QREPITSO.QAASNAP;


CREATE UNIQUE INDEX ITSO.PKIBMQREP_APPLYMON ON ITSO.IBMQREP_APPLYMON
(
 MONITOR_TIME,
 RECVQ
);


CREATE TABLE ITSO.IBMQREP_DONEMSG
(
 RECVQ VARCHAR(48) NOT NULL,
 MQMSGID CHARACTER(24) FOR BIT DATA NOT NULL,
 PRIMARY KEY(RECVQ, MQMSGID)
)
 IN QREPITSO.QAASNAP;


CREATE UNIQUE INDEX ITSO.PKIBMQREP_DONEMSG ON ITSO.IBMQREP_DONEMSG
(
 RECVQ,
 MQMSGID
);


CREATE TABLE ITSO.IBMQREP_SPILLEDROW
(
 SPILLQ VARCHAR(48) NOT NULL,
 MQMSGID CHARACTER(24) FOR BIT DATA NOT NULL,
 PRIMARY KEY(SPILLQ, MQMSGID)
)
 IN QREPITSO.QAASNAP;


CREATE UNIQUE INDEX ITSO.PKIBMQREP_SPILLEDROW ON
```

```
 ITSO.IBMQREP_SPILLEDROW
(
 SPILLQ,
 MQMSGID
);


CREATE TABLE ITSO.IBMQREP_SAVERI
(
 SUBNAME VARCHAR(132) NOT NULL,
 RECVQ VARCHAR(48) NOT NULL,
 CONSTNAME VARCHAR(18) NOT NULL,
 TABSCHEMA VARCHAR(128) NOT NULL,
 TABNAME VARCHAR(128) NOT NULL,
 REFTABSCHEMA VARCHAR(128) NOT NULL,
 REFTABNAME VARCHAR(128) NOT NULL,
 ALTER_RI_DDL VARCHAR(1680) NOT NULL,
 TYPE_OF_LOAD CHARACTER(1) NOT NULL,
 CONSTRAINT CA_TYPE_OF_LOAD CHECK(TYPE_OF_LOAD IN ('I','E'))
)
 IN QREPITSO.QAASNAP;


CREATE UNIQUE INDEX ITSO.IX1SAVERI ON ITSO.IBMQREP_SAVERI
(
 SUBNAME ASC,
 RECVQ ASC,
 CONSTNAME ASC
);


CREATE TABLE ITSO.IBMQREP_APPLYENQ
(
 LOCKNAME INTEGER
)
 IN QREPITSO.QAASNAR;


INSERT INTO ITSO.IBMQREP_APPLYPARMS
 (qmgr, monitor_limit, trace_limit, monitor_interval, prune_interval,
 autostop, logreuse, logstdout, apply_path, arch_level, term,
 deadlock_retries)
 VALUES
 ('MQZ1', 10080, 10080, 300, 300, 'N', 'N', 'N', '//DKELSEY', '0802',
 'Y', 3);


-- COMMIT;
```

Attention: Both the generated scripts *must be executed before* the next step of creating the replication queue maps.

Create the replication queue maps

Example 4-10 lists the ASNCLP commands for creating the replication queue maps on DT11 (STLABD1), while Example 4-13 does the same on D8G1 (WTSC53). These commands should be saved in a file that is then executed using the command described in Example 4-5 on page 377.

Example 4-10 Create the replication queue maps for replication from DT11 to D8G1

```
#############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 4-1 on page 301, Table 4-2 on page 302 and Table 4-3 on page 304
############################################################
##DT11 from reference A.7a in Table 4-1 on page 301
##QREPADM from reference A.9 in Table 4-1 on page 301
##D8G1 from reference B.7a in Table 4-1 on page 301
##QREPADM from reference B.9 in Table 4-1 on page 301
##Capture schema ITSO from reference A.22 in Table 4-3 on page 304
##Apply schema ITSO from reference B.23 in Table 4-3 on page 304
##QMAP_STL_TO_POK from reference A.27 in Table 4-3 on page 304
##QREP.STLA.ADMINQ from reference A.17 in Table 4-2 on page 302
##RECVQ QREP.STLA.TO.POKA.SENDQ from reference B.19 in Table 4-2 on page 302
##SENDQ QREP.STLA.TO.POKA.SENDQ from reference A.18 in Table 4-2 on page 302
##Number of apply agents 16 from reference A.27.c in Table 3-3 on page 96
##Memory Limit 2 from reference A.27.d in Table 3-3 on page 96
##Error Action 'S' from reference A.27.b in Table 3-3 on page 96
##MSGSIZE 64 from reference A.27.a in Table 3-3 on page 96
#############################################################

ASNCLP SESSION SET TO Q REPLICATION;
SET OUTPUT MULTIDIR;
SET SERVER CAPTURE TO DB DT11 ID "QREPADM" PASSWORD "xxxxxx";
SET SERVER TARGET  TO DB D8G1 ID "QREPADM" PASSWORD "xxxxxx";

SET CAPTURE SCHEMA SOURCE ITSO;
SET APPLY   SCHEMA ITSO;
SET RUN SCRIPT LATER;
CREATE REPLQMAP QMAP_STL_TO_POK USING ADMINQ "QREP.STLA.ADMINQ" RECVQ
"QREP.STLA.TO.POKA.RECVQ" SENDQ "QREP.STLA.TO.POKA.SENDQ" NUM APPLY AGENTS 16
MEMORY LIMIT 2 ERROR ACTION S MAX MESSAGE SIZE 64;
```

Two SQL members are generated (one each for the SEND and RECV replication queue map), as shown in Example 4-11 and Example 4-12, which can be ported to z/OS for execution under SPUFI or DSNTEP2.

Example 4-11 Generated SQL for replication queue map SENDQ DT11 to D8G1

```
-- DatabaseDB2OS390 (DT11)

-- CONNECT TO DT11 USER XXXX using XXXX;


INSERT INTO ITSO.IBMQREP_SENDQUEUES
 (pubqmapname, sendq, message_format, msg_content_type, state,
 error_action, heartbeat_interval, max_message_size, apply_alias,
 apply_schema, recvq, apply_server)
 VALUES
 ('QMAP_STL_TO_POK', 'QREP.STLA.TO.POKA.SENDQ', 'C', 'T', 'A', 'S',
 60, 64, 'D8G1', 'ITSO', 'QREP.STLA.TO.POKA.RECVQ', 'DB8G');


-- COMMIT;
```

Example 4-12 Generated SQL for replication queue map RECVQ DT11 to D8G1

```
-- DatabaseDB2OS390 (D8G1)

-- CONNECT TO D8G1 USER XXXX using XXXX;


INSERT INTO ITSO.IBMQREP_RECVQUEUES
 (repqmapname, recvq, sendq, adminq, capture_alias, capture_schema,
 num_apply_agents, memory_limit, state, capture_server)
 VALUES
 ('QMAP_STL_TO_POK', 'QREP.STLA.TO.POKA.RECVQ',
 'QREP.STLA.TO.POKA.SENDQ', 'QREP.STLA.ADMINQ', 'DT11', 'ITSO', 16, 2,
 'A', 'DSNT1');


-- COMMIT;
```

Example 4-13 Create the replication queue maps for replication from D8G1 to DT11

```
############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 4-1 on page 301, Table 4-2 on page 302 and Table 4-3 on page 304
############################################################
##DT11 from reference A.7a in Table 4-1 on page 301
```

```
##QREPADM from reference A.9 in Table 4-1 on page 301
##D8G1 from reference B.7a in Table 4-1 on page 301
##QREPADM from reference B.9 in Table 4-1 on page 301
##Capture schema ITSO from reference B.22 in Table 4-3 on page 304
##Apply schema ITSO from reference A.23 in Table 4-3 on page 304
##QMAP_POK_TO_STL from reference B.27 in Table 4-3 on page 304
##QREP.POKA.ADMINQ from reference B.17 in Table 4-2 on page 302
##RECVQ QREP.POKA.TO.STLA.RECVQ from reference A.19 in Table 4-2 on page 302
##SENDQ QREP.POKA.TO.STLA.RECVQ from reference B.18 in Table 4-2 on page 302
##Number of apply agents 16 from reference B.27.c in Table 3-3 on page 96
##Memory Limit 2 from reference B.27.d in Table 3-3 on page 96
##Error Action 'S' from reference B.27.b in Table 3-3 on page 96
##MSGSIZE 64 from reference B.27.a in Table 3-3 on page 96
#############################################################
ASNCLP SESSION SET TO Q REPLICATION;
SET OUTPUT MULTIDIR;

SET SERVER CAPTURE TO DB D8G1 ID "QREPADM" PASSWORD "xxxxxx";
SET SERVER TARGET  TO DB DT11 ID "QREPADM" PASSWORD "xxxxxx";

SET CAPTURE SCHEMA SOURCE ITSO;
SET APPLY   SCHEMA ITSO;
SET RUN SCRIPT LATER;
CREATE REPLQMAP QMAP_POK_TO_STL USING ADMINQ "QREP.POKA.ADMINQ" RECVQ
"QREP.POKA.TO.STLA.RECVQ" SENDQ "QREP.POKA.TO.STLA.RECVQ" NUM APPLY AGENTS 16
MEMORY LIMIT 2 ERROR ACTION S MAX MESSAGE SIZE 64;
```

Two SQL members are generated (one each for the SEND and RECV replication queue map), as shown in Example 4-14 and Example 4-15, which can be ported to z/OS for execution under SPUFI or DSNTEP2.

Example 4-14 Generated SQL for replication queue map SENDQ D8G1 to DT11

```
-- DatabaseDB2OS390 (D8G1)

-- CONNECT TO D8G1 USER XXXX using XXXX;


INSERT INTO ITSO.IBMQREP_SENDQUEUES
 (pubqmapname, sendq, message_format, msg_content_type, state,
 error_action, heartbeat_interval, max_message_size, apply_alias,
 apply_schema, recvq, apply_server)
 VALUES
 ('QMAP_POK_TO_STL', 'QREP.POKA.TO.STLA.SENDQ', 'C', 'T', 'A', 'S',
 60, 64, 'DT11', 'ITSO', 'QREP.POKA.TO.STLA.RECVQ', 'DSNT1');


-- COMMIT;
```

Example 4-15 Generated SQL for replication queue map RECVQ D8G1 to DT11

```
-- DatabaseDB2OS390 (DT11)

-- CONNECT TO DT11 USER XXXX using XXXX;


INSERT INTO ITSO.IBMQREP_RECVQUEUES
 (repqmapname, recvq, sendq, adminq, capture_alias, capture_schema,
 num_apply_agents, memory_limit, state, capture_server)
 VALUES
 ('QMAP_POK_TO_STL', 'QREP.POKA.TO.STLA.RECVQ',
 'QREP.POKA.TO.STLA.SENDQ', 'QREP.POKA.ADMINQ', 'D8G1', 'ITSO', 16, 2,
 'A', 'DB8G');


-- COMMIT;
```

Attention: These scripts *must be executed before* the next step of creating the subscriptions.

Create the subscriptions

Example 4-16 lists the ASNCLP commands for creating the subscriptions on DT11 (STLABD1) and D8G1 (WTSC53). These commands should be saved in a file (ASNCLP_STL04_subs1.in).

Example 4-16 Create the subscriptions

```
#############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 4-1 on page 301, Table 4-2 on page 302 and Table 4-3 on page 304
############################################################
##DT11 from reference A.7a in Table 4-1 on page 301
##QREPADM from reference A.9 in Table 4-1 on page 301
##D8G1 from reference B.7a in Table 4-1 on page 301
##QREPADM from reference B.9 in Table 4-1 on page 301
##Capture schema ITSO from reference A.22 in Table 4-3 on page 304
##Apply schema ITSO from reference B.23 in Table 4-3 on page 304
##QMAP_STL_TO_POK from reference A.27 in Table 4-3 on page 304
##QMAP_POK_TO_STL from reference B.27 in Table 4-3 on page 304
#############################################################
SET SUBGROUP "000001";
SET SERVER MULTIDIR TO DB D8G1 ID "QREPADM" PASSWORD "xxxxxx";
SET SERVER MULTIDIR TO DB DT11 ID "QREPADM" PASSWORD "xxxxxx";
```

```
SET MULTIDIR SCHEMA D8G1.ITSO;
SET MULTIDIR SCHEMA DT11.ITSO;


SET CONNECTION SOURCE DT11.ITSO TARGET D8G1.ITSO REPLQMAP QMAP_STL_TO_POK;
SET CONNECTION SOURCE D8G1.ITSO TARGET DT11.ITSO REPLQMAP QMAP_POK_TO_STL;
##################################################################
## The following set of statements need to be defined for each table to
## be replication (from reference A.29 on Table 3-3 page 96
## SET TABLES = (A.7).(A.22).CREATORNAME.TABLENAME,
## (B.7).(B.22).CREATORNAME.TABLENAME
## where A.7 and B.7 are from reference Table 3-1 page 93
## and A.22 and B.22 are from reference Table 3-3 page 96
##
## The CREATE QSUB statements all have the following format:
##
## CREATE QSUB SUBTYPE A.26        <-- A.26 from reference table 3-3 page 96
## SOURCE HAS LOADPHASE A.28.g     <-- A.28.g from reference table 3-3 page 96
## TARGET ERROR ACTION B.28.f      <-- B.28.f from reference table 3-3 page 96
###############################################
set tables (DT11.ITSO.ITSO.BAL,  D8G1.ITSO.ITSO.BAL);
   create qsub subtype p
   source has load phase e
   target error action s;
```

The ASNCLP script to execute this multiple subscription script is shown in Example 4-17. When this script is executed from a command prompt, it forces the execution of the CREATE statements in the first script above, and SQL is generated that can be ported to the z/OS systems for execution under SPUFI or DSNTEP2.

Example 4-17 Execute multiple subscription script

```
ASNCLP SESSION SET TO Q REPLICATION;
SET OUTPUT MULTIDIR;
SET RUN SCRIPT LATER;
LOAD MULTIDIR REPL SCRIPT "ASNCLP_STL04_subs1.in";
```

Start the Q Capture and Q Apply programs

"Step 6f: Start the Q Capture on both servers" on page 173, "Step 6g: Start the Q Apply on both servers" on page 176, and "Step 6h: Verify status of Q Capture and Q Apply processes" on page 178 describe the steps involved in starting Q Capture and Q Apply and verifying their status.

4.5.8 Step 7: Set up monitoring environment using the GUI

We recommend setting up the monitoring environment with appropriate alert conditions for the peer-to-peer replication environment to ensure a stable operating environment.

Figure 4-75 lists the steps involved in setting up the monitoring environment.

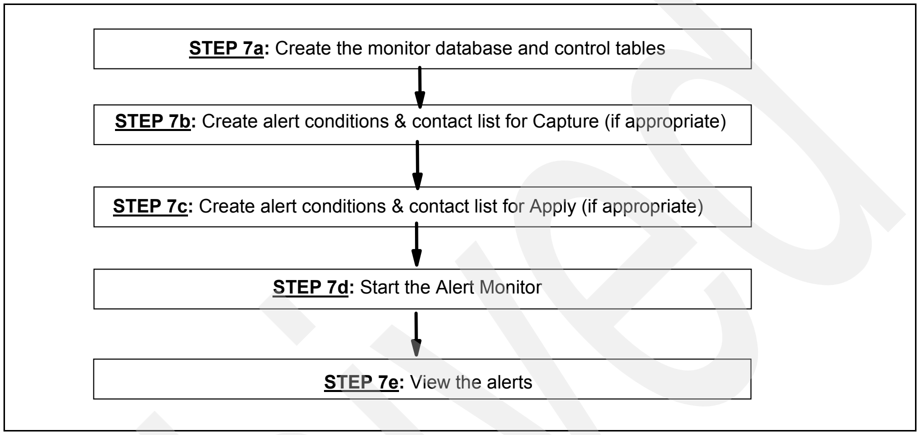


Figure 4-75 Steps to set up monitoring environment using GUI

Section 3.5.8, “Step 7: Set up monitoring environment using GUI” on page 230; and 3.5.9, “Step 7: Set up monitoring environment using commands” on page 271, describe in detail the steps in setting up a Q replication monitoring environment using a GUI and commands, respectively, for the peer-to-peer replication environment. The same series of steps apply to monitoring the bidirectional replication environment as well.

4.6 Failover considerations

Carthage’s Q replication peer-to-peer topology involves two z/OS servers, STLABD1 and WTSC53, where either server can have updates occurring and there is no concept of a primary or secondary server since the two servers are considered to be peers.

The requirement is for the surviving server to take over the workload of the failed server. When the failed server is restored, the requirement is for the peer-to-peer relationship to be reestablished between the two servers.

Attention: This section provides a very high-level overview of some of the considerations involved with failover processing associated with peer-to-peer replication. Depending upon the particular environment, the process involved in ensuring satisfactory failover processing can get quite complex. A full discussion of these considerations is beyond the scope of this book.

When the failover occurs to the surviving server, it is possible for some of the changes that occurred at the failed server not to be replicated over to the surviving 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 4-76), or messages in the WebSphere MQ queue that did not get transmitted to the surviving server (item 2 in Figure 4-76). These un-replicated changes should be considered to be data loss at the surviving server—at least until the failed server is restored. If there are messages in the receive queue on the secondary server that have not been drained (item 4 in Figure 4-76) 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.

Important: Identifying the data loss suffered can be critical for certain types of applications. Approaches for identifying the data loss suffered is beyond the scope of this book.

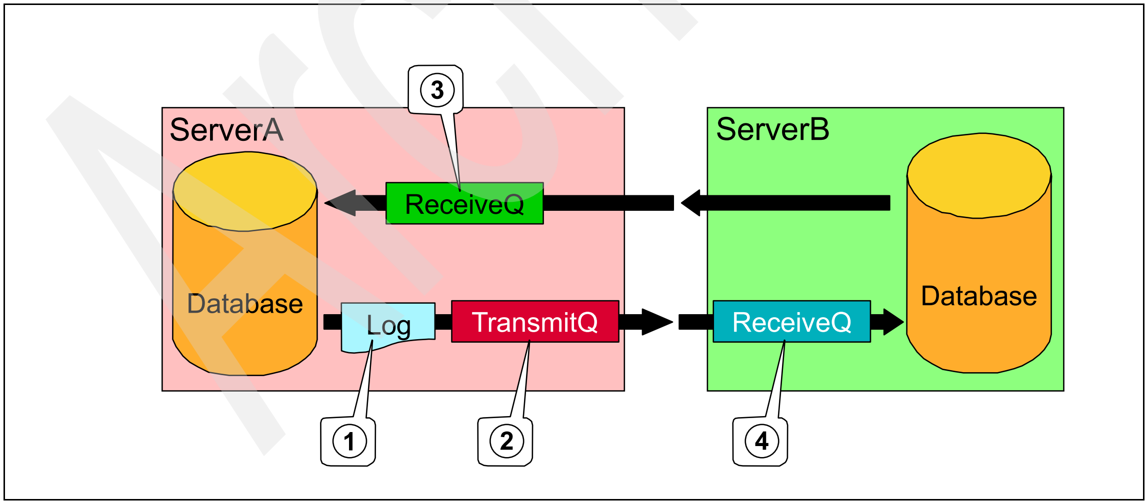


Figure 4-76 Potential sources of conflicts

The peer-to-peer scenario, by definition, has the potential for conflicts occurring at either server—during normal processing, as well as failover and switchback[3]. The timestamp-based conflict detection and resolution algorithm ensures that the most recent changes to the data are the winners in the event of conflicts.

Note: The triggering event for failover is assumed to originate external to the Q replication environment.

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

1. The Q Apply program will soon catch up with any messages sent by the failed server, and will have no additional processing to perform until switchback.
2. 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. The current depth of the queue may be determined from the MQ ISPF panels in TSO, or by executing the MVS console command shown in Example 4-18.

Example 4-18 Console command to determine depth of transmit queue on MQZ1

```
=MQZ1 dis ql(MQS1XMIT) curdepth
```

3. 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 failed server is unavailable. The value of MAXDEPTH depends on the:

- 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)

[3] Switchback is the process of restoring the Q replication environment to its normal operating environment after failover processing has occurred.

If the failed server outage is expected to be large, that may cause the transmit queue to fill up; then Q Capture must be shut down before the transmit queue fills up. Example 4-19 shows an SQL script that can be run on D8G1 to identify the states and load phases of all subscriptions.

Example 4-19 Check the subscriptions states

```
SELECT SUBNAME, STATE, HAS_LOADPHASE FROM ITSO.IBMQREP_SUBS;
```

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 failed 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 the updates on the DB2 log (after a Q Capture shutdown) exceeds an acceptable switchback time or the failed 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.

4.7 Switchback 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 Carthage's peer-to-peer topology this involves restoring the failed server as a peer of the surviving server. Switchback should involve minimum data loss.

Important: Identifying the data loss suffered can be critical for certain types of applications. Approaches for identifying the data loss suffered is beyond the scope of this book.

Attention: This section provides a very high-level overview of some of the considerations involved with switchback processing associated with peer-to-peer replication. Depending upon the particular environment, the process involved in ensuring satisfactory switchback processing can get quite complex. A full discussion of these considerations is beyond the scope of this book.

The effort involved in restoring the normal operating environment depends upon the expected duration of the outage of the failed server.

- Short outage

  If the duration of the failed server outage is short and Q Capture was not shut down on the surviving server, then the peer-to-peer replication configuration implemented will resynchronize and resume normal operations simply by restarting the failed server, and warm starting Q Capture on the failed server.

- Extended outage

  If the duration of the primary server outage was estimated to be so long that Q Capture was shut down to prevent the queues from filling up, then after restarting the failed server, adequate time must be provided so that the messages already stored in the WebSphere MQ queues can be processed by Q Apply on the primary server. This may be determined by starting the Q Apply proc (as described in Example 3-27 on page 176) after modifying it to add an additional parameter, as shown in Example 4-20. This will cause the Q Apply program to terminate after all receive queues are emptied once. After the messages have been processed, Q Capture can be restarted by starting its proc, as described in Example 3-20 on page 173, after modifying it to add an additional parameter, as shown in Example 4-21. This will cause the Q Capture to stop after reaching the end of the DB2 active log.

Example 4-20 Modify Q Apply proc parameter

```
APARM='autostop=y'
```

Example 4-21 Modify Q Capture proc parameter

```
APARM='autostop=y'
```

  If despite adequate planning, Q Capture on the surviving server deactivated the subscriptions, then a careful analysis is required to determine the appropriate recovery action, such as:

  – If the Q Subscription was configured for Automatic Load, then reactivation of the subscription will cause the target table to be emptied and reloaded. For a large table this could take significant time and resources. The CAPSTART signal must be issued to the Q Capture program on the surviving side to designate source and target tables.
  – If the Q Subscription was configured for a Manual Load, then a CAPSTART signal must be issued. Reconfirm that the tables are synchronized, and then a LOADDONE signal is issued to complete subscription activation.
  – If appropriate, the existing subscriptions can be dropped and recreated.

Note: It is assumed that the failed server with WebSphere II is restarted successfully to the point of failure.

5

Bidirectional Q replication on AIX platforms

In this chapter we describe a step-by-step approach to implementing a bidirectional queue replication financial industry solution on AIX platforms.

The topics covered are:

- Business requirements
- Rationale for the bidirectional solution
- Environment configuration
- Step-by-step setup
- Failover considerations
- Switchback considerations

5.1 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 backup 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 cycle 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 implements it in the target environment. The entire process and associated considerations are documented as follows:

- Business requirements
- Rationale for choosing the bidirectional solution
- Environment configuration
- Step-by-step setup
- Failover considerations
- Switchback considerations

5.2 Business requirements

Our fictitious company Corona 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 to 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, Corona is required to make the hot site available for read-only access for reporting purposes to offload processing from the primary server.

These requirements may be summarized as follows:

- 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.
- Replicated tables on the second server must be available for switchover within 30 minutes.
- Replicated tables on the second server must be available for read-only access while the primary server is available.
- The primary server is to be resynchronized with the secondary after it is restored to service.
- Limited resources allowed for replication-only purposes.

5.3 Rationale for the bidirectional solution

Table 2-1 on page 39 lists the evaluation criteria for choosing between unidirectional, bidirectional, and peer-to-peer replication topologies.

Since Corona's business requirement is support for a single backup site, the choice is between a bidirectional and peer-to-peer replication topology.

The bidirectional replication topology is appropriate for Corona for the following reasons:

- Less stringent requirements for switchover time (within 30 minutes instead of an instantaneous requirement, which would tilt in favor of peer-to-peer).
- Non-dedicated backup server requirement.
- Absence of conflicts[1] due to the lack of simultaneous updates—only one server is updated at a given time.
- Limited computing resources available.
- As discussed in 2.3, "Choosing a particular Q replication topology" on page 39, 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

[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 conditions are resolved partially by the conflict resolution mechanism during switchback, and may require manual intervention.

alternative and does not require additional columns and triggers to be defined on existing data.

- Corona's requirement for a controlled switchback to the primary server from the secondary server is handled well by a bidirectional replication topology.

5.4 Environment configuration

Figure 5-1 shows the configuration used in the Corona bidirectional replication topology.

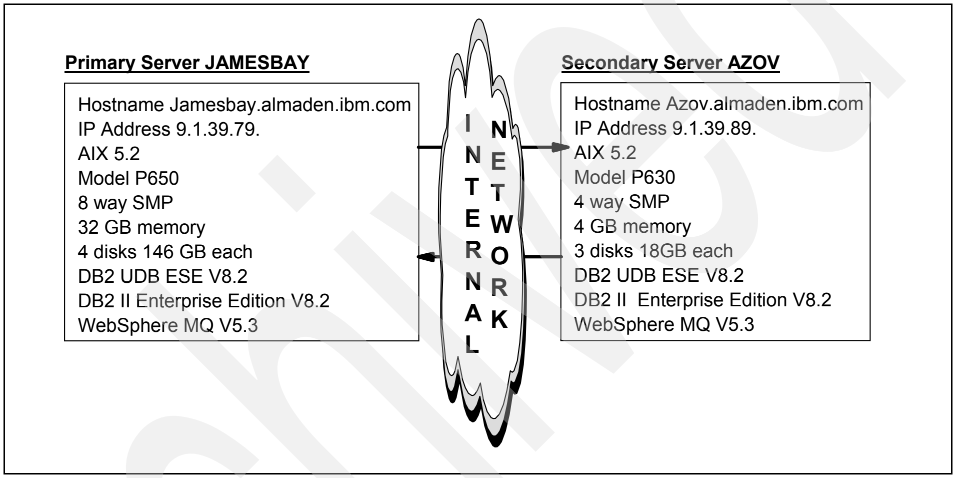


Figure 5-1 Bidirectional replication topology configuration for Corona

We installed a very simple set of three tables with referential integrity constraints defined between them. These tables included LOB data that were replicated as well. Full details of these tables are documented in Appendix F, "Tables used in Q replication scenarios" on page 781.

5.5 Step-by-step setup

In this section we document the step-by-step setup of the bidirectional replication topology in our fictitious company. Figure 5-2 lists the main steps involved in setting up the environment. Each of these steps is described in detail in the following subsections.

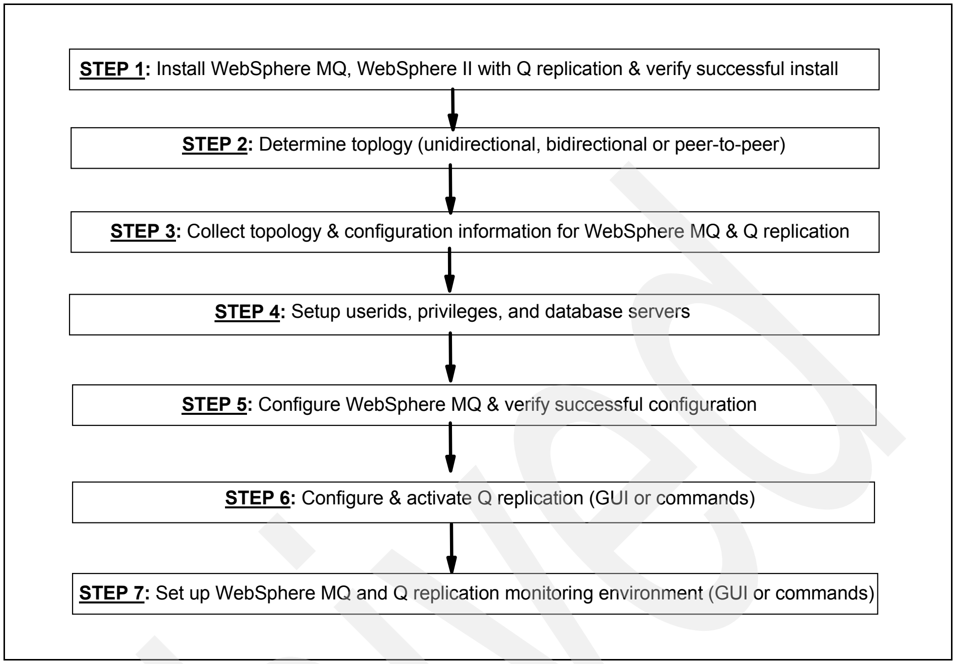


Figure 5-2 Overview of setup steps

5.5.1 Step 1: Install WebSphere MQ, WebSphere II with Q replication

Refer to *IBM WebSphere MQ for AIX Quick Beginnings Version 5.3*, GC34-6076-02; and *IBM DB2 Information Integrator Installation Guide for Linux, Unix and Windows Version 8.2,* GC18-7036-01, for details on prerequisites and steps for installing WebSphere MQ and WebSphere II with Q replication, respectively.

5.5.2 Step 2: Determine topology

We chose the bidirectional replication topology to address the Corona business requirement, as described in 5.3, “Rationale for the bidirectional solution” on page 421.

5.5.3 Step 3: Collect topology and configuration information

Implementing bidirectional replication is a complex task involving effective coordination of the configuration settings of the operating system, database management system, WebSphere MQ and WebSphere Information Integrator Q replication offerings.

Important: To ensure successful and error-free implementation, we strongly encourage systematic and careful planning that involves identifying and gathering all required information prior to commencing implementation.

Towards this end, we have developed a template that identifies all the information required to implement a bidirectional replication topology, and the cross-relationships between the information elements to ensure a smooth implementation.

Figure 5-3 provides a high-level overview of the various objects involved in implementing a bidirectional replication topology, and serves as a reference for the host and DB2 system information template (Table 5-1 on page 425), WebSphere MQ configuration information template (Table 5-2 on page 426), Q replication configuration information template (Table 5-3 on page 428), and Replication Alert Monitor configuration information template (Table 5-4 on page 428). Each of the information elements for each server is associated with a reference identification such as A.7a or B.7b, where A represents one of the servers and B the other server. These reference IDs are then cross-referenced in the template itself as well as the WebSphere II Q replication configuration screens and scripts. For example, in Table 5-2 on page 426 on WebSphere MQ information, for the XMITQ parameter (reference ID A.18a for Server A) in the SendQ, we require reference ID A.13 (which is the name of the TransmitQ), and that value is QM_WEST.

Attention: In Figure 5-3, there appear to be two sets of transmission queues, and sender and receiver channels, on each server. However, there is only one set on each server, as can be deduced from the identical names. Figure 5-3 has the appearance of two sets so that the flow of data and messages between the two servers is easily understood.

Once all the information identified in the template had been collected, we can proceed with the actual implementation.

Note: The template shows certain parameter values that are required for Q replication (such as parameter DEFPSIST *must be* set to YES for the TransmitQ [reference ID A.13] and does *not* have a reference ID), while others can be customized for a particular environment (such as reference ID A.13d parameter MAXDEPTH *may be* set to any value).

We have collected all the identified information for the Corona bidirectional replication, and implemented and recorded them in the templates Table 5-1 on

page 425, Table 5-2 on page 426, Table 5-3 on page 428, and Table 5-4 on page 428. We are now ready to proceed with configuring the various resources.

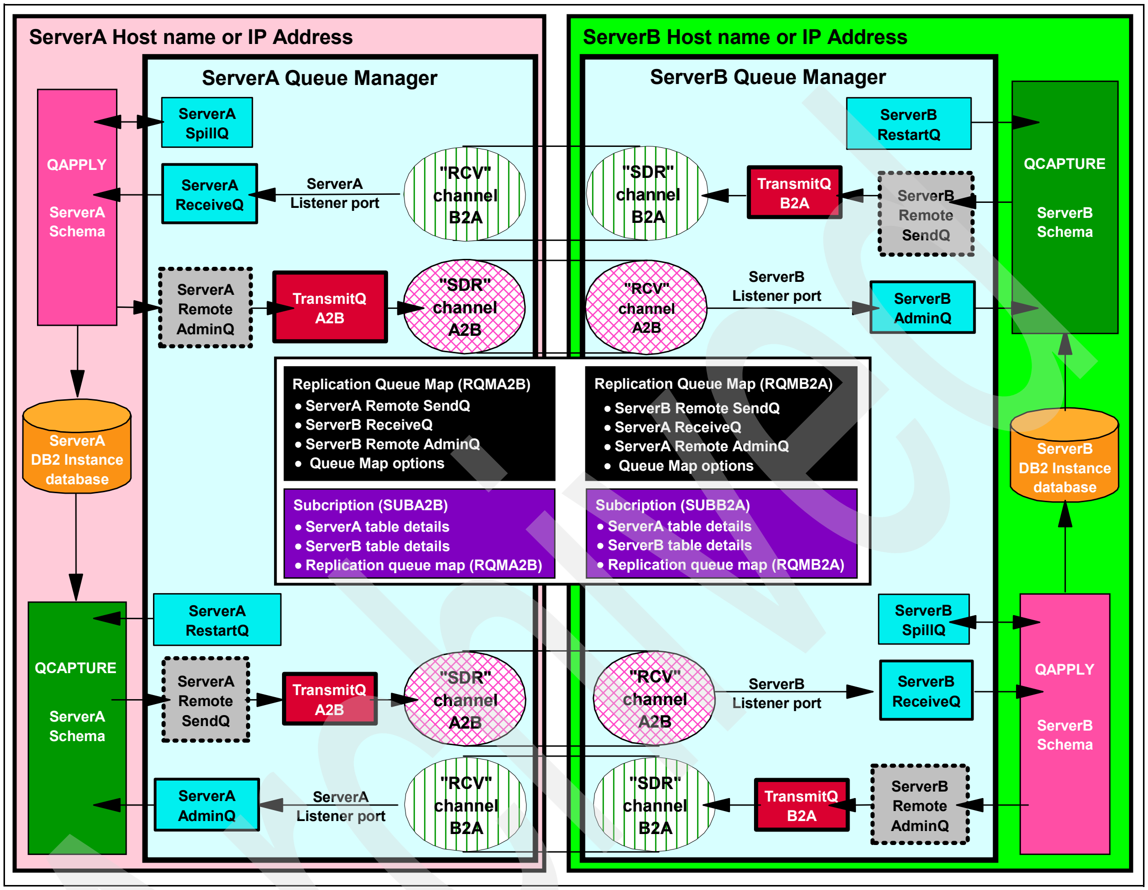


Figure 5-3 Bidirectional replication topology objects overview

Table 5-1 Host and DB2 system information

Description	Server A	Server B
Host name	(A.1) JAMESBAY	(B.1) AZOV
Host IP address	(A.2) 9.1.39.79	(B.2) 9.1.39.89
DB2 subsystem name or instance owner	(A.3) db2inst1	(B.3) db2inst1
DB2 group name (z/OS only)	(A.4)	(B.4)
DB2 location name (z/OS only)	(A.5)	(B.5)

Description	Server A	Server B
DB2 tcp port	(A.6) 50000	(B.6) 50000
Database server/alias information ▶ Database server/alias ▶ Q Capture user ID and password ▶ Q Apply user ID and password	 ▶ (A.7a) EAST_DB ▶ (A.7b) qrepladm/xxx ▶ (A.7c) qrepladm/xxx	 ▶ (B.7a) WEST_DB ▶ (B.7b) qrepladm/xxx ▶ (B.7c) qrepladm/xxx
User ID group (Unix only)	(A.8) qreplgrp	(B.8) qreplgrp
Other configuration user IDs requiring access	(A.9)	(B.9)
Logical database for control tables (z/OS only)	(A.10)	(B.10)

Table 5-2 WebSphere MQ information

Description	Server A	Server B
Queue Manager ▶ DEADQ (optional) ▶ MAXUMSGS (or use default) ▶ MAXMSGL (or use default)	(A.11) QM_EAST ▶ (A.11a) ▶ (A.11b) 10000 ▶ (A.11c) 4194304	(B.11) QM_WEST ▶ (B.11a) ▶ (B.11b) 10000 ▶ (B.11c) 4194304
Listener port	(A.12) 1451	(B.12) 1450
TransmitQ ▶ USAGE ▶ PUT ▶ GET ▶ DEFPSIST ▶ TRIGGER ▶ TRIGTYPE ▶ TRIGDATA ▶ INITQ ▶ MAXMSGL (or use default) ▶ MAXDEPTH (or use default)	(A.13) QM_WEST ▶ XMITQ ▶ ENABLED ▶ ENABLED ▶ YES ▶ YES ▶ FIRST ▶ (A.13a)=(A14) EAST_TO_WEST ▶ (A.13b) SYSTEM.CHANNEL.INITQ ▶ (A.13c) 4194304 ▶ (A.13d) 5000	(B.13) QM_EAST ▶ XMITQ ▶ ENABLED ▶ ENABLED ▶ YES ▶ YES ▶ FIRST ▶ (B.13a)=(B14) WEST_TO_EAST ▶ (B.13b) SYSTEM.CHANNEL.INITQ ▶ (B.13c) 4194304 ▶ (B.13d) 5000
SDR channel ▶ CHLTYPE ▶ TRPTYPE ▶ XMITQ ▶ CONNAME ▶ HBINT (or use default)	(A.14) EAST_TO_WEST ▶ SDR ▶ (A.14a) TCP ▶ (A.14b)=(A.13) QM_WEST ▶ (A.14c) 9.1.39.89 (1450) ▶ (A.14d) 300	(B.14) WEST_TO_EAST ▶ SDR ▶ (B.14a) TCP ▶ (B.14b)=(B.13) QM_EAST ▶ (B.14c) 9.1.39.79 (1451) ▶ (B.14d) 300
RCV channel ▶ CHLTYPE ▶ TRPTYPE ▶ HBINT (or use default)	(A.15)=(B.14) WEST_TO_EAST ▶ RCVR ▶ (A.15a) TCP ▶ (A.15b) 300	(B.15)=(A.14) EAST_TO_WEST ▶ RCVR ▶ (B.15a) TCP ▶ (B.15b) 300

Description	Server A	Server B
RestartQ ► PUT ► GET ► SHARE ► DEFSOPT ► DEFPSIST ► MAXMSGL (or use default) ► MAXDEPTH (or use default)	(A.16) EAST_RESTARTQ ► ENABLED ► ENABLED ► YES ► SHARED ► YES ► (A.16a) 4194304 ► (A.16b) 5000	(B.16) WEST_RESTARTQ ► ENABLED ► ENABLED ► YES ► SHARED ► YES ► (B.16a) 4194304 ► (B.16b) 5000
AdminQ ► PUT ► GET ► SHARE ► DEFSOPT ► DEFPSIST ► MAXMSGL (or use default) ► MAXDEPTH (or use default)	(A.17) EAST_ADMINQ ► ENABLED ► ENABLED ► YES ► SHARED ► YES ► (A.17a) 4194304 ► (A.17b) 5000	(B.17) WEST_ADMINQ ► ENABLED ► ENABLED ► YES ► SHARED ► YES ► (B.17a) 4194304 ► (B.17b) 5000
SendQ (remote) ► PUT ► DEFPSIST ► XMITQ ► RNAME ► RQMNAME	(A.18) EAST_TO_WEST_Q ► ENABLED ► YES ► (A.18a)=(A.13) QM_WEST ► (A.18b)=(B.19) EAST_TO_WEST_Q ► (A.18c)=(B.11) QM_WEST	(B.18) WEST_TO_EAST_Q ► ENABLED ► YES ► (B.18a)=(B.13) QM_EAST ► (B.18b)=(A.19) WEST_TO_EAST_Q ► (B.18c)=(A.11) QM_EAST
ReceiveQ (local) ► PUT ► GET ► SHARE ► DEFSOPT ► DEFPSIST ► INDXTYPE ► MSGDLVSQ ► MAXMSGL (or use default) ► MAXDEPTH (or use default)	(A.19) = (B.18b) WEST_TO_EAST_Q ► ENABLED ► ENABLED ► YES ► SHARED ► YES ► MSGID ► PRIORITY ► (A.19a) 4194304 ► (A.19b) 5000	(B.19) = (A.18b) EAST_TO_WEST_Q ► ENABLED ► ENABLED ► YES ► SHARED ► YES ► MSGID ► PRIORITY ► (B.19a) 4194304 ► (B.19b) 5000
SpillQ ► DEFTYPE ► DEFSOPT ► MSGDLVSQ ► MAXMSGL (or use default) ► MAXDEPTH (or use default)	IBMQREP.SPILL.MODELQ ► PERMDYN ► SHARED ► FIFO ► (A.20a) 4194304 ► (A.20b) 5000	IBMQREP.SPILL.MODELQ ► PERMDYN ► SHARED ► FIFO ► (B.20a) 4194304 ► (B.20b) 5000

Description	Server A	Server B
AdminQ (remote) ▸ PUT ▸ DEFPSIST ▸ XMITQ ▸ RNAME ▸ RQMNAME	(A.21) = (B.17) WEST_ADMINQ ▸ ENABLED ▸ YES ▸ (A.21a)=(A.13) QM_WEST ▸ (A.21b)=(B.17) WEST_ADMINQ ▸ (A.21c)=(B.11) QM_WEST	(B.21) = (A.17) EAST_ADMINQ ▸ ENABLED ▸ YES ▸ (B.21a)=(B.13) QM_EAST ▸ (B.21b)=(A.17) EAST_ADMINQ ▸ (B.21c)=(A.11) QM_EAST

Table 5-3 Q replication configuration information

Description	Server A	Server B
QCAPTURE Schema	(A.22) EAST	(B.22) WEST
QAPPLY Schema	(A.23) EAST	(B.23) WEST
Q Capture path (log files)	(A.24) /db2_data/capture	(B.24) /db2_data/capture
Q Apply path (log files)	(A.25) /db2_data/apply	(B.25) /db2_data/apply
Q replication type (U/B/P)	(A.26) B	(B.26) B
Replication Queue Map name ▸ Max message length (KB) ▸ Error handling ▸ Num apply agents ▸ Memory buffer for Recvq (MB) ▸ Allow QCapture to send heartbeat ▸ Heartbeat interval (sec)	(A.27) E_TO_W_MAP ▸ (A.27a) 64 ▸ (A.27b) Stop Q Capture ▸ (A.27c) 16 ▸ (A.27d) 2 ▸ (A.27e) yes ▸ (A.27f) 60	(B.27) W_TO_E_MAP ▸ (B.27a) 64 ▸ (B.27b) Stop Q Capture ▸ (B.27c) 16 ▸ (B.27d) 2 ▸ (B.27e) yes ▸ (B.27f) 60
Q Subscriptions parameters ▸ Target table schema/creator ▸ Target table index schema/name ▸ Target tablespaces ▸ Check for conflicts setting ▸ Conflict resolution action ▸ Error response action ▸ Initial load option ▸ Source server for initial load ▸ Qsub start automatic	▸ (A.28a) same ▸ (A.28b) same ▸ (A.28c) each gets own ▸ (A.28d) all columns ▸ (A.28e) EAST_DB ▸ (A.28f) Stop receive queue ▸ (A.28g) Automatic ▸ (A.28h) EAST_DB ▸ (A.28i) yes	▸ (B.28a) same ▸ (B.28b) same ▸ (B.28c) each gets own ▸ (B.28d) all columns ▸ (B.28e) EAST_DB ▸ (B.28f) Stop receive queue ▸ (B.28g) Automatic ▸ (B.28h) EAST_DB ▸ (B.28i) yes
Q Subscriptions list	(A.29)	(B.29)

Table 5-4 Replication Alert Monitor configuration information

Description	Designated server
Host name	(M.1) KANAGA
Host IP address	(M.2) 9.1.39.90

Description	Designated server
DB2 subsystem name or instance owner	(M.3) db2inst1
DB2 tcp port	(M.4) 50000
Monitor database server/alias information ► Database server/alias ► db2instance ► User ID/password ► Contact name ► Contact e-mail address ► Monitor qualifier ► Schema name	 ► (M.5a) MONDB ► (M.5b) DB2 ► (M.5c) qrepladm/xxx ► (M.5d) Qrepl Admin ► (M.5e) qrepladm@kanaga.almaden.ibm.com ► (M.5f) MON ► ASN
Monitor log path	(M.6) /db2_data/MON/log
DB2 group name (z/OS only)	(M.7)
DB2 location name (z/OS only)	(M.8)

5.5.4 Step 4: Set up user IDs, privileges, and database servers

The user IDs that run the Q replication and event publishing programs need authority to connect to servers, access or update tables, put and get from WebSphere MQ queues, and perform other operations such as write to directories. In addition, appropriate databases to be used in replication need to be created and cataloged for access by the Replication Center. The steps involved are described here.

User IDs and privileges

User IDs need to be defined for executing Q Capture (reference A.7b in Table 5-1 on page 425), and Q Apply (reference A.7c in Table 5-1 on page 425).

- Q Capture

  All user IDs that run a Q Capture program must have authorization to access the DB2® system catalog, access and update all Q Capture control tables, read the DB2 log, and run the Q Capture program packages. The following list summarizes the DB2 UDB requirements and operating system requirements for the UNIX® platform.

  – DBADM or SYSADM authority on the databases involved.

    In a massively parallel processing (MPP) configuration, the user IDs must be able to connect to database partitions and read the password file.

  – WRITE privilege on the directory that is specified by the CAPTURE_PATH parameter. The Q Capture program creates diagnostic files in this directory.

- MQM[2] privileges on the WebSphere MQ queues or necessary privileges to operate each queue and queue manager for Q Replication.

- Q Apply

  All user IDs that run a Q Apply program must have authorization to access the DB2® system catalog, access and update targets, access and update the Q Apply control tables, run the Q Apply program packages, and read the Q Apply password file.

  The following list summarizes the DB2 UDB requirements, operating system requirements, and WebSphere MQ requirements for the UNIX® platform.

  - DBADM or SYSADM authority.
  - SELECT privilege for the source tables if the Q Apply program will be used to load target tables.

    If the Q Apply program uses the LOAD from CURSOR option of the LOAD utility to load target tables, the Q Apply server must be a federated server, and one must create nicknames, server definitions, and user mappings on the Q Apply server. The user ID that is supplied in the user mappings must have privilege to read from nicknames on the federated Q Apply server and read from the source tables.

  - WRITE privilege on the directory that is specified by the APPLY_PATH parameter. The Q Apply program creates diagnostic files in this directory.
  - MQM privileges on the WebSphere MQ queues or necessary privileges to operate each queue and queue manager for Q Replication.

The following steps created the user IDs and assigned them the required system and database privileges:

1. We created a single user ID, qrepladm (references A.7b, A.7c, B.7b, and B.7c), and assigned it to group qreplgrp (reference A.8 and B.8), as shown in Example 5-1, on each of the servers Jamesbay and Azov.

Example 5-1 Create group qreplgrp and user ID qrepladm and associate them

```
##############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 5-1 on page 425, and Table 5-2 on page 426
#############################################################
##QREPLGRP from reference A.8 and B.8 in Table 5-2 on page 426
##QREPLADM from reference A.7b and B.7b in Table 5-1 on page 425
##############################################################
mkgroup qreplgrp
mkuser qrepladm
```

2 The MQM privilege in WebSphere MQ is equivalent to having the SYSADM privilege in DB2.

```
chuser groups=qreplgrp, qrepladm
```

2. We created directories /db2_data/capture (reference A.24 and B.24) and /db2_data/apply (reference A.25 and B.25) where the Q Capture and Q Apply programs will operate, and need to have write permissions (as shown in Example 5-2) on each of the servers Jamesbay and Azov.

   Note: Since we planned to automatically load the tables for our subscriptions, we needed to ensure that the DB2 instance owner db2inst1 (reference A.3 and B.3) has full permissions in the APPLY_PATH directory so that it can create subdirectories for all the data files.

   We therefore chose to allow public read/write access to these directories.

Example 5-2 Create Q Capture and Q Apply directories and give write permission to it

```
#############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 5-3 on page 428
############################################################
##/db2_data/capture from reference A.24 and B.24 in Table 5-3 on page 428
##/db2_data/apply from reference A.25 and B.25 in Table 5-3 on page 428
#############################################################
mkdir /db2_data/capture
mkdir /db2_data/apply
chmod 777 /db2_data/capture
chmod 777 /db2_data/apply
```

3. We then granted user ID qrepladm on each server DBADM and SYSCTRL authority on the respective database—on server Jamesbay, it is on EAST_DB; while on server Azov, it is on WEST_DB (as shown in Example 5-3 and Example 5-4, respectively).

   Note: SYSADM privilege may be granted instead of DBADM and SYSCTRL.

Example 5-3 Grant DBADM and SYSCTRL privilege on EAST_DB to qrepladm

```
#############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 5-1 on page 425
############################################################
##EAST_DB from reference A.7a in Table 5-1 on page 425
##QREPLADM from reference A.7b and A.7c in Table 5-1 on page 425
```

```
##QREPLGRP from reference A.8 in Table 5-1 on page 425
#############################################################
db2 connect to EAST_DB
db2 grant DBADM on DATABASE to USER qrepladm
db2 update dbm cfg using SYSCTRL_GROUP qreplgrp
```

Example 5-4 Grant DBADM and SYSCTRL privilege on WEST_DB to qrepladm

```
#############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 5-1 on page 425
############################################################
##WEST_DB from reference B.7a in Table 5-1 on page 425
##QREPLADM from reference B.7b and A.7c in Table 5-1 on page 425
##QREPLGRP from reference B.8 in Table 5-1 on page 425
#############################################################
db2 connect to WEST_DB
db2 grant DBADM on DATABASE to USER qrepladm
db2 update dbm cfg using SYSCTRL_GROUP qreplgrp
```

4. The user IDs that will be used to run the Q replication Apply and Capture programs need to have adequate MQ authorities for all WebSphere MQ objects. This may be accomplished by either:

   – Adding the user IDs to the mqm group.

     We granted mqm privilege to qrepladm on each of the servers Jamesbay and Azov, as shown in Example 5-5, to allow the Q Capture and Q Apply programs to operate the WebSphere MQ queues. This grants the IDs full administrative authority over all MQ objects.

Example 5-5 Add user ID qrepladm to mqm group

```
#############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 5-1 on page 425
############################################################
##QREPLGRP from reference A.8 and B.8 in Table 5-1 on page 425
##QREPLADM from reference A.7b and B.7b in Table 5-1 on page 425
#############################################################
chuser groups=qreplgrp,mqm qrepladm
```

   – Granting individual authorities to the specific objects required.

     For environments where granting the mqm privilege may not be acceptable, Example 5-6 and Example 5-7 provide details on granting individual privileges instead of mqm privileges.

Note: This assumes that the WebSphere MQ objects referenced in Example 5-6 and Example 5-7 already exist. In our case, these WebSphere MQ objects are defined in 5.5.5, “Step 5: Configure WebSphere MQ” on page 436.

For full details on these privileges refer to Chapter 10 of the *WebSphere MQ System Administration Guide*, SC34-6068-02.

Example 5-6 Grant individual privileges on node EAST on server Jamesbay

```
#############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 5-1 on page 425, and Table 5-2 on page 426
############################################################
##QM_EAST from reference A.11 in Table 5-2 on page 426
##QREPLADM from reference A.7b in Table 5-1 on page 425
##WEST_TO_EAST_Q from reference A.19 in Table 5-2 on page 426
##EAST_TO_WEST_Q from reference A.18 in Table 5-2 on page 426
##EAST_ADMINQ from reference A.17 in Table 5-2 on page 426
##EAST_RESTARTQ from reference A.16 in Table 5-2 on page 426
##QM_WEST from reference B.11 in Table 5-2 on page 426
##WEST_ADMINQ from reference B.17 in Table 5-2 on page 426
##QREPLGRP from reference A.8 in Table 5-1 on page 425
#############################################################
setmqaut -m QM_EAST -t qmgr -p qrepladm +connect +inq
setmqaut -m QM_EAST -t queue -n WEST_TO_EAST_Q -p qrepladm +inq +get +browse
setmqaut -m QM_EAST -t queue -n IBMQREP.SPILL.MODELQ -p qrepladm +inq +put
setmqaut -m QM_EAST -t queue -n EAST_TO_WEST_Q -p qrepladm +inq +put
setmqaut -m QM_EAST -t queue -n EAST_ADMINQ -p qrepladm +inq +get +browse +put
setmqaut -m QM_EAST -t queue -n EAST_RESTARTQ -p qrepladm +inq +get +put
+browse
setmqaut -m QM_EAST -t queue -n QM_WEST -p qrepladm +put +inq
setmqaut -m QM_EAST -t queue -n WEST_ADMINQ -p qrepladm +inq +put
setmqaut -m QM_EAST -t queue -n IBMQREP.SPILL.MODELQ -g qreplgrp +allmqi +dlt
+chg +dsp +clr
```

Example 5-7 Grant individual privileges on WEST on server Azov

```
#############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 5-1 on page 425, and Table 5-2 on page 426
############################################################
##QM_WEST from reference B.11 in Table 5-2 on page 426
##QREPLADM from reference B.7b in Table 5-1 on page 425
##EAST_TO_WEST_Q from reference A.18 in Table 5-2 on page 426
```

```
##WEST_TO_EAST_Q from reference A.19 in Table 5-2 on page 426
##WEST_ADMINQ from reference B.17 in Table 5-2 on page 426
##WEST_RESTARTQ from reference B.16 in Table 5-2 on page 426
##QM_EAST from reference A.11 in Table 5-2 on page 426
##EAST_ADMINQ from reference A.17 in Table 5-2 on page 426
##QREPLGRP from reference B.8 in Table 5-1 on page 425
#############################################################
setmqaut -m QM_WEST -t qmgr -p qrepladm +connect +inq
setmqaut -m QM_WEST -t queue -n EAST_TO_WEST_Q -p qrepladm +inq +get +browse
setmqaut -m QM_WEST -t queue -n IBMQREP.SPILL.MODELQ -p qrepladm +inq +put
setmqaut -m QM_WEST -t queue -n WEST_TO_EAST_Q -p qrepladm +inq +put
setmqaut -m QM_WEST -t queue -n WEST_ADMINQ -p qrepladm +inq +get +browse +put
setmqaut -m QM_WEST -t queue -n WEST_RESTARTQ -p qrepladm +inq +get +put
+browse
setmqaut -m QM_WEST -t queue -n QM_EAST -p qrepladm +put +inq
setmqaut -m QM_WEST -t queue -n EAST_ADMINQ -p qrepladm +inq +put
setmqaut -m QM_WEST -t queue -n IBMQREP.SPILL.MODELQ -g qreplgrp +allmqi +dlt
+chg +dsp +clr
```

Database servers

The following actions need to be performed to set up the database servers:

1. In the AIX environment, we need to set the EXTSHM[3] environment variable in the DB2 instance owner environment, as shown in Example 5-8.

Example 5-8 Set EXTSHM environment variable

```
EXTSHM=ON; export EXTSHM
db2set DB2ENVLIST=EXTSHM
```

2. We assume that the databases used for source (EAST_DB reference A.7a) and target (WEST_DB reference B.7a) of bidirectional Q replication already exist on servers Jamesbay and Azov, respectively. We also assume that these databases are recoverable—they have the database configuration parameter logretain set to recovery. If not, set it for each database, as shown in Example 5-9.

Example 5-9 Setting logretain to recovery

```
db2 update db cfg using logretain recovery;
```

   After executing this command, the database needs to be backed up using the command similar to that shown in Example 5-10.

[3] In a 32-bit AIX environment, the EXTSHM environment variable allows a process to attach to more than 11 shared memory segments. When this environment variable is set to ON, one can attach multiple shared memory segments within the same 256 MB region. Rather than beginning on a region boundary, these shared memory segments begin on a (typically 4 K) page boundary, which allows one to create thousands of shared memory segments per region.

Example 5-10 Backup database

```
#############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 5-1 on page 425
############################################################
##EAST_DB from reference A.7a in Table 5-1 on page 425
#############################################################
db2 backup db EAST_DB to /Backup
```

3. The EAST_DB database on server Jamesbay and WEST_DB database on server Azov must be cataloged on server Azov and server Jamesbay, respectively, as shown in Example 5-11 and Example 5-12.

Example 5-11 Catalog EAST_DB on server Azov

```
#############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 5-1 on page 425
############################################################
##EAST is a madeup name - not referenced in the template
##9.1.39.79 from reference A.2 in Table 5-1 on page 425
##50000 from reference B.6 in Table 5-1 on page 425
##EAST_DB from reference A.7a in Table 5-1 on page 425
#############################################################
db2 catalog tcpip node EAST remote 9.1.39.79 server 50000
db2 catalog database EAST_DB at node EAST
db2 terminate;
```

Example 5-12 Catalog WEST_DB on server Jamesbay

```
#############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 5-1 on page 425
############################################################
##WEST is a madeup name - not referenced in the template
##9.1.39.89 from reference B.2 in Table 5-1 on page 425
##50000 from reference A.6 in Table 5-1 on page 425
##WEST_DB from reference B.7a in Table 5-1 on page 425
#############################################################
db2 catalog tcpip node WEST remote 9.1.39.89 server 50000
db2 catalog database WEST_DB at node WEST
db2 terminate;
```

4. We tested the connectivity to WEST_DB and EAST_DB from servers Jamesbay and Azov, respectively, as shown in Example 5-13 and Example 5-14.

Example 5-13 Test connectivity to WEST_DB from server Jamesbay

```
#############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 5-1 on page 425
############################################################
##WEST_DB from reference B.7a in Table 5-1 on page 425
##QREPLADM from reference B.7b in Table 5-1 on page 425
#############################################################
db2 connect to WEST_DB user qrepladm using xxx
db2 terminate;
```

Example 5-14 Test connectivity to EAST_DB from server Azov

```
#############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 5-1 on page 425
############################################################
##EAST_DB from reference A.7a in Table 5-1 on page 425
##QREPLADM from reference A.7b in Table 5-1 on page 425
#############################################################
db2 connect to EAST_DB user qrepladm using xxx
db2 terminate;
```

5.5.5 Step 5: Configure WebSphere MQ

Several WebSphere MQ objects need to be created in support of Q replication, and configured to suit an organization's particular workload. The WebSphere MQ objects are:

- Queue Manager
- Channels
- Local queues
- Remote queues

There is a hierarchical relationship between these objects—the Queue Manager is created at the highest level and all objects are managed by the Queue Manager.

To support a typical Q replication bidirectional set up, the following objects must be defined on each system:

- Queue Manager
- Sender channel
- Receiver channel
- Transmit queue (local)
- Capture administration queue (local)
- Capture restart queue (local)
- Capture send queue (remote)
- Apply receive queue (local)
- Apply spill queue (local model)
- Apply administration queue (remote)

> **Note:** *Local* refers to the fact that the queue is defined as a local queue on the system, while *remote* refers to the fact that the queue is defined as a remote queue.

After configuring the WebSphere MQ objects, we verified its success by testing the put and get of messages from the local and remote queues.

Configure WebSphere MQ objects

Example 5-15 is the script for creating all the WebSphere MQ objects necessary for bidirectional replication on node EAST (Jamesbay), while Example 5-15 is the script for creating the objects on node WEST (Azov).

> **Important:** The scripts should be run by a member of the mqm group.

> **Note:** The values for the names of the WebSphere MQ objects and the configuration parameters are substituted from the templates listed in Table 5-1 on page 425, Table 5-2 on page 426, and Table 5-3 on page 428. The specific references are listed in the script.

Example 5-15 Create necessary MQ objects for bidirectional replication on EAST

```
#################################################################
# Set up the EAST server
#################################################################
#
# This section ensures that any existing MQ listener and Queue
# Manager are stopped and deleted before attempting to recreate
# them.
#
# Uncomment the following lines if one wishes to rerun this script
```

```
# to recreate all objects, after an earlier run                     #

#endmqm QM_EAST
#sleep 30
#endmqlsr
#sleep 20
#dltmqm QM_EAST

######################################################
# 1) Create the Queue Manager on EAST
######################################################
##Queue Manager reference A.11

crtmqm -q QM_EAST

######################################################
# 2) Start the Queue Manager on EAST
#####################################################

strmqm

######################################################
# 3) Start a WebSphere MQ listener
#####################################################
##WebSphere MQ listener port reference A.12

runmqlsr -t tcp -p 1451 &

######################################################
# 4) Create the queue/channel objects
######################################################
##Local AdminQ reference A.17
##Local AdminQ MAXMSGL reference A.17a
##Local AdminQ MAXDEPTH reference A.17b

runmqsc <<!

DEFINE QLOCAL(EAST_ADMINQ) +
REPLACE +
DESCR('LOCAL DEFN OF ADMINQ FOR EAST CAPTURE') +
PUT(ENABLED) +
GET(ENABLED) +
DEFPSIST(YES) +
MAXMSGL(4194403) +
MAXDEPTH(5000)
##
##Local RestartQ reference A.16
##Local RestartQ MAXMSGL reference A.16a
##Local RestartQ MAXDEPTH reference A.16b
```

```
DEFINE QLOCAL(EAST_RESTARTQ) +
REPLACE +
DESCR('LOCAL DEFN OF RESTART FOR EAST CAPTURE') +
PUT(ENABLED) +
GET(ENABLED) +
DEFPSIST(YES)  +
MAXMSGL(4194403) +
MAXDEPTH(5000)
##
##Local SpillQ reference A.20
##Local SpillQ MAXMSGL reference A.20a
##Local SpillQ MAXDEPTH reference A.20b

DEFINE QMODEL(IBMQREP.SPILL.MODELQ) +
REPLACE +
DEFSOPT(SHARED) +
MAXDEPTH(500000) +
MSGDLVSQ(FIFO) +
DEFTYPE(PERMDYN) +
MAXMSGL(100000)
##
##Remote SendQ QREMOTE reference A.18
##Remote SendQ XMITQ reference A.18a
##Remote SendQ RNAME reference A.18b
##Remote SendQ RQMNAME reference A.18c

DEFINE QREMOTE(EAST_TO_WEST_Q) +
REPLACE +
DESCR('REMOTE DEFN OF SEND QUEUE FROM EAST TO WEST') +
PUT(ENABLED) +
XMITQ(QM_WEST)  +
RNAME(EAST_TO_WEST_Q) +
RQMNAME(QM_WEST) +
DEFPSIST(YES)
##
##Local ReceiveQ QLOCAL reference A.19
##Local ReceiveQ MAXMSGL reference A.19a
##Local ReceiveQ MAXDEPTH reference A.19b

DEFINE QLOCAL(WEST_TO_EAST_Q) +
REPLACE +
DESCR('LOCAL RECEIVE QUEUE - EAST FROM WEST') +
GET(ENABLED) +
SHARE +
DEFPSIST(YES) +
MSGDLVSQ(PRIORITY) +
MAXMSGL(4194403) +
MAXDEPTH(5000)
```

```
##
##Remote AdminQ QREMOTE reference A.21
##Remote AdminQ XMITQ reference A.21a
##Remote AdminQ RNAME reference A.21b
##Remote AdminQ RQMNAME reference A.21c

DEFINE QREMOTE(WEST_ADMINQ) +
REPLACE +
DESCR('REMOTE DEFN OF ADMINQ FOR WEST CAPTURE') +
PUT(ENABLED) +
XMITQ(QM_WEST) +
RNAME(WEST_ADMINQ) +
RQMNAME(QM_WEST) +
DEFPSIST(YES)
##
##Local TransmitQ QLOCAL reference A.13
##Local TransmitQ TRIGDATA reference A.13a
##Local TransmitQ INITQ reference A.13b
##Local TransmitQ MAXMSGL reference A.13c
##Local TransmitQ MAXDEPTH reference A.13d

DEFINE QLOCAL(QM_WEST) +
REPLACE +
DESCR('TRANSMISSION QUEUE TO WEST') +
USAGE(XMITQ) +
DEFPSIST(YES) +
TRIGGER +
TRIGTYPE(FIRST) +
TRIGDATA(EAST_TO_WEST) +
INITQ(SYSTEM.CHANNEL.INITQ) +
MAXMSGL(4194403) +
MAXDEPTH(5000)
##
##Local SDR channel CHANNEL reference A.14
##Local SDR channel TRPTYPE reference A.14a
##Local SDR channel XMITQ reference A.14b
##Local SDR channel CONNAME reference A.14c
##Local SDR channel HBINT reference A.14d

DEFINE CHANNEL(EAST_TO_WEST) +
CHLTYPE(SDR) +
REPLACE +
TRPTYPE(TCP) +
DESCR('SENDER CHANNEL TO WEST') +
XMITQ(QM_WEST) +
CONNAME('9.1.39.89(1450)')  +
HBINT(300)
##
##Local RCV channel CHANNEL reference A.15
```

```
##Local RCV channel TRPTYPE reference A.15a
##Local RCV channel HBINT reference A.15b

DEFINE CHANNEL(WEST_TO_EAST) +
CHLTYPE(RCVR) +
REPLACE +
TRPTYPE(TCP) +
DESCR('RECEIVER CHANNEL FROM WEST') +
HBINT(300)
##
##MAXUMSGS reference A.11b
##MAXMSGL reference A.11c

ALTER QMGR +
MAXUMSGS(10000) +
MAXMSGL(4194403)

!

# END OF SCRIPT
```

Note: The same listener port can be used on both host servers, but it may be easier to debug if one uses a different port on each.

Example 5-16 Create necessary MQ objects for bidirectional replication on WEST

```
#################################################################
# Set up the WEST server
#################################################################
#
# This section ensures that any existing MQ listener and Queue
# Manager are stopped and deleted before attempting to recreate
# them.
#
# Uncomment the following lines if one wishes to rerun this script
# to recreate all objects, after an earlier run

#endmqm QM_WEST
#sleep 30
#endmqlsr
#sleep 20
#dltmqm QM_WEST

######################################################
# 1) Create the Queue Manager on WEST
######################################################
##Queue Manager reference B.11
```

```
crtmqm -q QM_WEST

####################################################
# 2) Start the Queue Manager on WEST
####################################################

strmqm

####################################################
# 3) Start a WebSphere MQ listener
####################################################
##WebSphere MQ listener port reference B.12

runmqlsr -t tcp -p 1450 &

####################################################
# 4) Create the queue/channel objects
###################################################
##Local AdminQ reference B.17
##Local AdminQ MAXMSGL reference B.17a
##Local AdminQ MAXDEPTH reference B.17b

runmqsc <<!

DEFINE QLOCAL(WEST_ADMINQ) +
REPLACE +
DESCR('LOCAL DEFN OF ADMINQ FOR WEST CAPTURE') +
PUT(ENABLED) +
GET(ENABLED) +
DEFPSIST(YES) +
MAXMSGL(4194403) +
MAXDEPTH(5000)

##Local RestartQ reference B.16
##Local RestartQ MAXMSGL reference B.16a
##Local RestartQ MAXDEPTH reference B.16b

DEFINE QLOCAL(WEST_RESTARTQ) +
REPLACE +
DESCR('LOCAL DEFN OF RESTART FOR WEST CAPTURE') +
PUT(ENABLED) +
GET(ENABLED) +
DEFPSIST(YES)    +
MAXMSGL(4194403) +
MAXDEPTH(5000)

##Local SpillQ reference B.20
##Local SpillQ MAXMSGL reference B.20a
##Local SpillQ MAXDEPTH reference B.20b
```

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

##
##Remote SendQ QREMOTE reference B.18
##Remote SendQ XMITQ reference B.18a
##Remote SendQ RNAME reference B.18b
##Remote SendQ RQMNAME reference B.18c

DEFINE QREMOTE(WEST_TO_EAST_Q) +
REPLACE +
DESCR('REMOTE DEFN OF SEND QUEUE FROM WEST TO EAST') +
PUT(ENABLED) +
XMITQ(QM_EAST)  +
RNAME(WEST_TO_EAST_Q) +
RQMNAME(QM_EAST) +
DEFPSIST(YES)
##
##Local ReceiveQ QLOCAL reference B.19
##Local ReceiveQ MAXMSGL reference B.19a
##Local ReceiveQ MAXDEPTH reference B.19b

DEFINE QLOCAL(EAST_TO_WEST_Q) +
REPLACE +
DESCR('LOCAL RECEIVE QUEUE - WEST FROM EAST') +
GET(ENABLED) +
SHARE +
DEFPSIST(YES) +
MSGDLVSQ(PRIORITY) +
MAXMSGL(4194403) +
MAXDEPTH(5000)
##
##Remote AdminQ QREMOTE reference B.21
##Remote AdminQ XMITQ reference B.21a
##Remote AdminQ RNAME reference B.21b
##Remote AdminQ RQMNAME reference B.21c

DEFINE QREMOTE(EAST_ADMINQ) +
REPLACE +
DESCR('REMOTE DEFN OF ADMINQ FOR EAST CAPTURE') +
PUT(ENABLED) +
XMITQ(QM_EAST) +
RNAME(EAST_ADMINQ) +
```

```
RQMNAME(QM_EAST) +
DEFPSIST(YES)
##
##Local TransmitQ QLOCAL reference B.13
##Local TransmitQ TRIGDATA reference B.13a
##Local TransmitQ INITQ reference B.13b
##Local TransmitQ MAXMSGL reference B.13c
##Local TransmitQ MAXDEPTH reference B.13d

DEFINE QLOCAL(QM_EAST) +
REPLACE +
DESCR('TRANSMISSION QUEUE TO EAST') +
USAGE(XMITQ) +
DEFPSIST(YES) +
TRIGGER +
TRIGTYPE(FIRST) +
TRIGDATA(WEST_TO_EAST) +
INITQ(SYSTEM.CHANNEL.INITQ) +
MAXMSGL(4194403) +
MAXDEPTH(5000)
##
##Local SDR channel CHANNEL reference B.14
##Local SDR channel TRPTYPE reference B.14a
##Local SDR channel XMITQ reference B.14b
##Local SDR channel CONNAME reference B.14c
##Local SDR channel HBINT reference B.14d

DEFINE CHANNEL(WEST_TO_EAST) +
CHLTYPE(SDR) +
REPLACE +
TRPTYPE(TCP) +
DESCR('SENDER CHANNEL TO EAST') +
XMITQ(QM_EAST) +
CONNAME('9.1.39.79(1451)') +
HBINT(300)
##
##Local RCV channel CHANNEL reference B.15
##Local RCV channel TRPTYPE reference B.15a
##Local RCV channel HBINT reference B.15b

DEFINE CHANNEL(EAST_TO_WEST) +
CHLTYPE(RCVR) +
REPLACE +
TRPTYPE(TCP) +
DESCR('RECEIVER CHANNEL FROM EAST') +
HBINT(300)
##
##MAXUMSGS reference B.11b
##MAXMSGL reference B.11c
```

```
ALTER QMGR +
MAXUMSGS(10000) +
MAXMSGL(4194403)

!

# END OF SCRIPT
```

Note: The same listener port can be used on both host servers, but it may be easier to debug if one uses a different port on each.

Verify successful WebSphere MQ configuration

After performing the initial configuration the channels must be started on each Queue Manager, and then messages may be inserted and retrieved from local and remote queues to verify successful configuration.

Example 5-17 and Example 5-18 show scripts that start the sender channel on node EAST and WEST, respectively.

Example 5-17 Start the sender channel on EAST

```
##################################################################
# Run on EAST
##################################################################
#
# Start the sender channel
#
runmqchl -c EAST_TO_WEST -m QM_EAST &
```

Example 5-18 Start the sender channel on WEST

```
##################################################################
# Run on WEST
##################################################################
#
# Start the sender channel
#
runmqchl -c WEST_TO_EAST -m QM_WEST &
```

After the channels have been started, some of the several queue pairs may be tested to ensure that communications are working properly. We tested local as well as remote queues as follows.

Testing local queues

Example 5-19 lists a local queue on the EAST and WEST nodes.

Example 5-19 Local queues

```
##################################################
Simple Queues:           amqsput/amqsget
##################################################
#
WEST_RESTARTQ            QM_WEST
EAST_RESTARTQ            QM_EAST
```

For each of these queues, we inserted a message (using /usr/mqm/samp/bin/amqsput) and read back the message (using /usr/mqm/samp/bin/amqsget) on the same node, as shown in Example 5-20. Testing local queues ensures that they are set up properly, but does not test the connectivity between the queue managers—for that we need to test the remote queues.

Example 5-20 Test put and get from a local queue

```
##################################################
# Execute the following command to put a message into the queue
##################################################
/usr/mqm/samp/bin/amqsput WEST_RESTARTQ QM_WEST
Sample AMQSPUT0 start
target queue is WEST_RESTARTQ
This is a test message to a local queue

Sample AMQSPUT0 end
#
##################################################
# Execute the following command to get the message from the queue.

# The program should display the text one entered above then pause
# for a few seconds waiting for more messages.
##################################################
#
/usr/mqm/samp/bin/amqsget WEST_RESTARTQ QM_WEST
Sample AMQSGET0 start
message <This is a test message to a local queue>
no more messages
Sample AMQSGET0 end
```

Testing remote queues

Example 5-21 lists remote queues on the EAST and WEST nodes. The queue manager listed next to each queue in Example 5-21 is the LOCAL side.

Example 5-21 Remote queues

```
##################################################
```

```
Queue Pairs:       amqsput side       amqsget side
####################################################
#
WEST_ADMINQ        QM_EAST            QM_WEST
WEST_TO_EAST_Q     QM_EAST            QM_WEST

EAST_ADMINQ        QM_WEST            QM_EAST
EAST_TO_WEST_Q     QM_WEST            QM_EAST
```

For each of these queues we inserted a message on the REMOTE side of the pair (using /usr/mqm/samp/bin/amqsput) and read it from the LOCAL side (using /usr/mqm/samp/bin/amqsget), as shown in Example 5-22. Testing these queues ensures that they are set up properly and that communications are operating properly between the queue managers.

Example 5-22 Test put and get from a remote queue

```
#####################################################
# Execute the following command to put a message into the queue
####################################################
#
/usr/mqm/samp/bin/amqsput WEST_ADMINQ QM_EAST
Sample AMQSPUT0 start
target queue is WEST_ADMINTQ
This is a test message to a remote queue

Sample AMQSPUT0 end
#
####################################################
# Execute the following command to get the message from the queue.
# The program should display # the text one entered above then pause
# for a few seconds waiting for more messages.
####################################################
#
/usr/mqm/samp/bin/amqsget WEST_ADMINQ QM_WEST
Sample AMQSGET0 start
message <This is a test message to a remote queue>
no more messages
Sample AMQSGET0 end
```

Note: The Model Queues (IBMQREP.SPILL.MODELQ) cannot be tested—they are only models from which queues are dynamically created during a load operation.

Attention: Ensure that one “gets” all the messages that one “puts”. Do not leave test messages in the queues prior to configuring Q replication.

5.5.6 Step 6: Configure and activate Q replication using GUI

In this section we document the step-by-step configuration of Q replication for the bidirectional replication topology in our fictitious company. Figure 5-4 expands “STEP 6: Configure & activate Q replication (GUI or commands)” (in Figure 5-2 on page 423) into a number of substeps involved in configuring Q replication. Each of these substeps is described in detail in the following subsections.

Very important: The Replication Center is typically used to configure and manage an SQL or Q replication environment because of its ease-of-use GUI interface. In order for a Replication Center client to be aware of the database servers that need to be defined as a first and second server in Figure 5-30 on page 475 in a replication environment, the Q Capture and Q Apply control tables *must* be created from the *same* Replication Center client. If the Q Capture and Q Apply control tables are created by ASNCLP scripts, or using *another* Replication Center client, then those database servers will *not* appear in the list of available servers for the first and second server selection. In such cases, you must catalog them in this Replication Center client using the process described in Appendix H, “Cataloging remote database servers” on page 881.

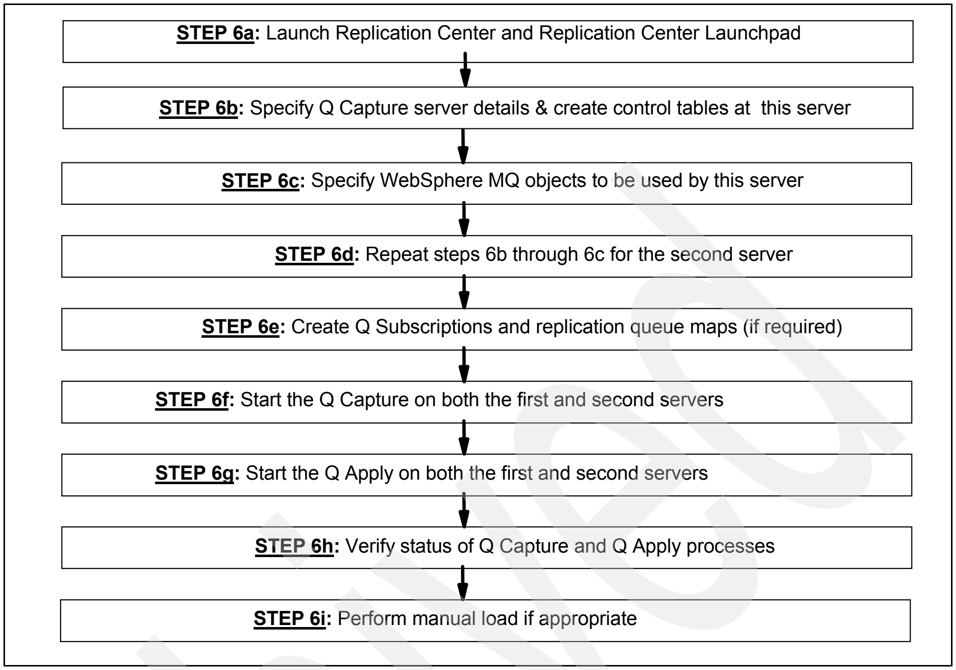


Figure 5-4 Overview of Q replication configuration steps

Step 6a: Launch Replication Center

Figure 5-5 shows the launching of the Replication Center from the DB2 Control Center by clicking the **Tools** tab and selecting **Replication Center**. This displays the Replication Center Launchpad screen shown in Figure 5-6.

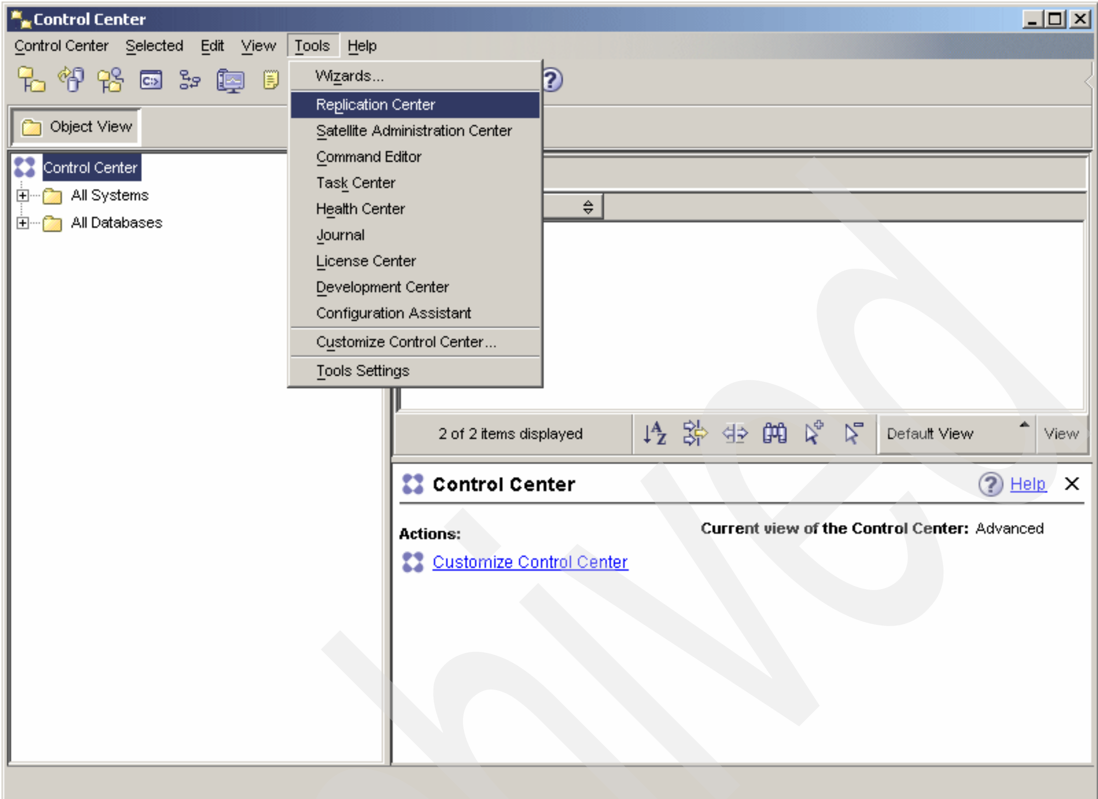


Figure 5-5 Launch Replication Center

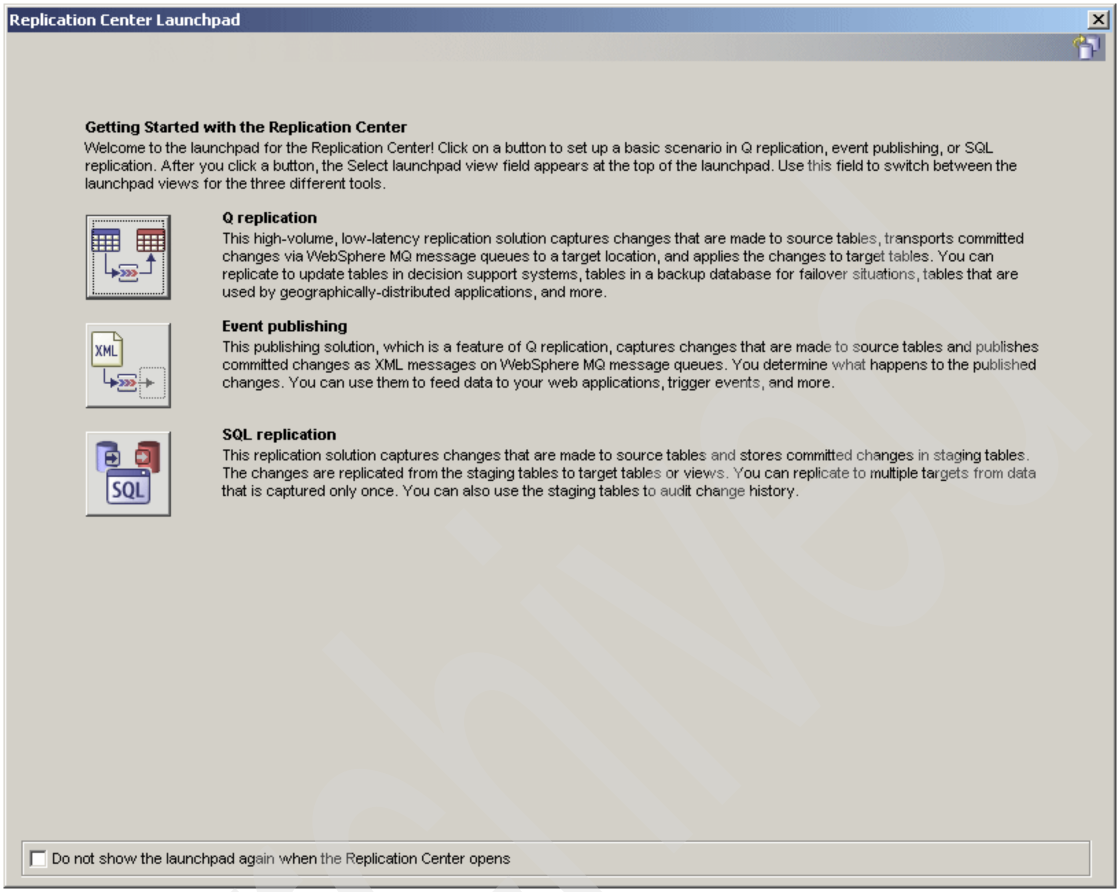


Figure 5-6 Replication Center Launchpad

> **Note:** Figure 5-6 can be skipped in the future by selecting the **Do not show this launchpad again when the Replication Center opens** option.

Step 6b: Specify Q Capture details

Click **Q replication** in Figure 5-6 to display Figure 5-7, which describes the five steps in setting up and activating Q replication, as follows:

1. Create Q Capture Control tables.
2. Create Q Apply control tables.
3. Create a Q Subscription.
4. Start a Q Capture program.
5. Start a Q Apply program.

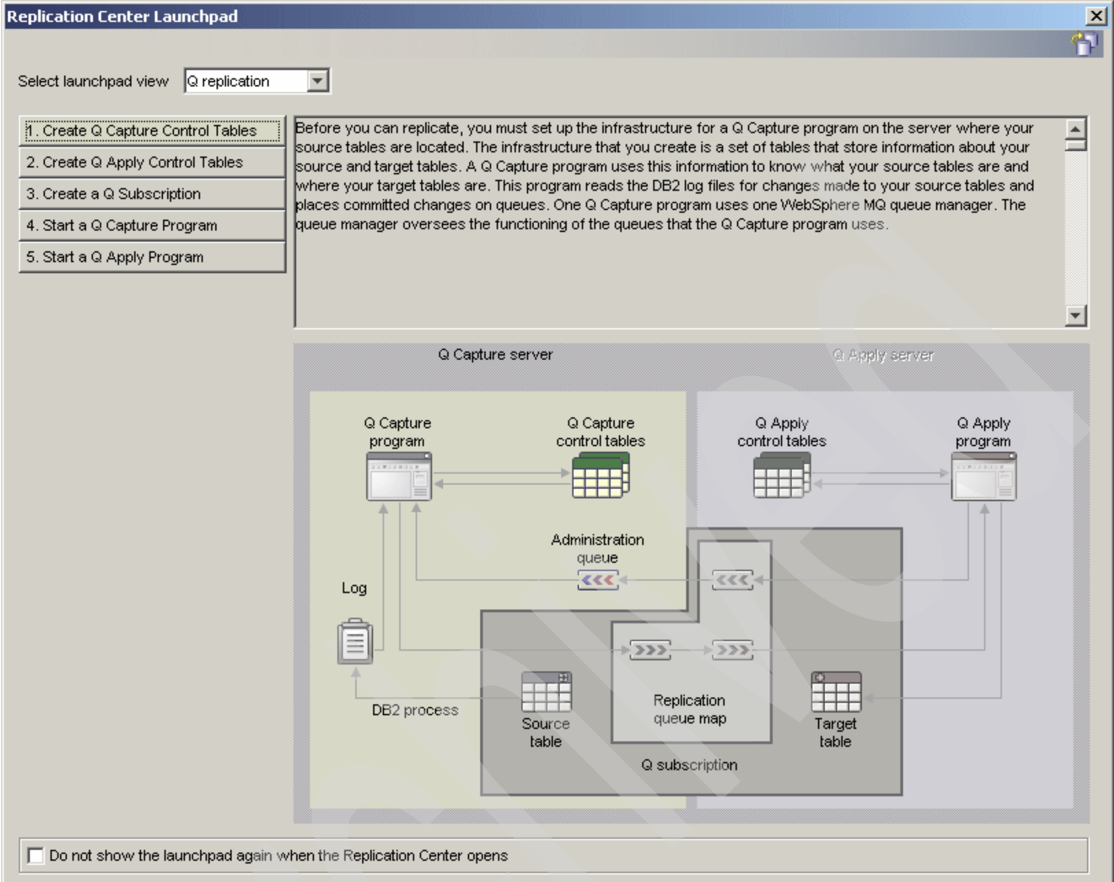


Figure 5-7 Five steps in setting up Q replication infrastructure

Click **Create Q Capture Control Tables** to display Figure 5-8, which enables the specification of details for the Q Capture infrastructure.

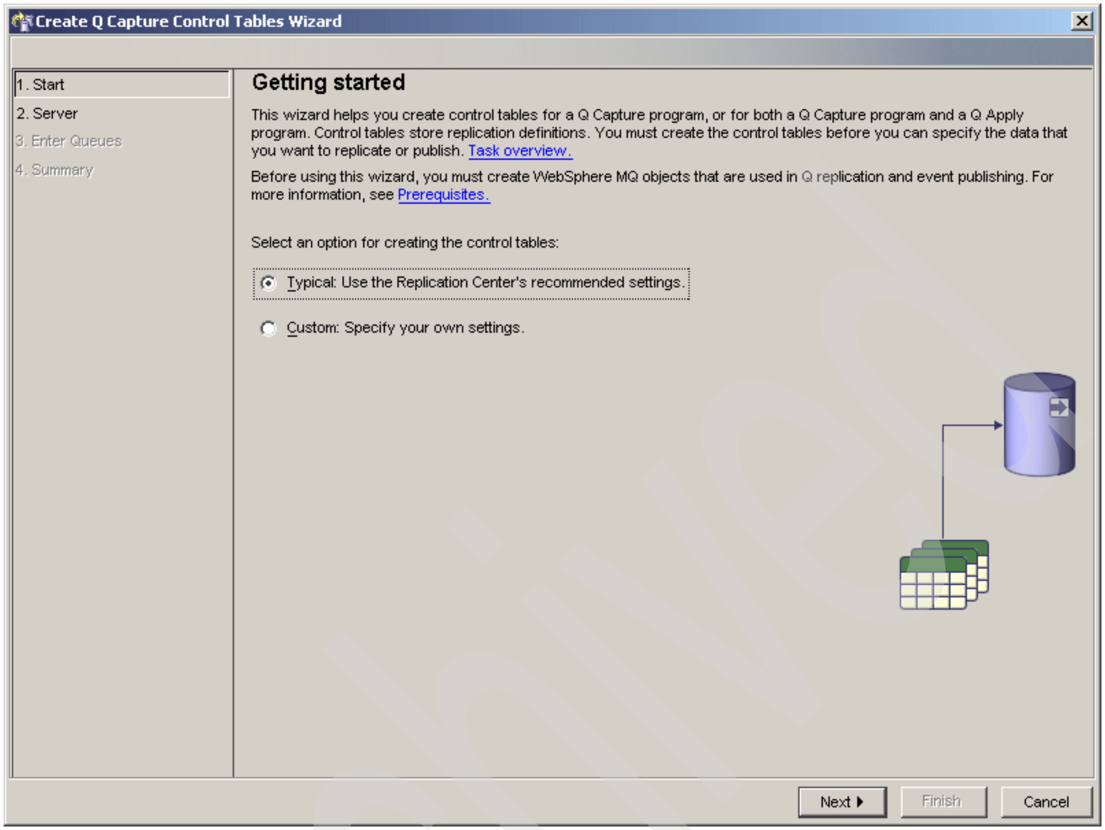


Figure 5-8 Getting started - Create Q Capture Control Tables Wizard

We selected the **Typical** setting and clicked **Next** to display Figure 5-9, where the Q Capture server details are specified.

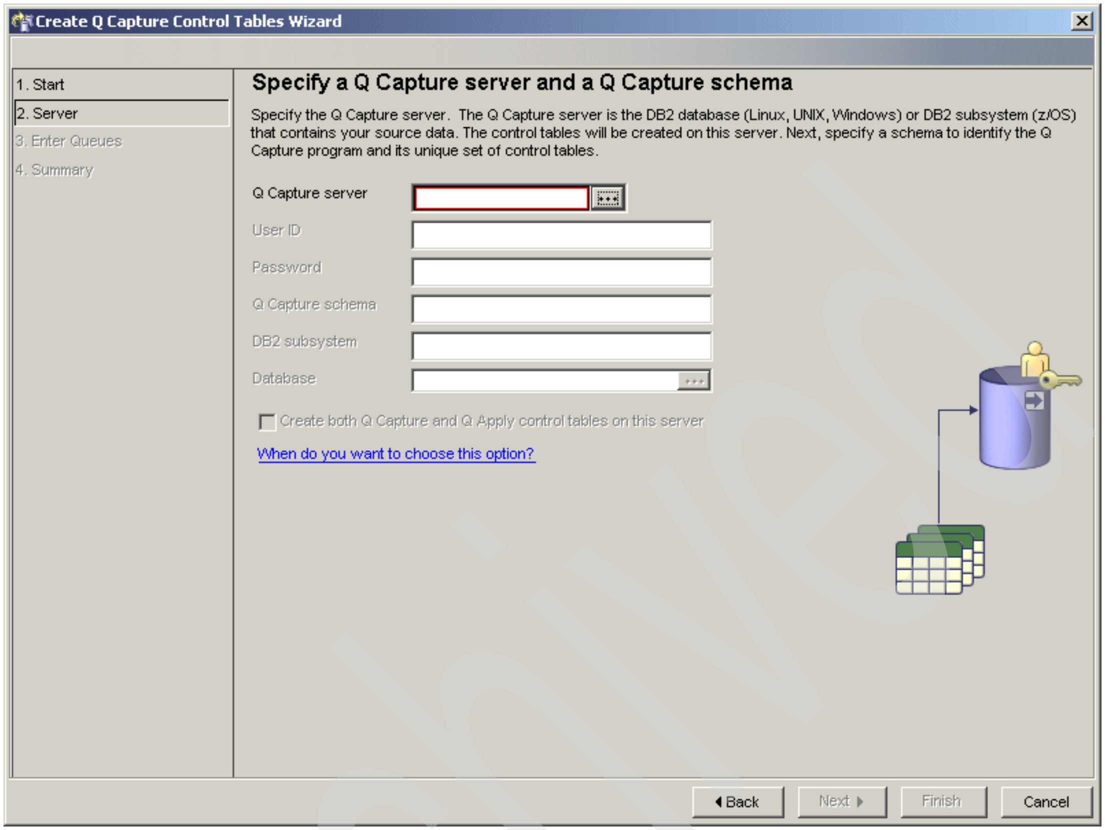


Figure 5-9 Server - Create Q Capture Control Tables Wizard

Click the **...** tab to display the list of available servers, as shown in Figure 5-10.

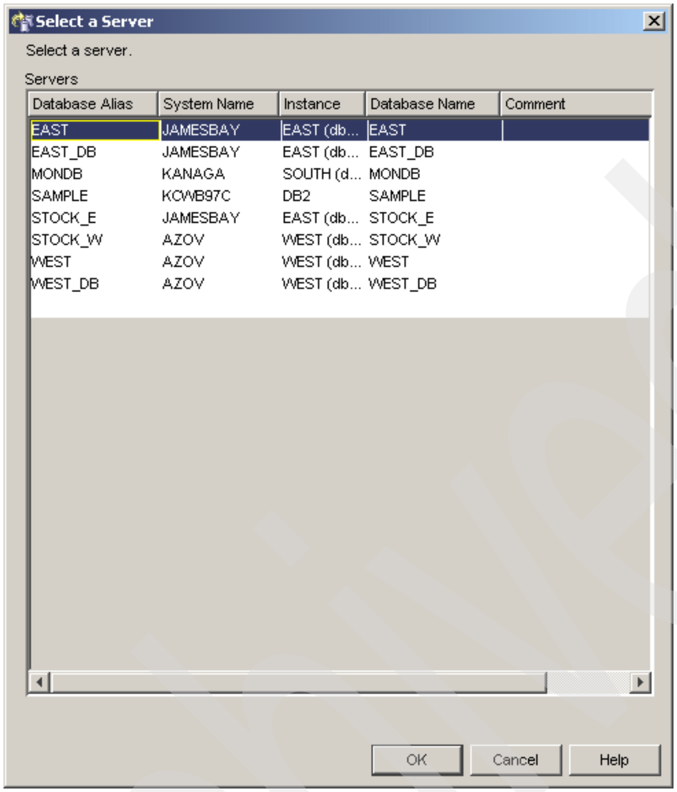


Figure 5-10 Select a Server

Select the **EAST** database alias on Jamesbay as the database server for Q Capture, and click **OK** to display Figure 5-11 for providing Q Capture details.

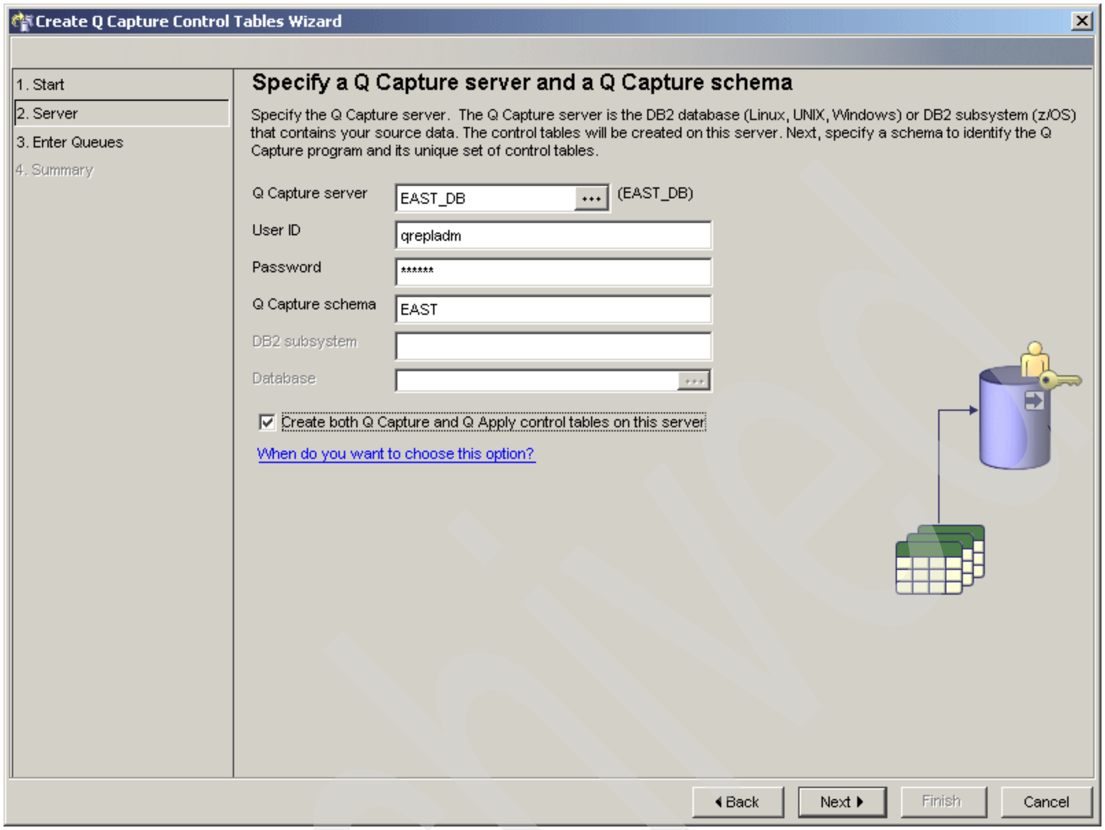


Figure 5-11 Q Capture server and schema details

Supply the Q Capture user ID and password “qrepladm/xxx” (reference A.7b in Table 5-2 on page 426) and Q Capture schema “EAST” (reference A.22 in Table 5-3 on page 428), and select the **Create both Q Capture and Q Apply control tables on this server** option.

Click **Next** to display Figure 5-12 for specifying WebSphere MQ objects.

Step 6c: Specify WebSphere MQ objects

Supply WebSphere MQ queues for this server. This includes the Queue Manager “QM_EAST” (reference A.11 in Table 5-3 on page 428), Administration queue “EAST_ADMINQ” (reference A.17 in Table 5-3 on page 428), and Restart queue “EAST_RESTARTQ” (reference A.16 in Table 5-3 on page 428).

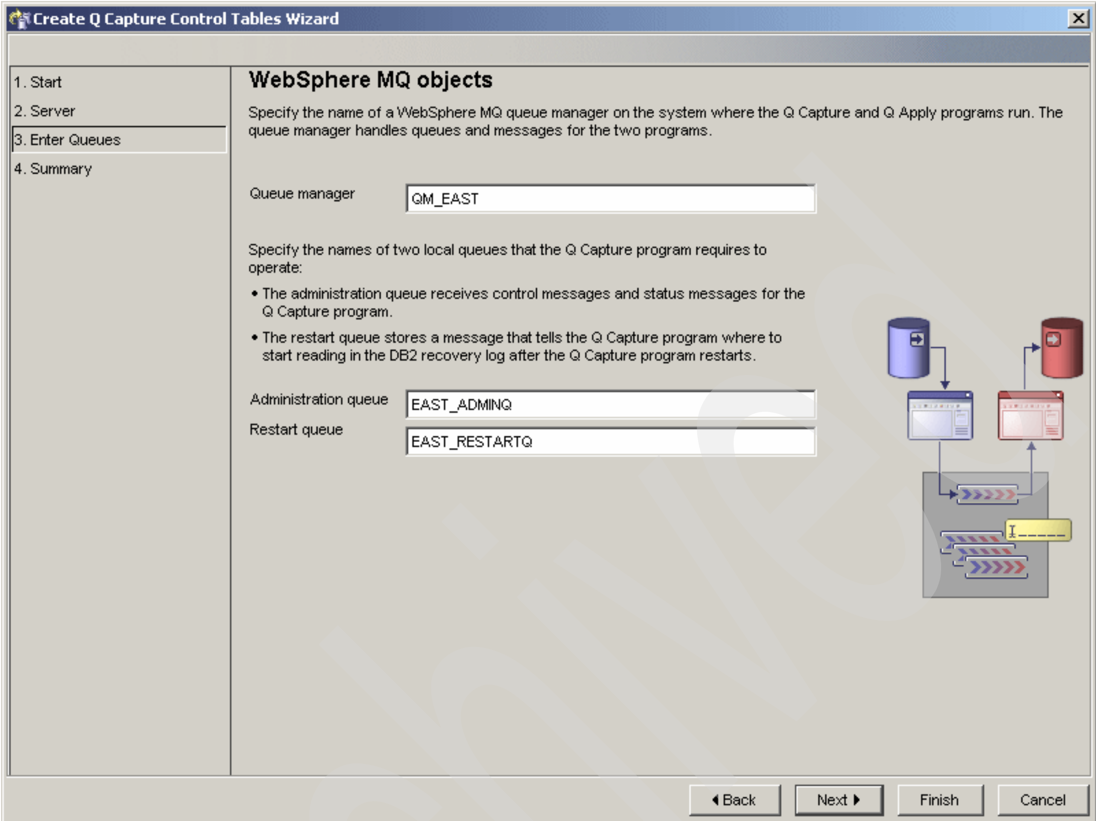


Figure 5-12 Queue information for Q Capture server

Click **Next** to continue to Figure 5-13, which summarizes the information provided in the previous screens.

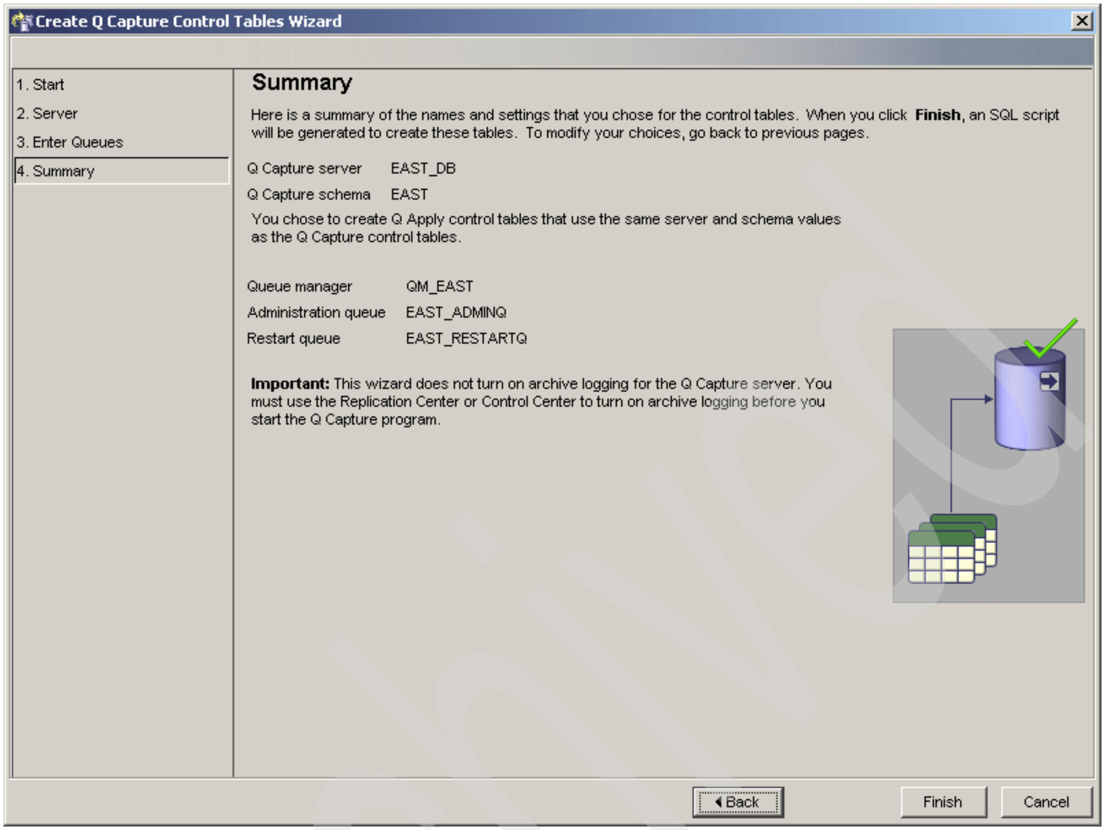


Figure 5-13 Q Capture server summary information

Verify the accuracy of the information supplied, and click **Finish** to generate the scripts that need to be executed on the Q Capture server to define the specified objects as shown in Figure 5-14.

Click the **Back** button to go back and correct any errors.

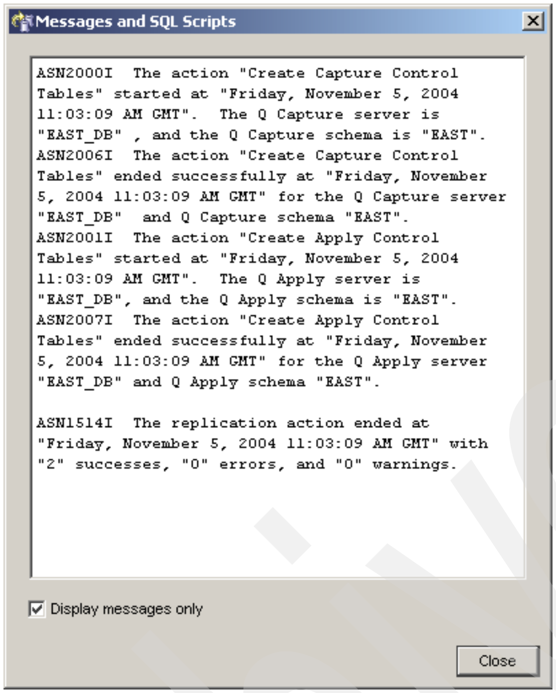


Figure 5-14 Messages and SQL Scripts screen for Q Capture server

Check that the errors are 0. Click **Close** to display Figure 5-15, and decide whether to execute the script or save it for later execution.

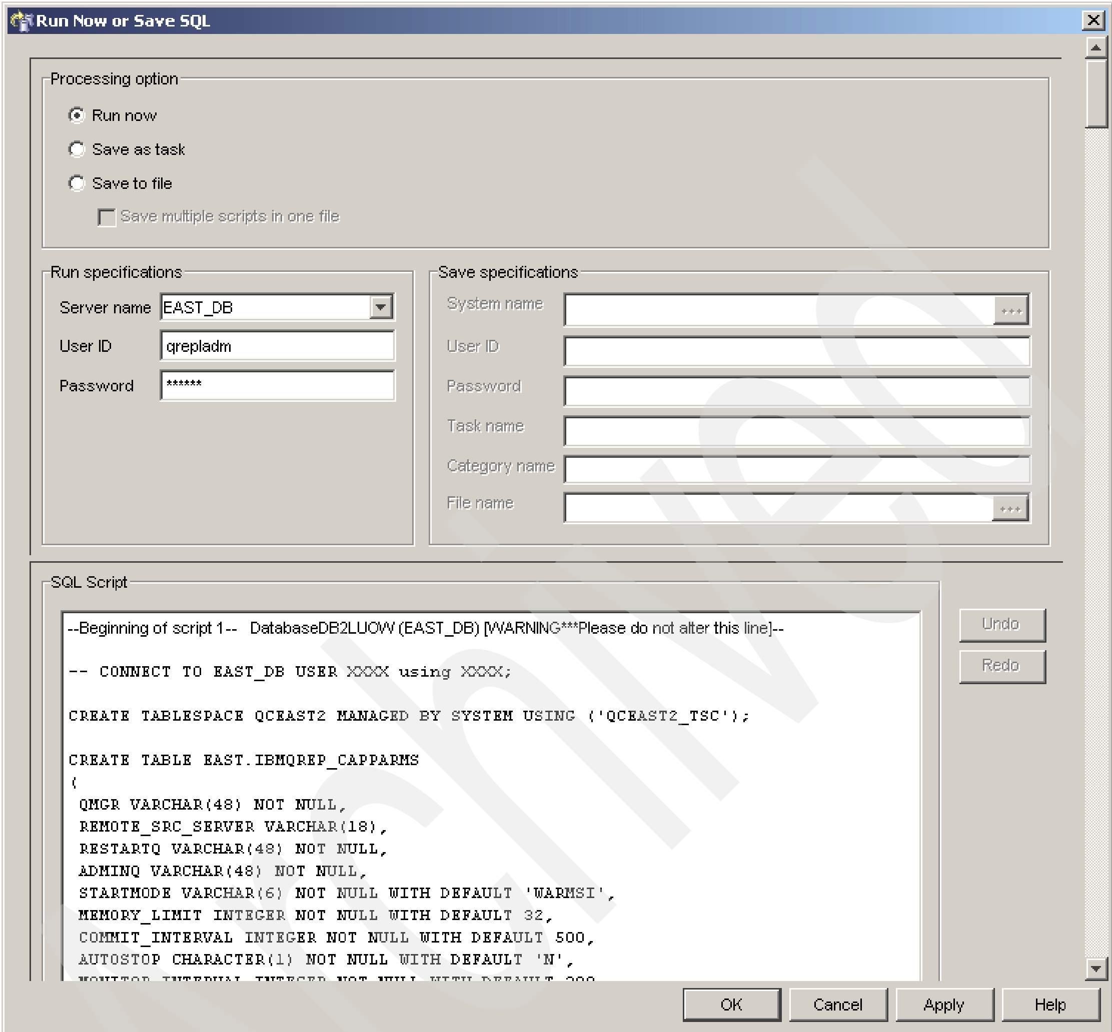


Figure 5-15 Run Now or Save SQL screen

Select the **Run now** option and click **OK** to display the message that the scripts executed successfully, as shown in Figure 5-16.

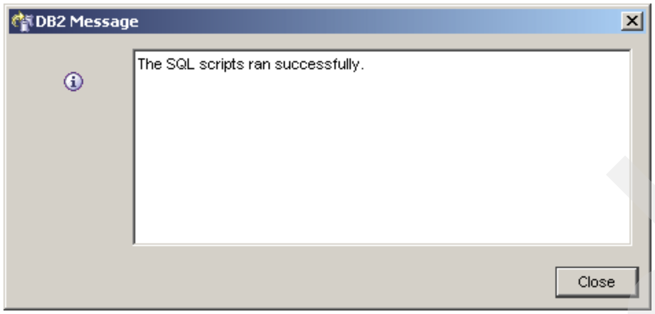


Figure 5-16 SQL scripts ran successfully message

Click **Close** to complete the definition of the Q Capture server infrastructure.

Step 6d: Repeat steps 6b through 6c for second server

The same steps for the Q Capture server on the EAST node (Jamesbay) need to be repeated for the WEST node (Azov) using the appropriate reference information in Table 5-3 on page 428, as shown in Figure 5-17 through Figure 5-26 on page 471.

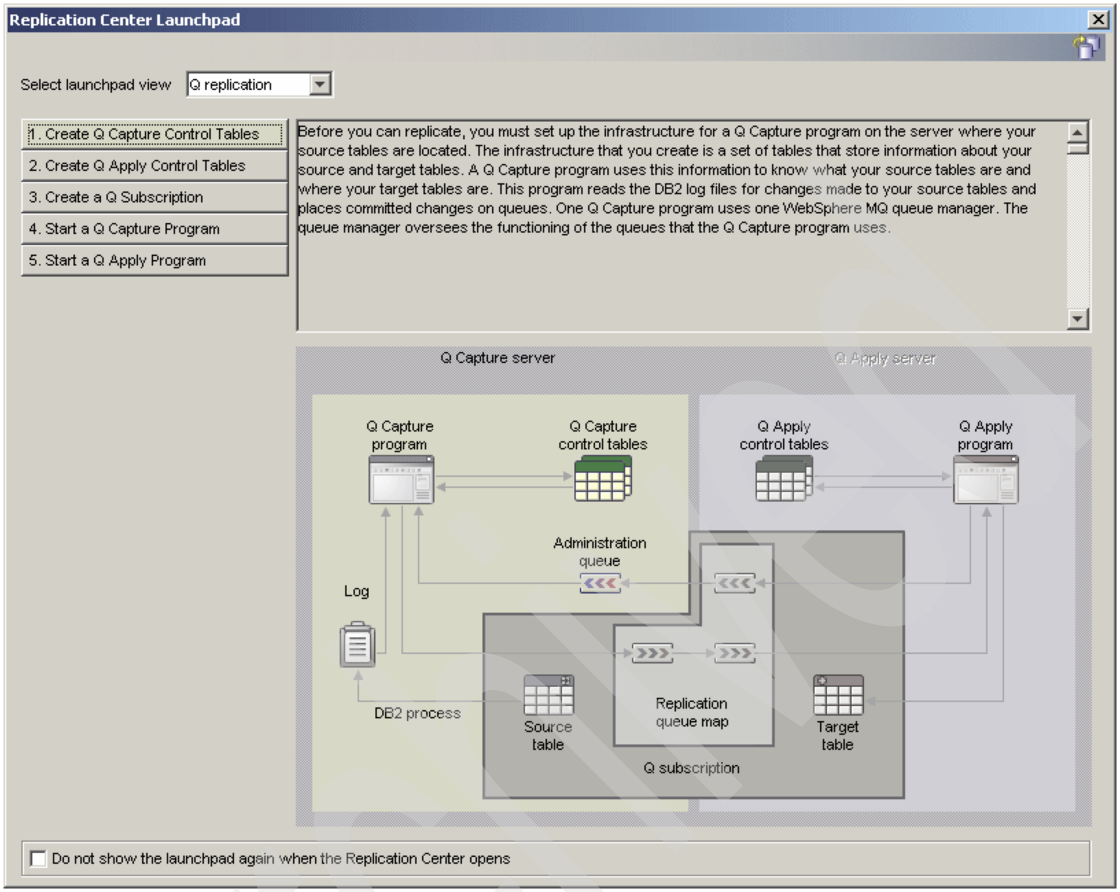


Figure 5-17 Five steps in setting up Q replication infrastructure

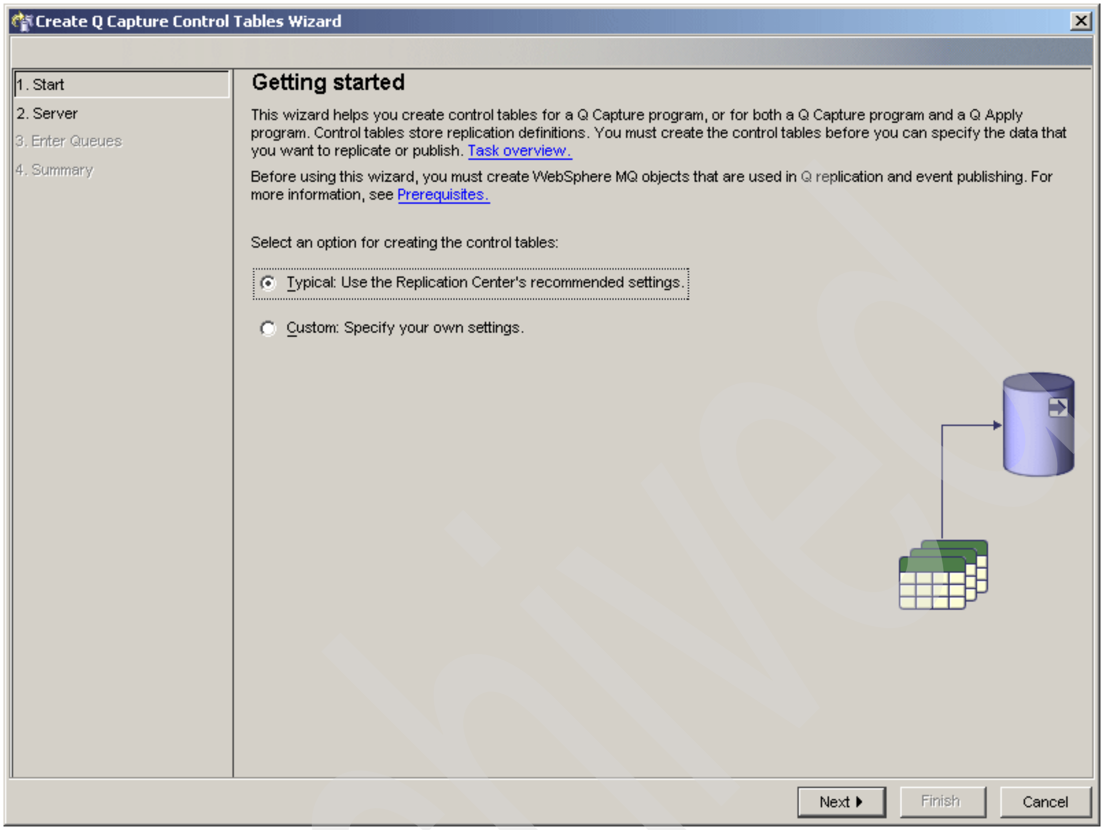


Figure 5-18 Getting started - Create Q Capture Control Tables Wizard

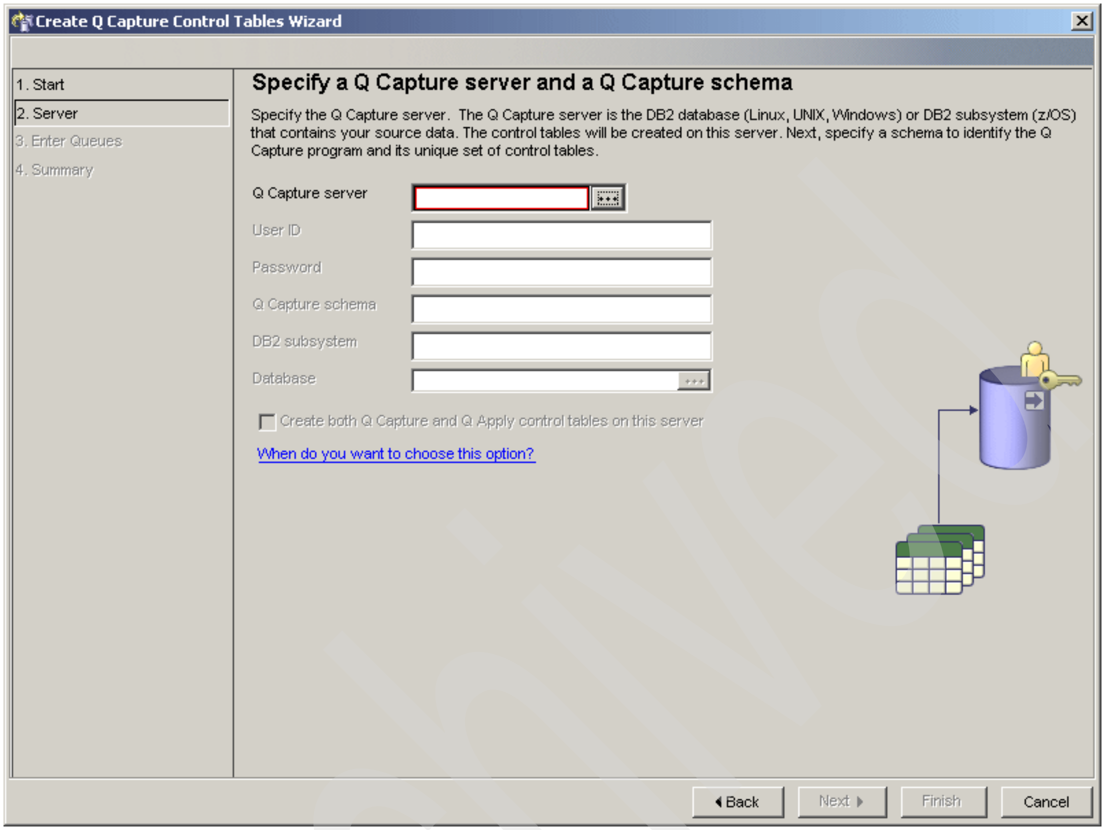


Figure 5-19 Server - Create Q Capture Control Tables Wizard

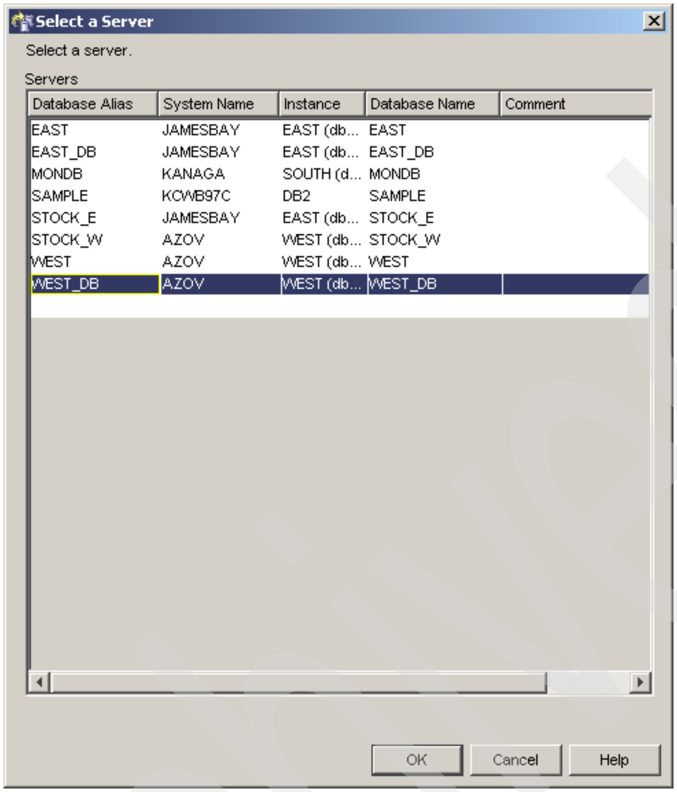


Figure 5-20 Select a Server

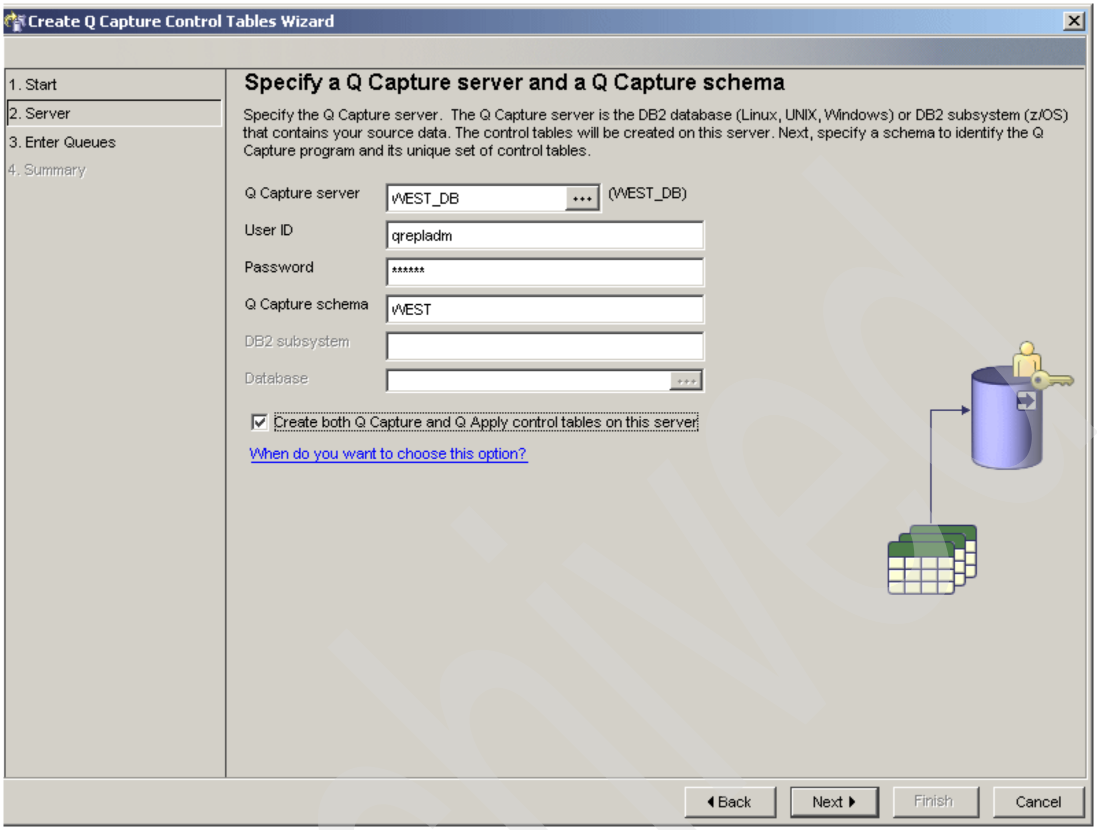


Figure 5-21 Q Capture server and schema details

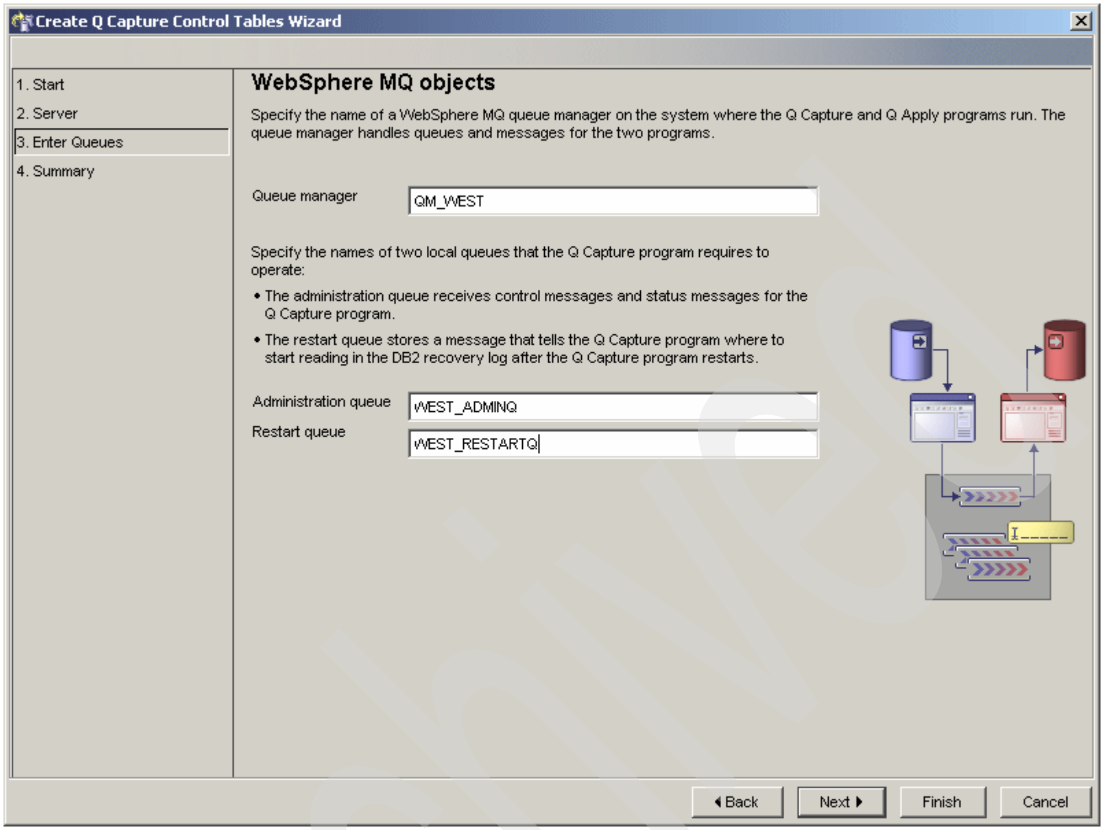


Figure 5-22 Queue information for Q Capture server

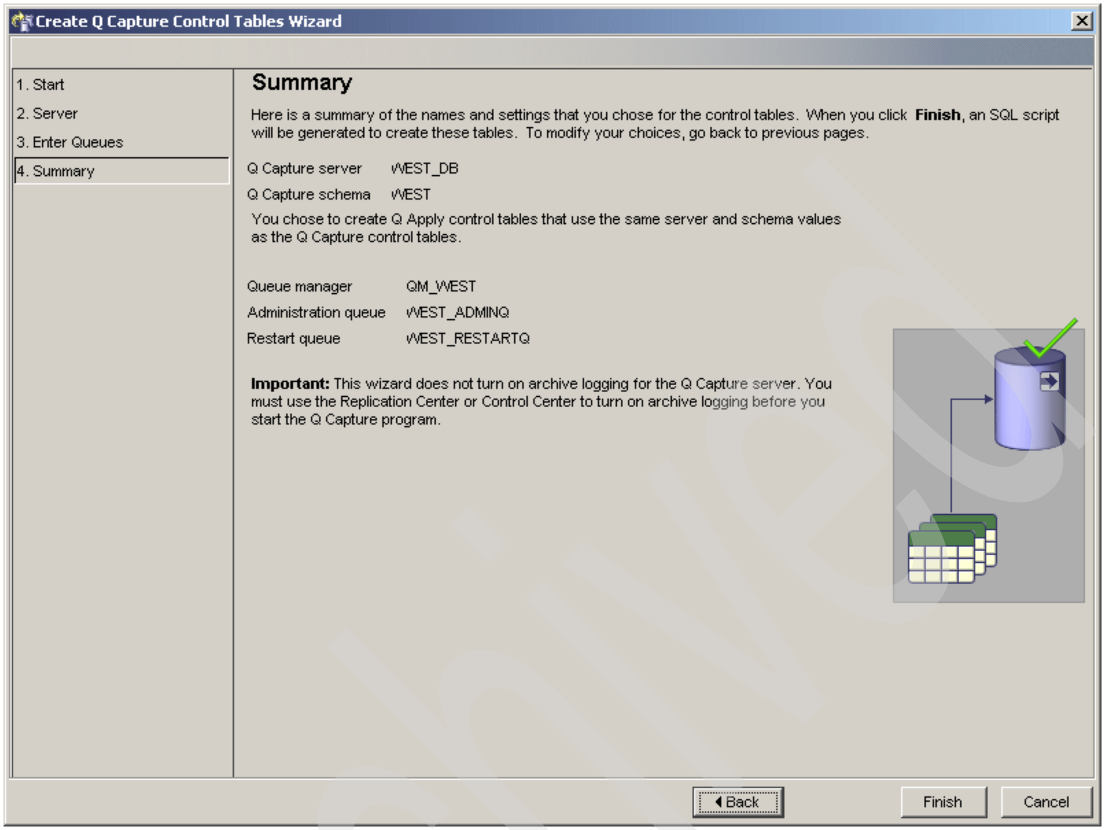


Figure 5-23 Q Capture server summary information

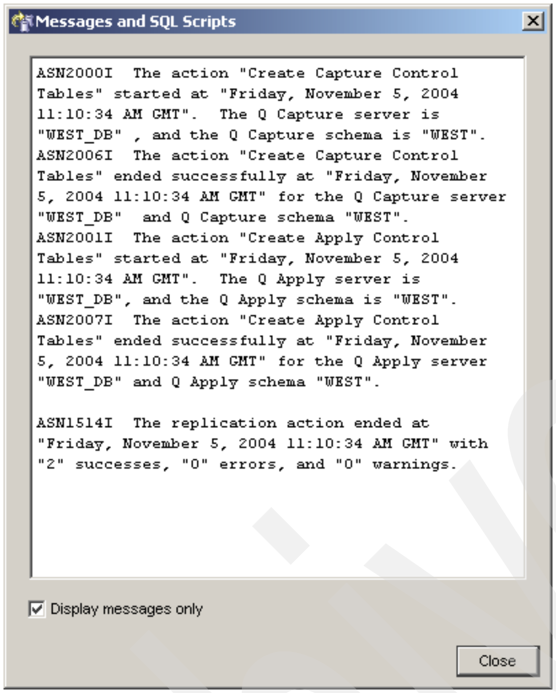


Figure 5-24 Messages and SQL Scripts screen for Q Capture server

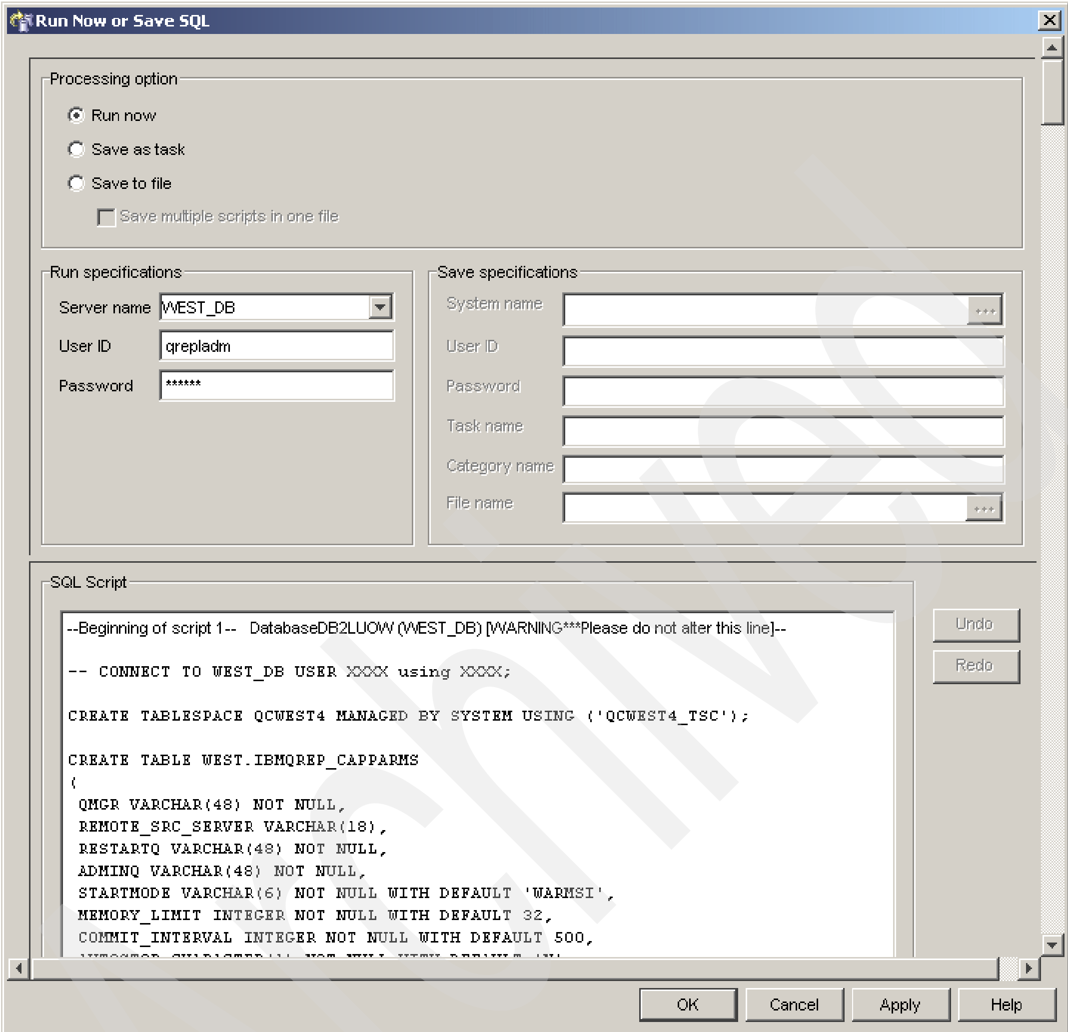


Figure 5-25 Run Now or Save SQL screen

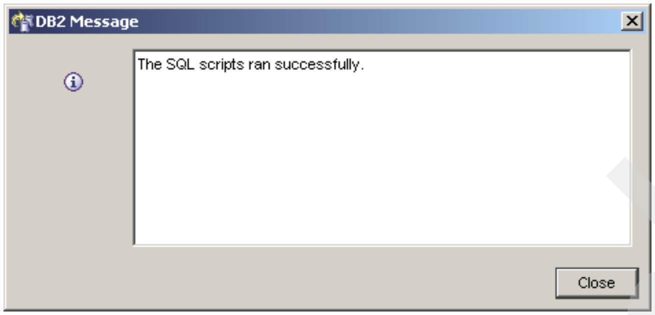


Figure 5-26 SQL scripts ran successfully message

Step 6e: Create Q Subscriptions and replication queue maps

After the Q Capture and Q Apply control tables have been created, the Q subscriptions and replication queue maps need to be created. Click the **Create a Q Subscription** option in Figure 5-27 to display Figure 5-28 to begin the process.

> **Note:** Since we created the Q Apply control tables earlier by selecting the **Create both Q Capture and Q Apply control tables on this server** in Figure 5-21 on page 466, we skip the Create Q Apply Control Tables option in Figure 5-28.

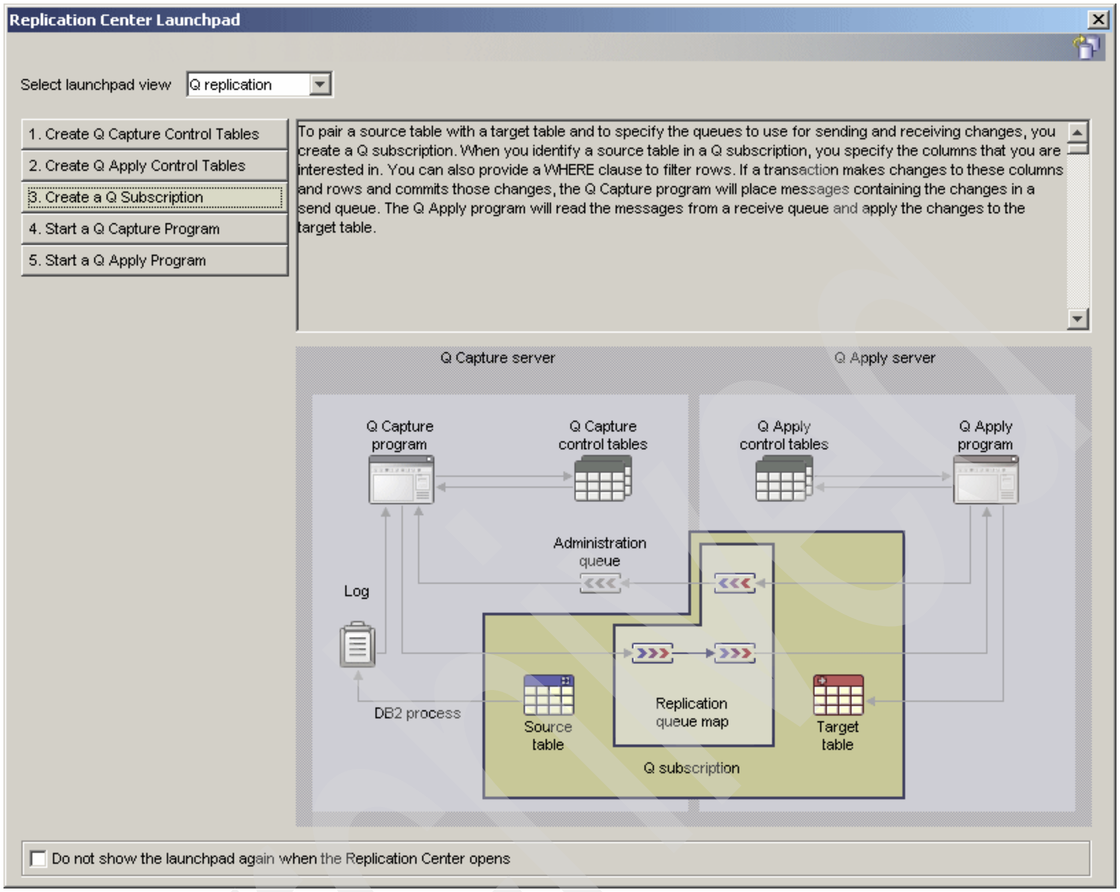


Figure 5-27 Five steps in setting up Q replication infrastructure

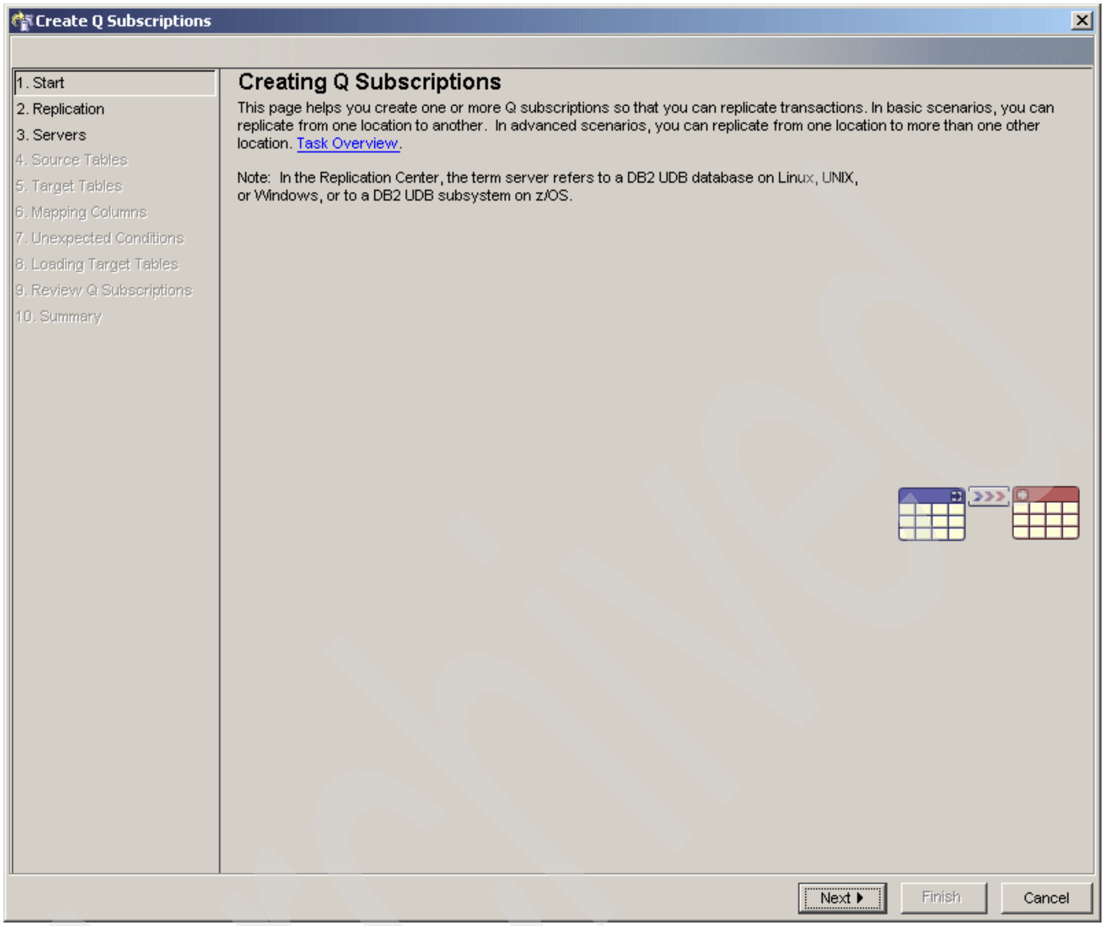


Figure 5-28 Create Q Subscriptions

Click **Next** to continue to Figure 5-29 to choose the type of replication topology desired.

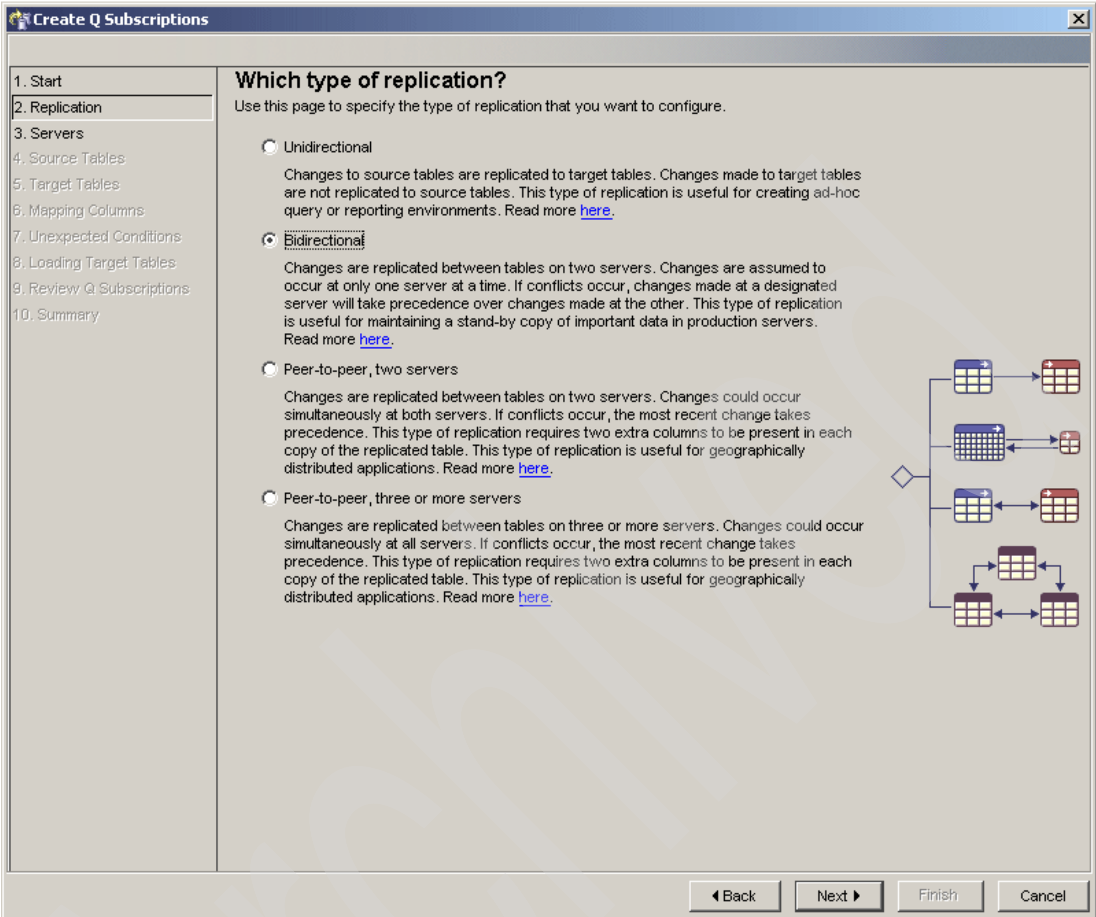


Figure 5-29 Which type of replication

Select the **Bidirectional** setting and click **Next** to provide details about the servers involved in the replication topology, as shown in Figure 5-30.

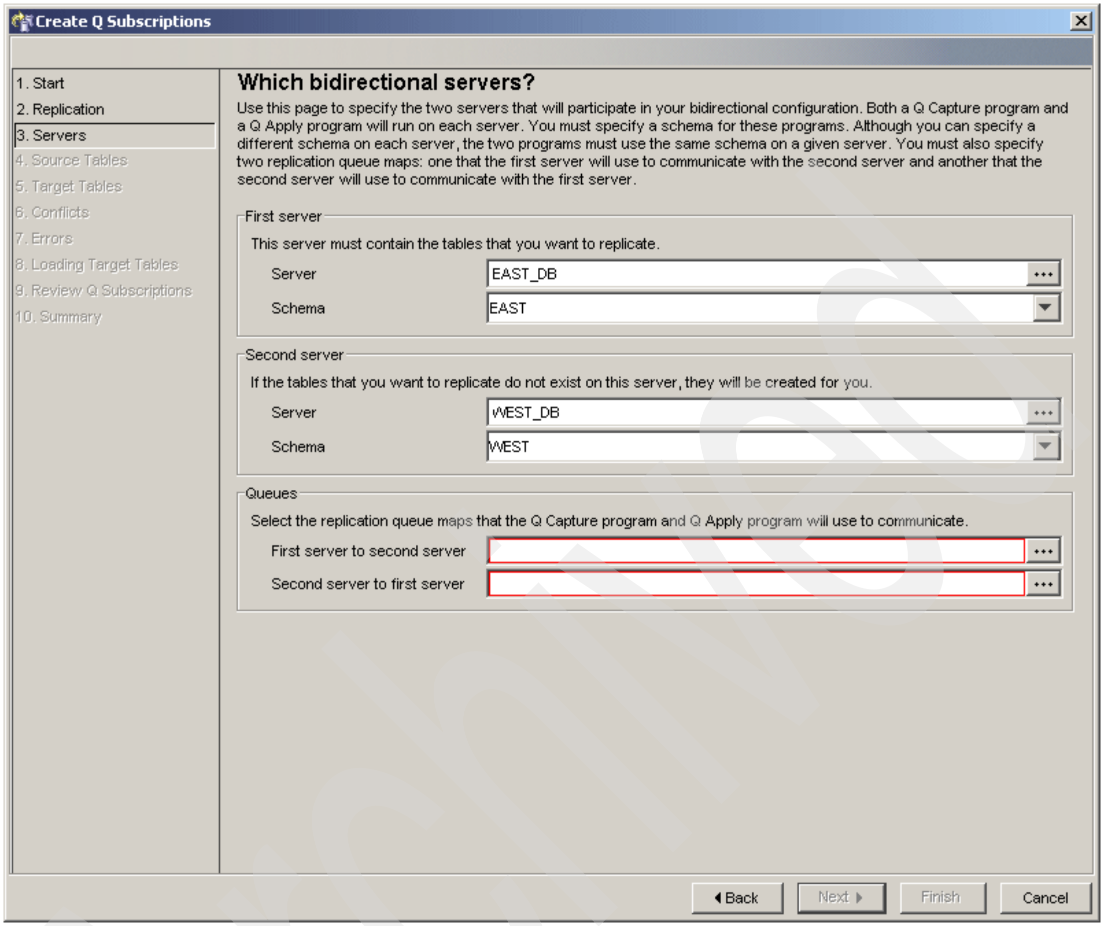


Figure 5-30 Which bidirectional servers

Click the **...** button for the "First server to second server" to get a list of the available replication queue maps, as shown in Figure 5-31.

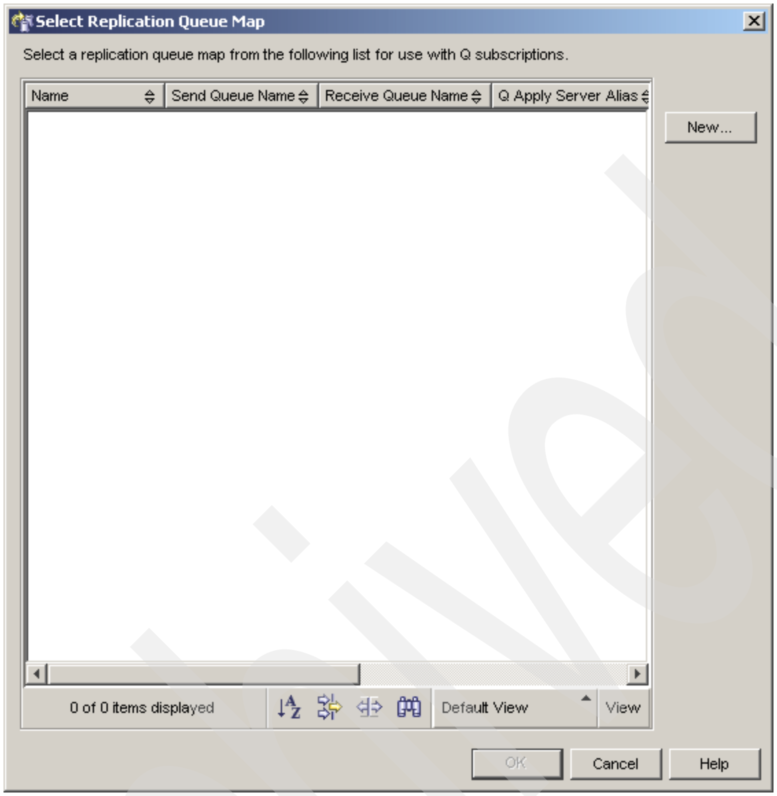


Figure 5-31 Select Replication Queue Map

Since no replication queue maps have been defined as yet, Figure 5-31 has no list of existing replication queue maps. A new replication queue map needs to be created by clicking the **New...** button. This displays Figure 5-32.

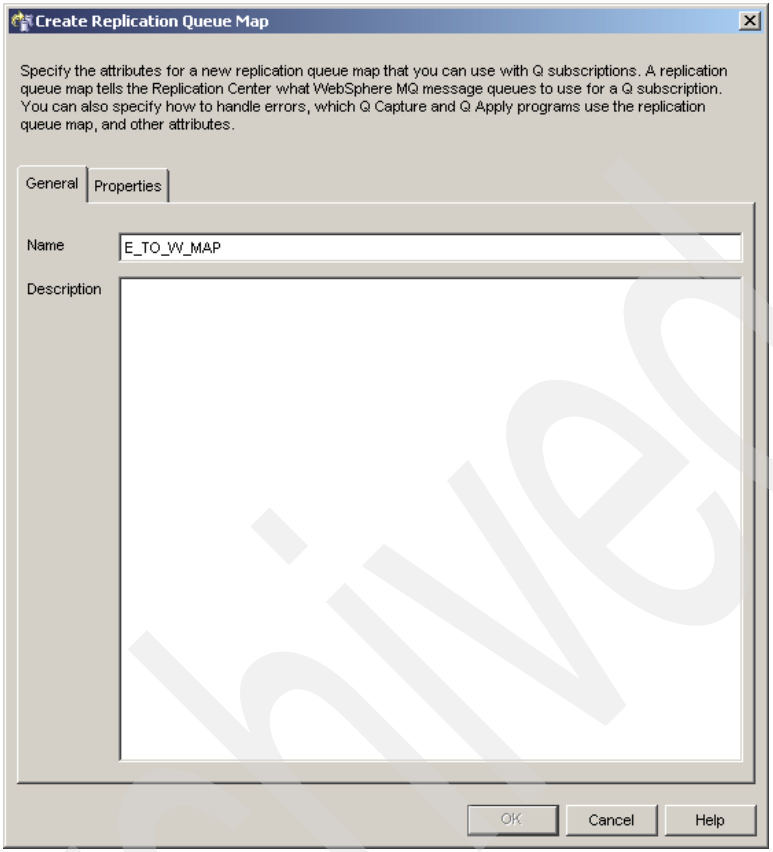


Figure 5-32 Create Replication Queue Map

Supply the name E_TO_W_MAP (reference A.27 in Table 5-3 on page 428), and then choose the **Properties** tab to display Figure 5-33.

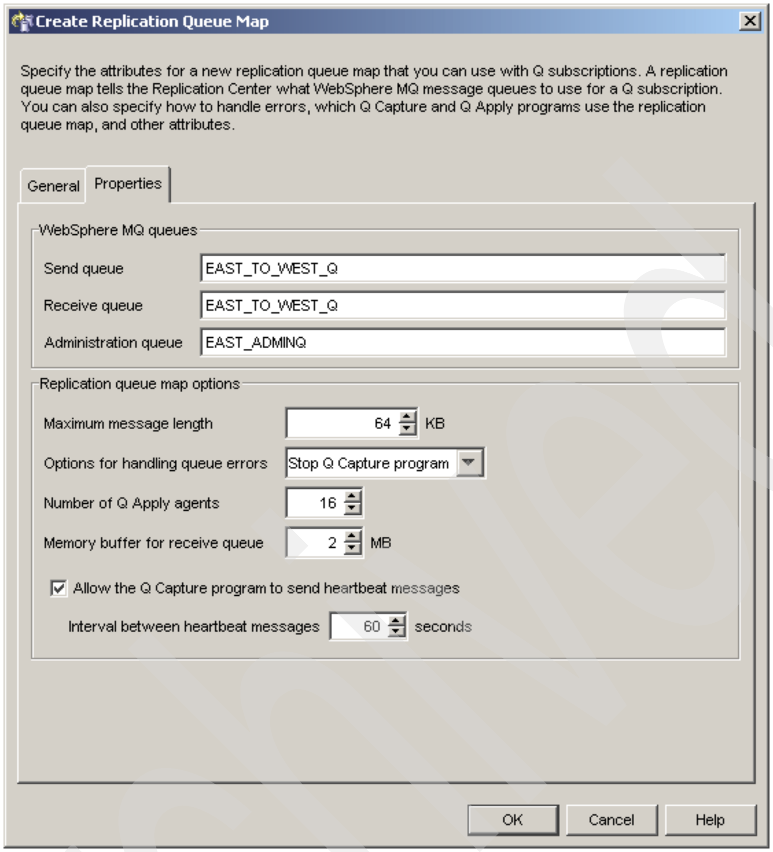


Figure 5-33 Create Replication Queue Map - Properties

Supply values for the Send queue (“EAST_TO_WEST_Q” reference A.18 in Table 5-3 on page 428), Receive queue (“EAST_TO_WEST_Q” reference B.19 in Table 5-3 on page 428), Administration queue (“EAST_ADMINQ” reference A.17 in Table 5-3 on page 428), and let other values default.

Click **OK** to complete the definition, and generate the scripts for creating the replication queue map objects on the servers, as shown in Figure 5-36.

Attention: In fixpak 9, which is due out in April 2005, the screens shown in Figure 5-32 and Figure 5-33 have been enhanced to appear as shown in Figure 5-34 and Figure 5-35, respectively. Please ignore the values shown therein, since the screens have been shown to only highlight the difference in the new screen layout.

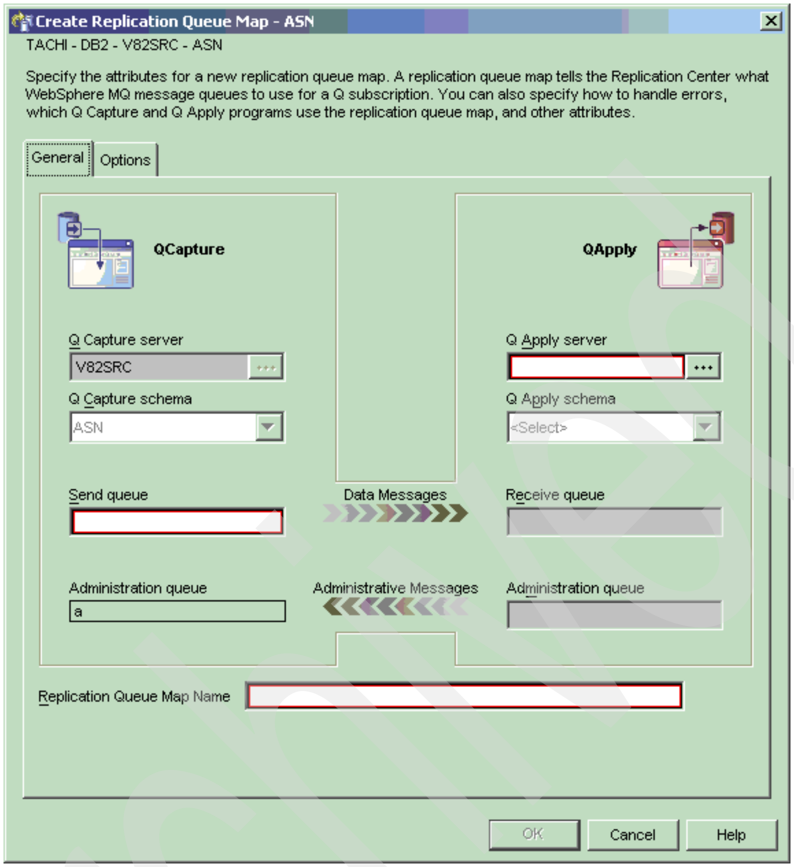


Figure 5-34 Fixpak 9 Create Replication Queue Map screen

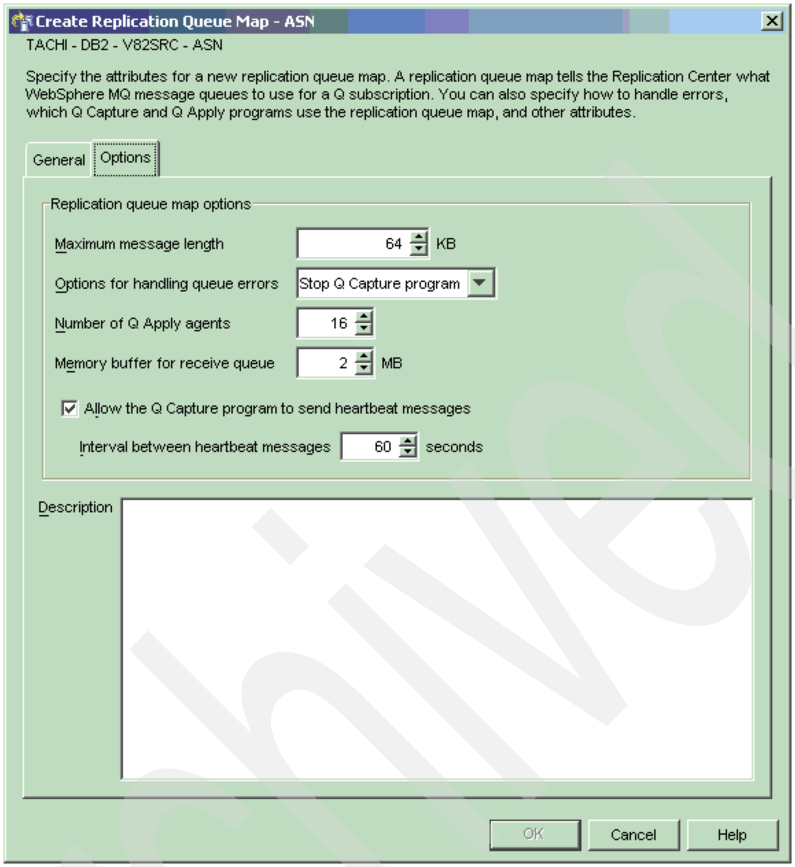


Figure 5-35 Fixpak 9 Create Replication Queue Map - Options screen

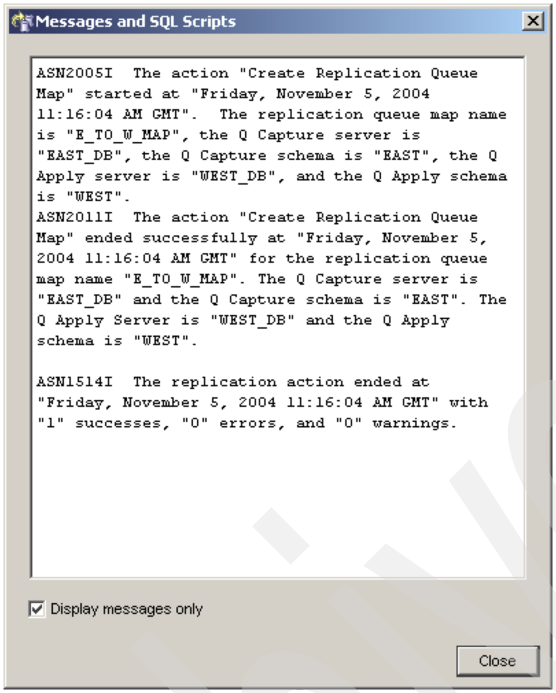


Figure 5-36 Messages and SQL Scripts

Click **Close** to bring up Figure 5-37 for choosing whether to execute the scripts right away or save the scripts for later execution.

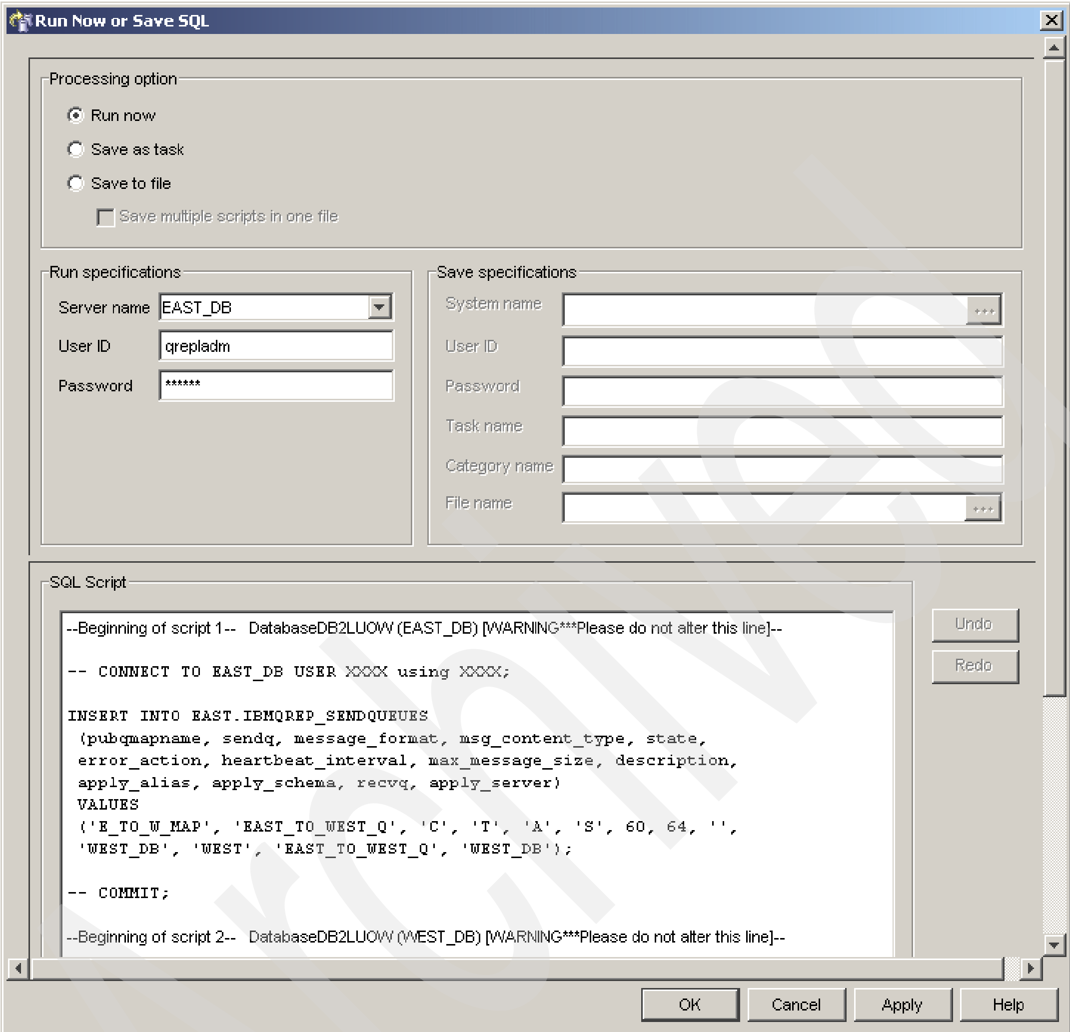


Figure 5-37 Run Now or Save SQL

Select the **Run now** option and click **OK** to execute the script immediately. Figure 5-38 is displayed when the scripts run successfully.

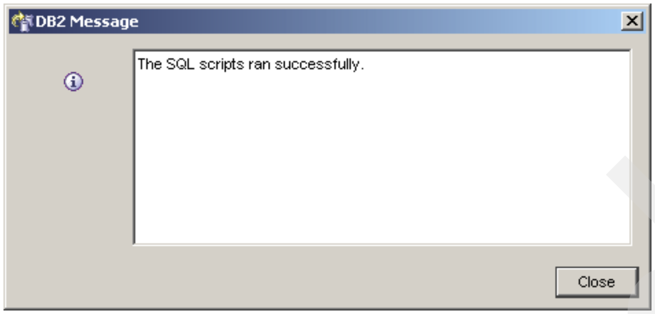


Figure 5-38   SQL scripts ran successfully message

Click **Close** to proceed to Figure 5-39, which displays the just created Replication Queue Map.

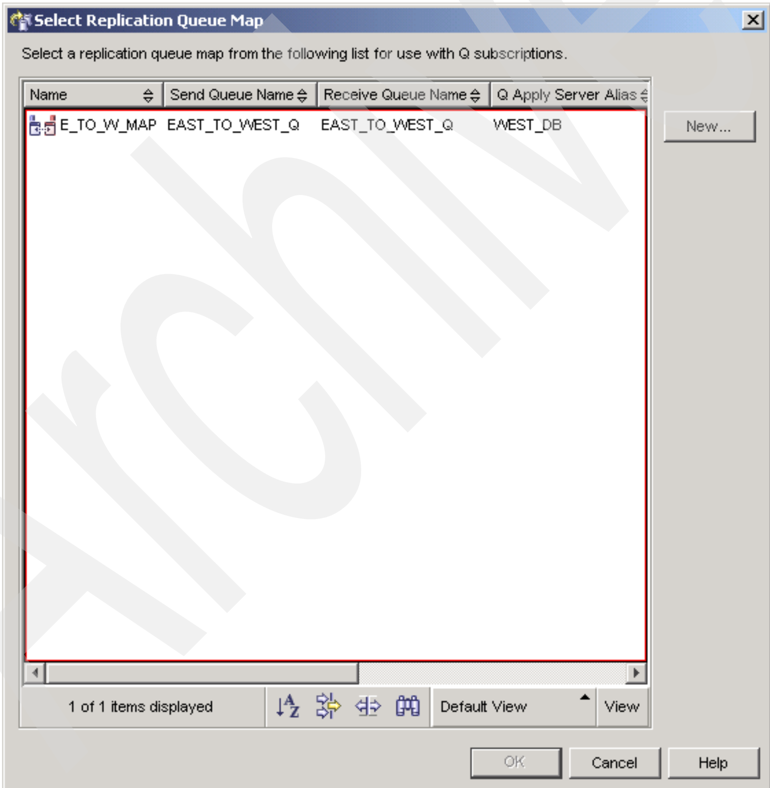


Figure 5-39   Select Replication Queue Map

Highlight the created replication queue map and click **OK** to proceed to Figure 5-40.

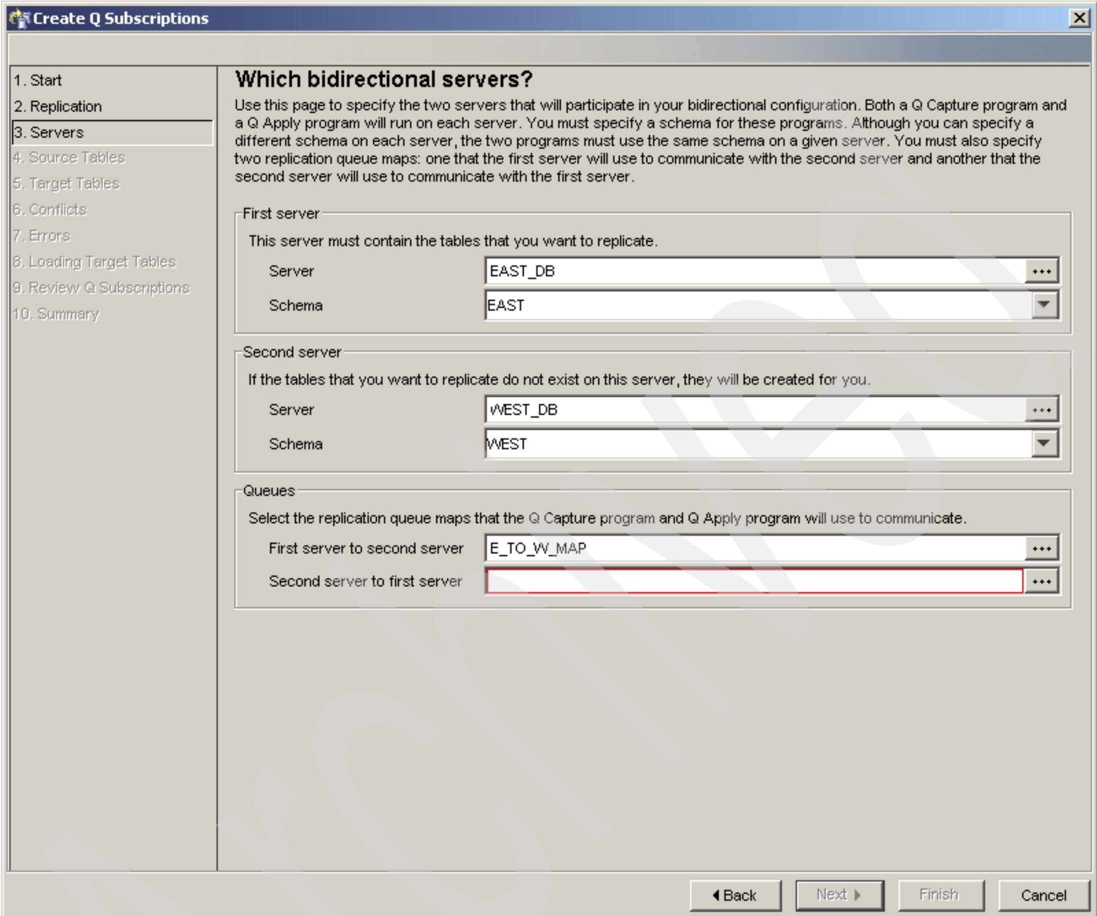


Figure 5-40 Which bidirectional servers

The next step is to select the replication queue map for communication between the “Second server to first server.”

Repeat the steps performed earlier for the replication queue map definition from the “First server to second server” (as described in Figure 5-30 on page 475 through Figure 5-38 on page 483) using the appropriate template values corresponding to node WEST on Azov. This is described in Figure 5-40 through Figure 5-47 on page 490.

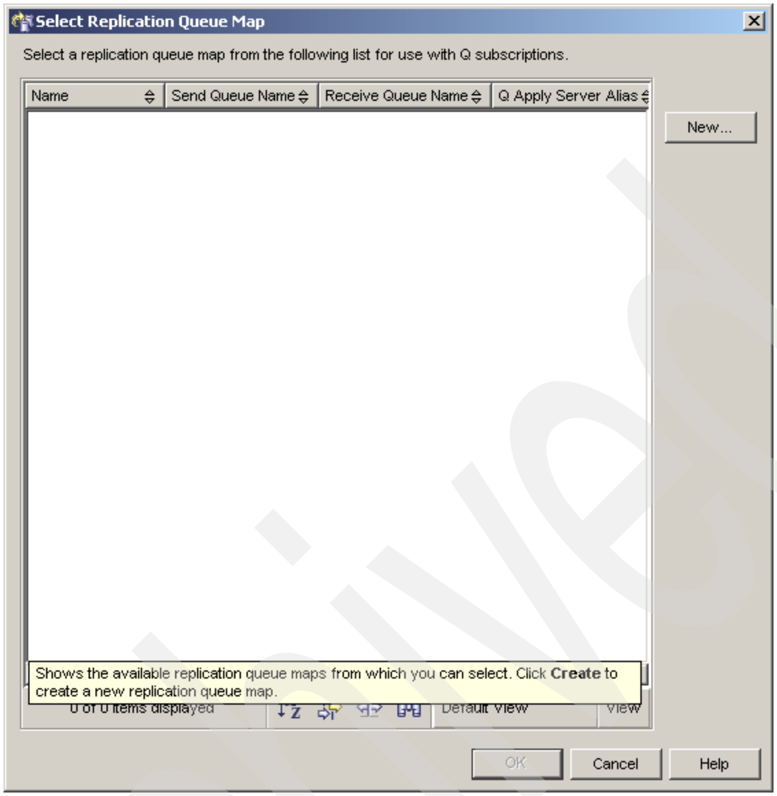


Figure 5-41 Select Replication Queue Map

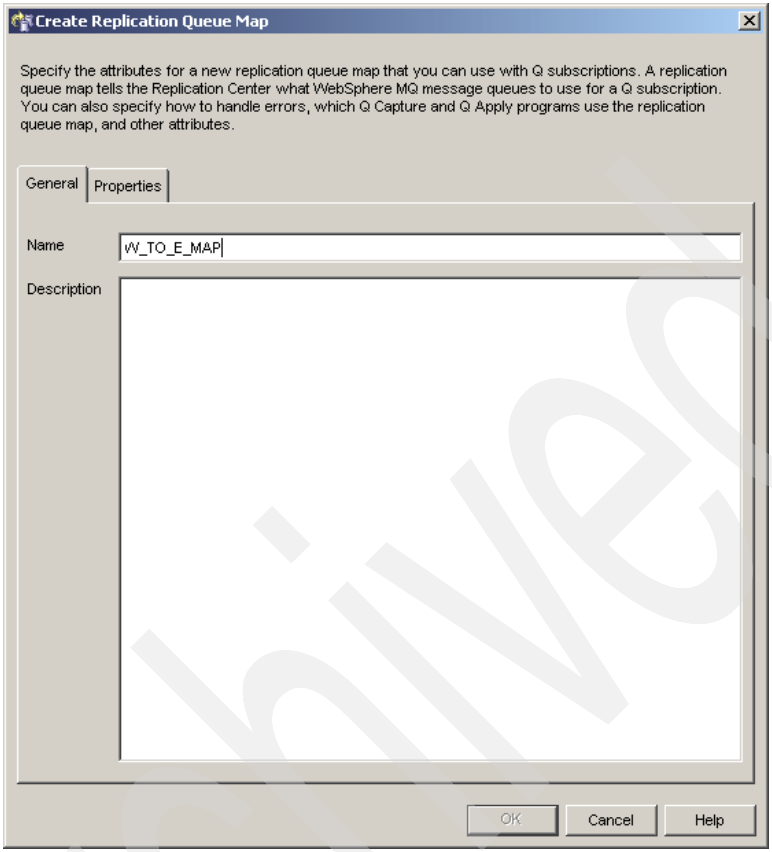


Figure 5-42 Create Replication Queue Map

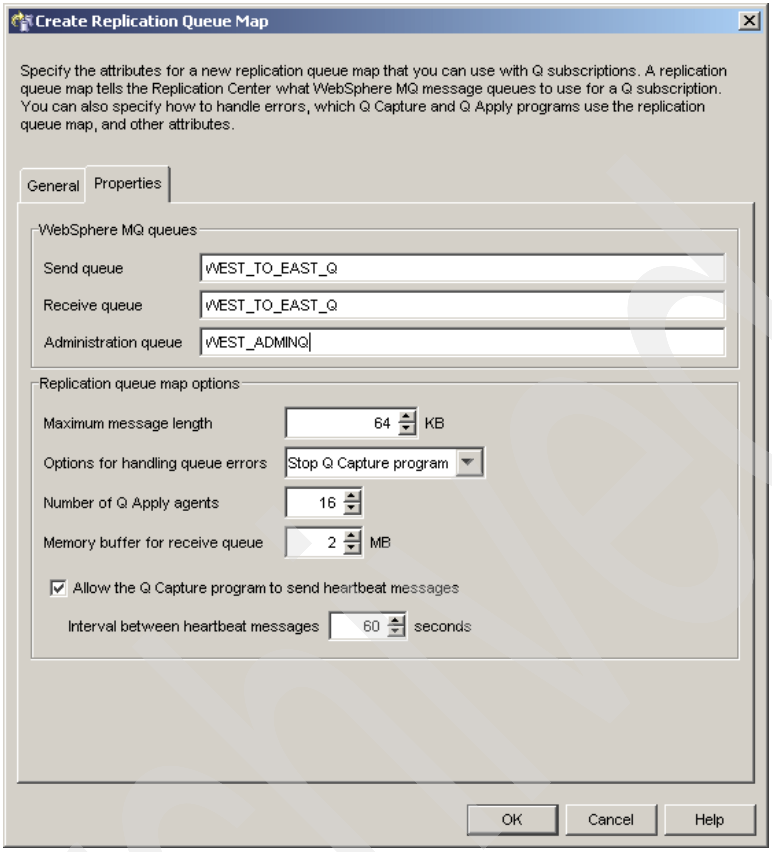


Figure 5-43 Create Replication Queue Map - Properties

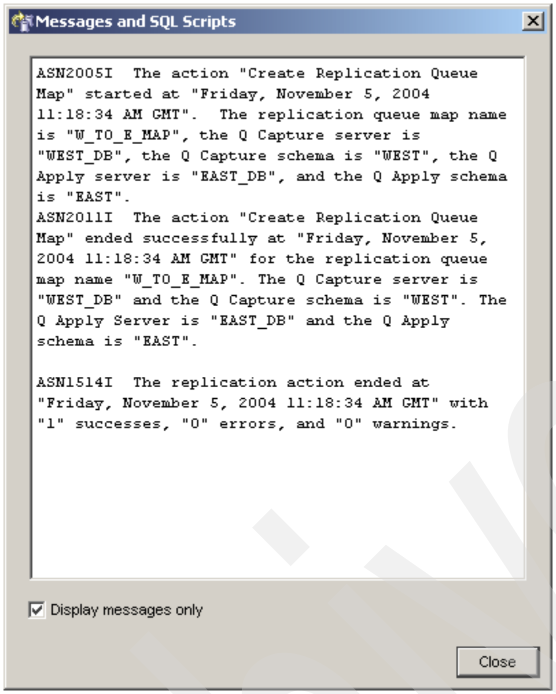


Figure 5-44 Messages and SQL Scripts

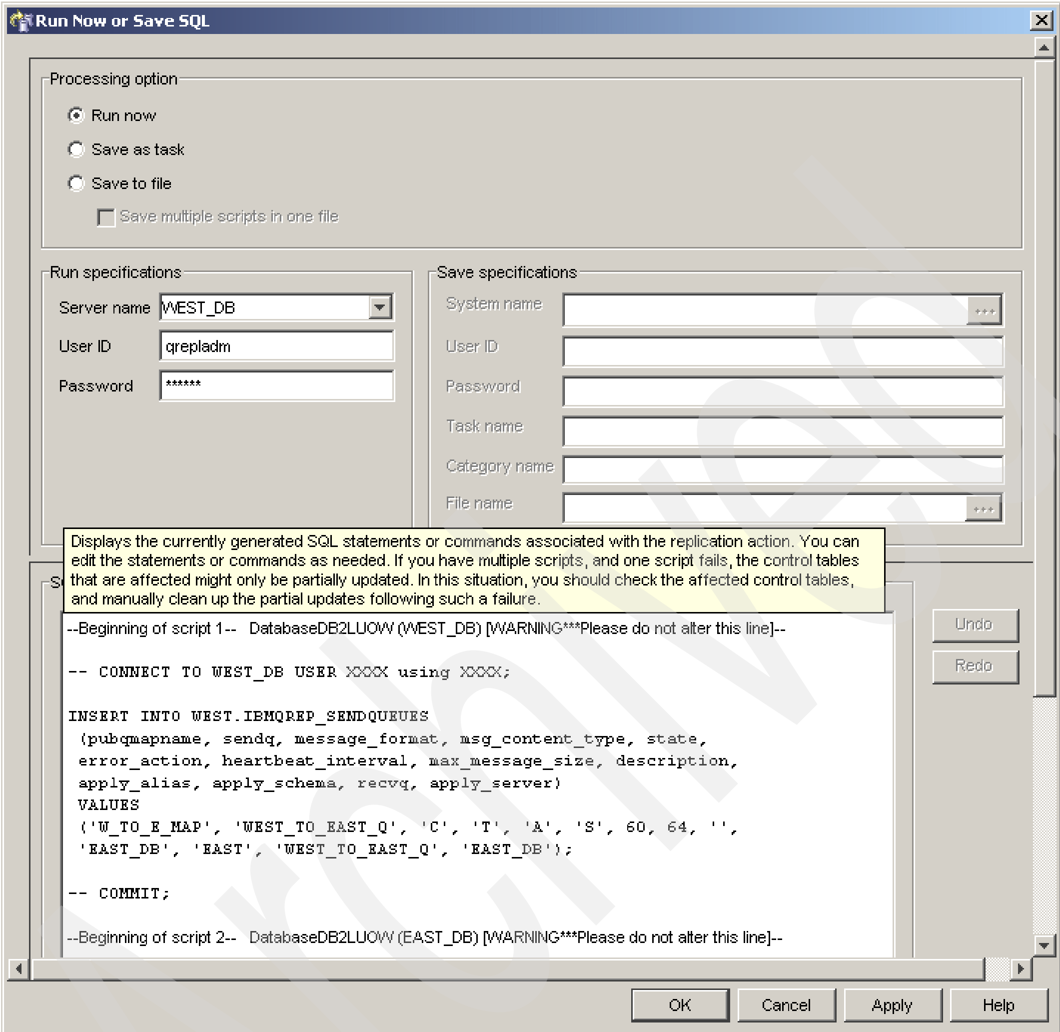


Figure 5-45 Run Now or Save SQL

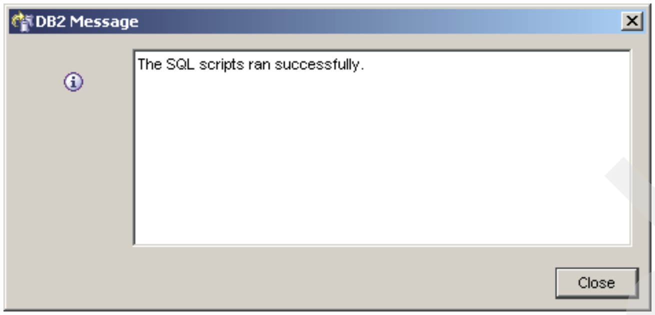


Figure 5-46 SQL scripts ran successfully message

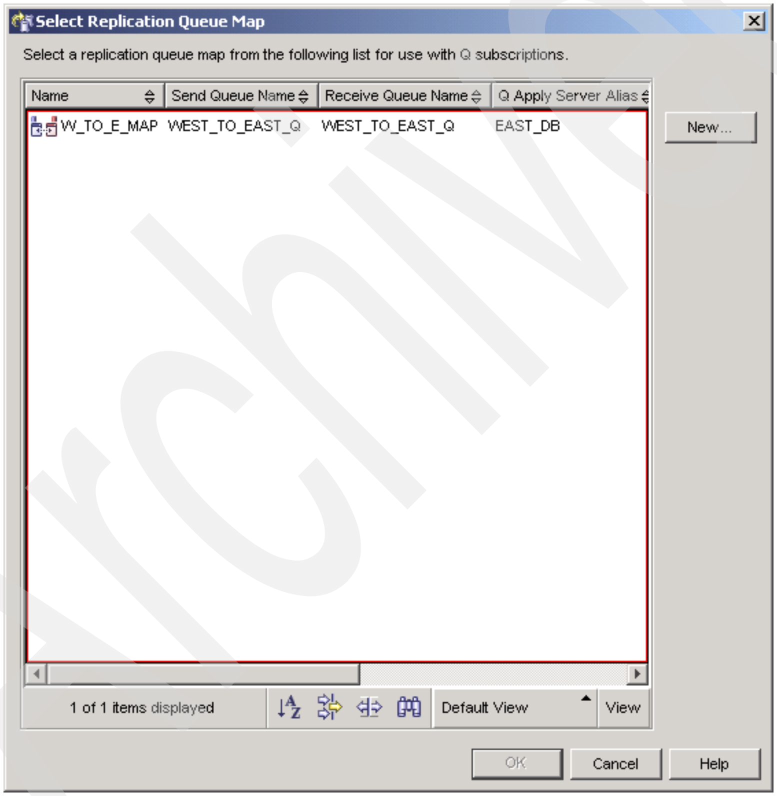


Figure 5-47 Select Replication Queue Map

At the completion of creating the replication queue maps for communication between the servers (see Figure 5-48), the individual subscriptions need to be defined.

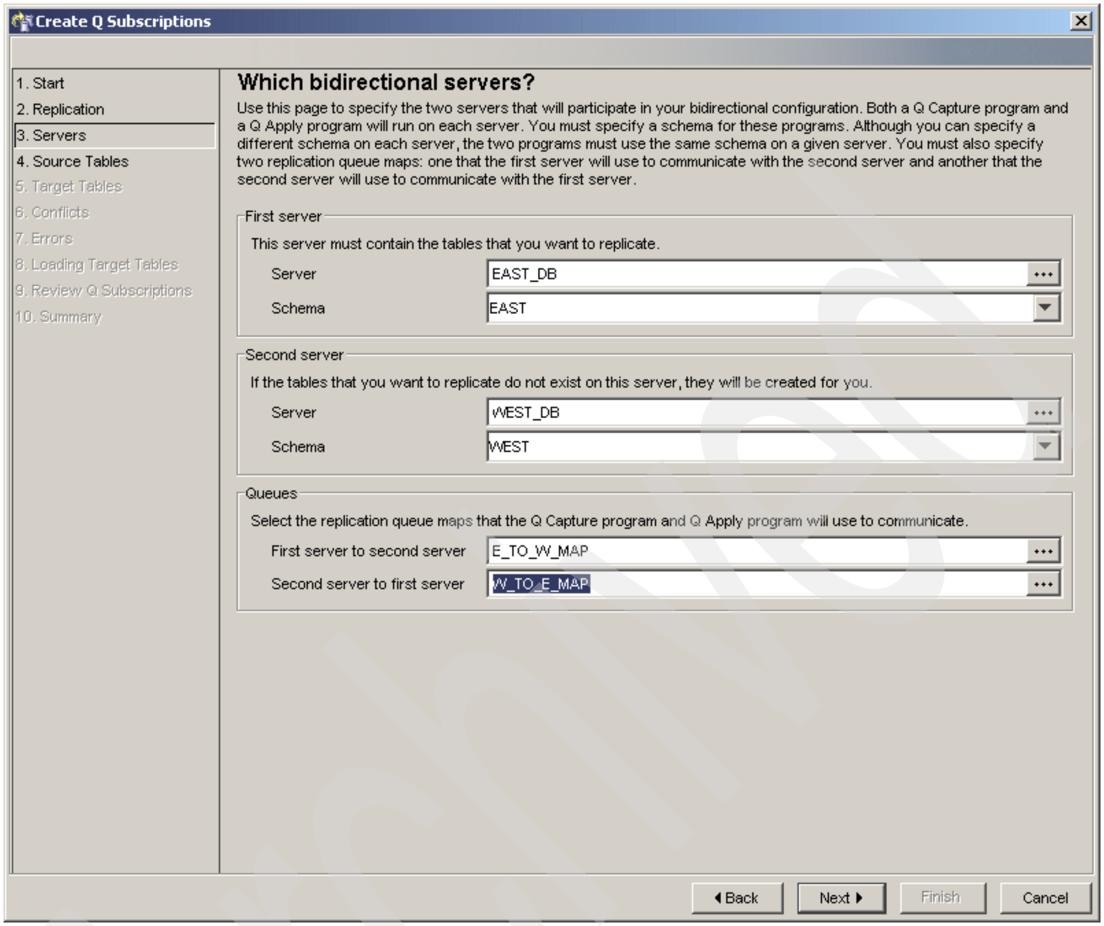


Figure 5-48 Completion of the replication queue maps selection

Click **Next** in Figure 5-48 to begin the process of defining the subscriptions as shown in Figure 5-49.

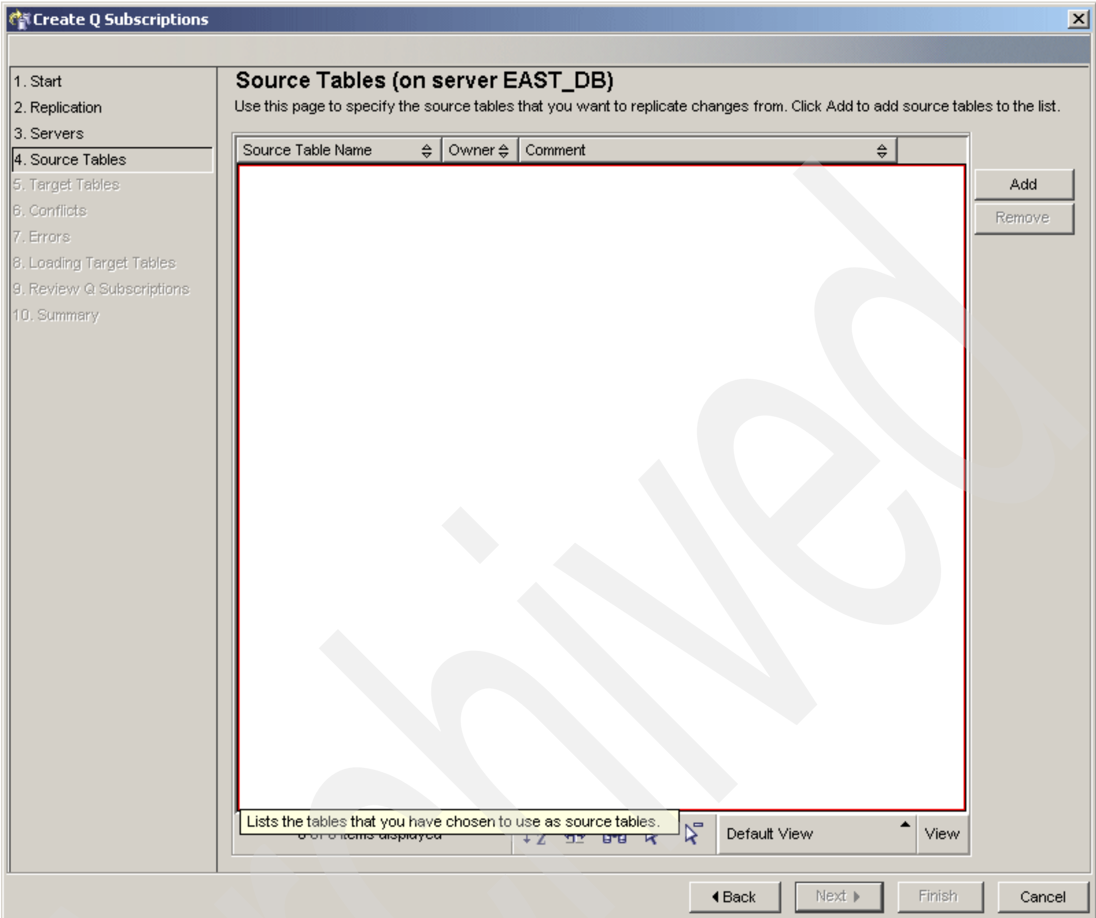


Figure 5-49   Create Q Subscriptions - Source Tables (on server EAST_DB)

Since there are no tables to select from, click the **Add** button in Figure 5-49 to add to the list of tables available for subscriptions, as shown in Figure 5-50.

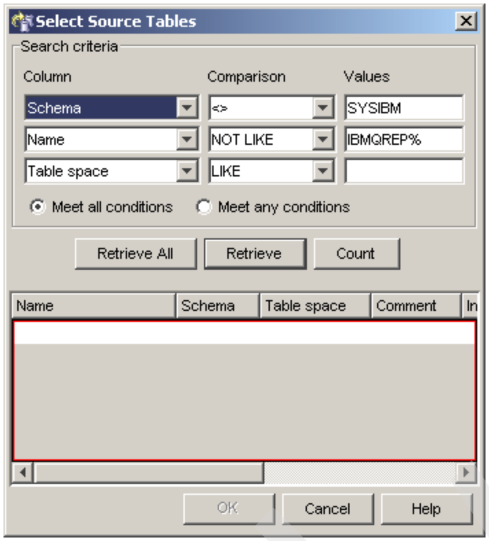


Figure 5-50   Select Source Tables

Exclude all tables that have a schema name of SYSIBM, and tables with names beginning with IBMQREP%. Click **Retrieve** to list all the tables that qualify, as shown in Figure 5-51. Clicking **Retrieve All** ignores the criteria.

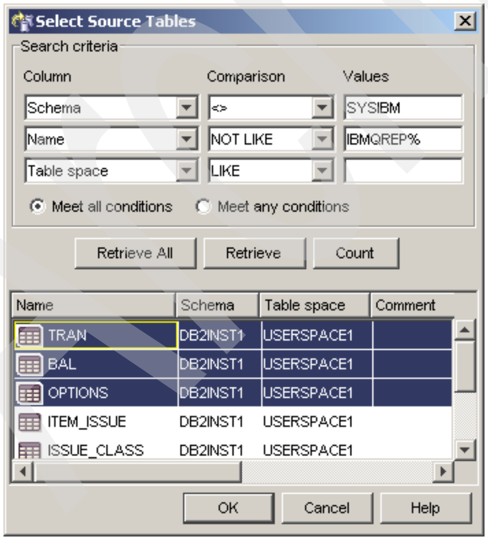


Figure 5-51   Select Source Tables

Select the **TRAN**, **BAL** and **OPTIONS** tables and click **OK** to display Figure 5-52.

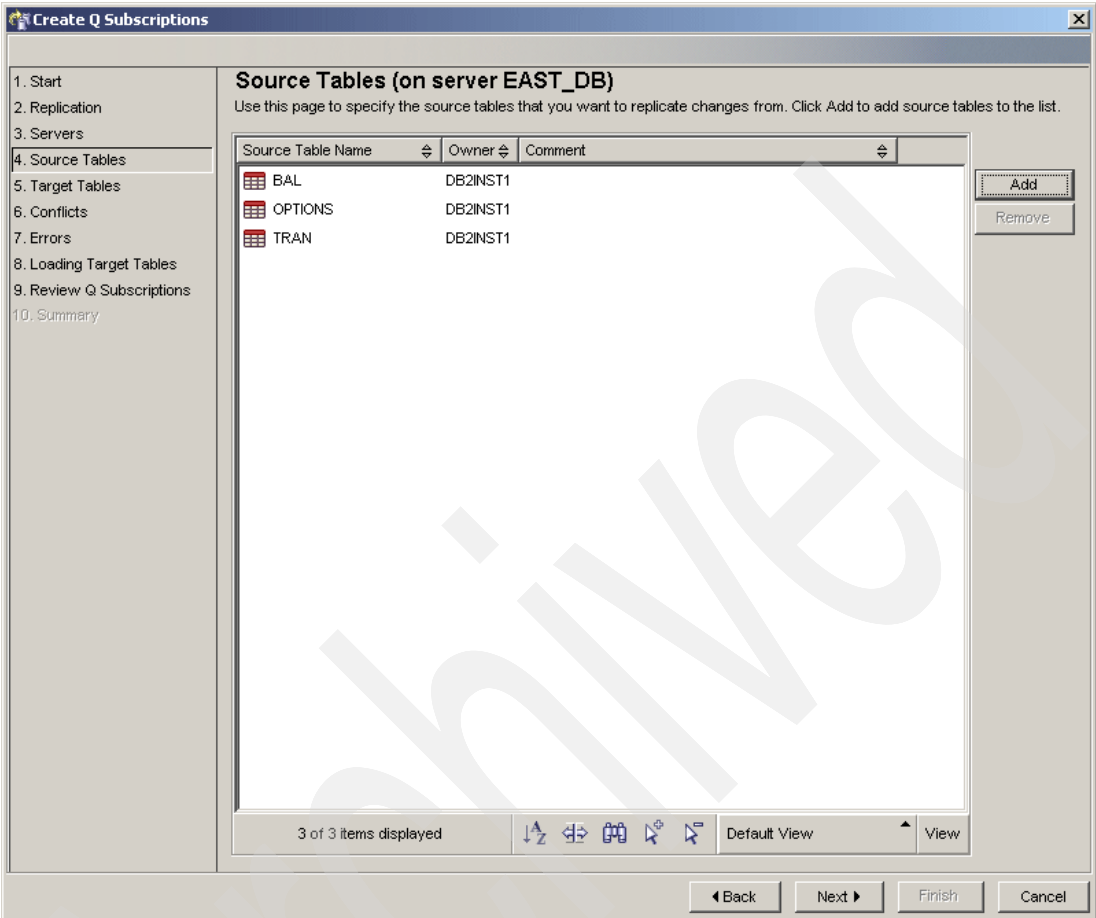


Figure 5-52 Create Q Subscriptions (on server EAST_DB)

Click **Next** to set the profiles for the target tables, as shown in Figure 5-53.

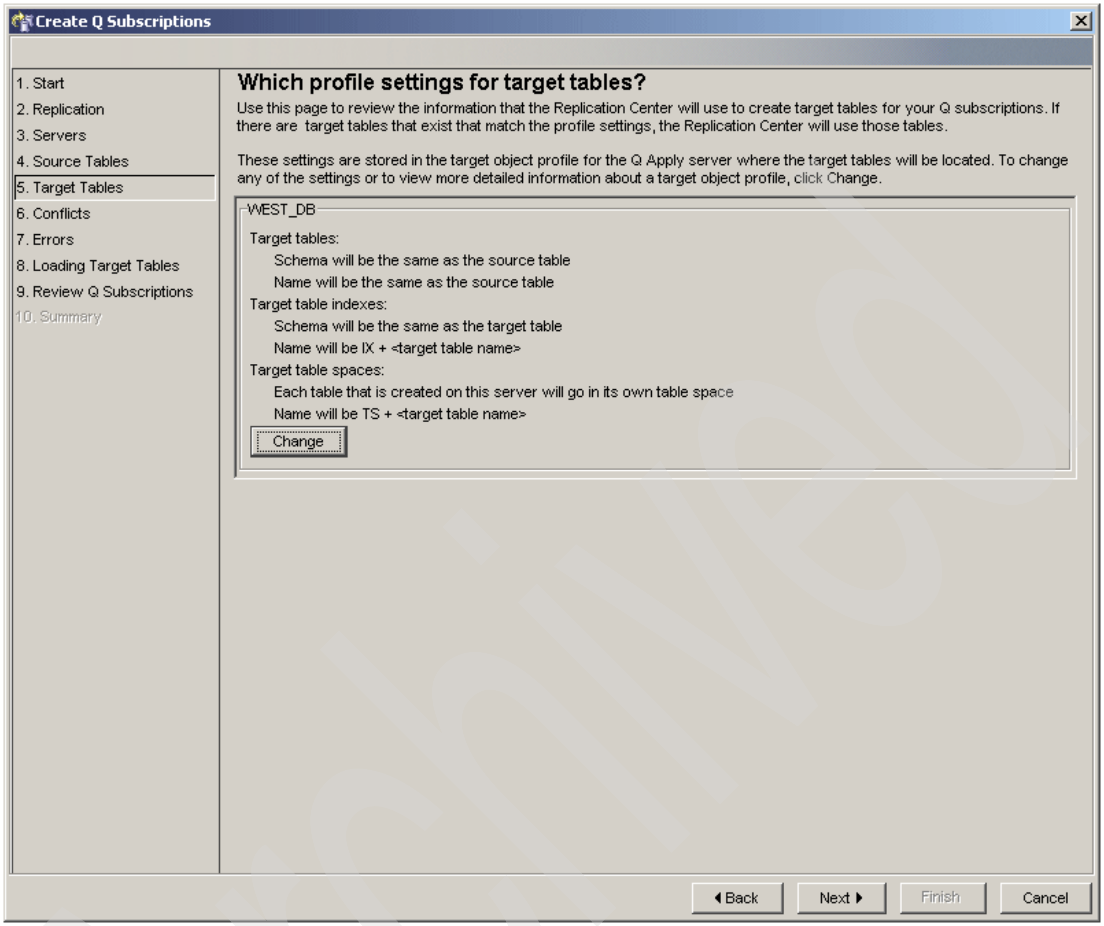


Figure 5-53 Create Q Subscriptions - Which profile settings for target tables

Figure 5-53 provides the option for changing the default settings for the target tables. We chose to go with the defaults. Click **Next** to specify the conflict resolution rules for the bidirectional topology in Figure 5-54.

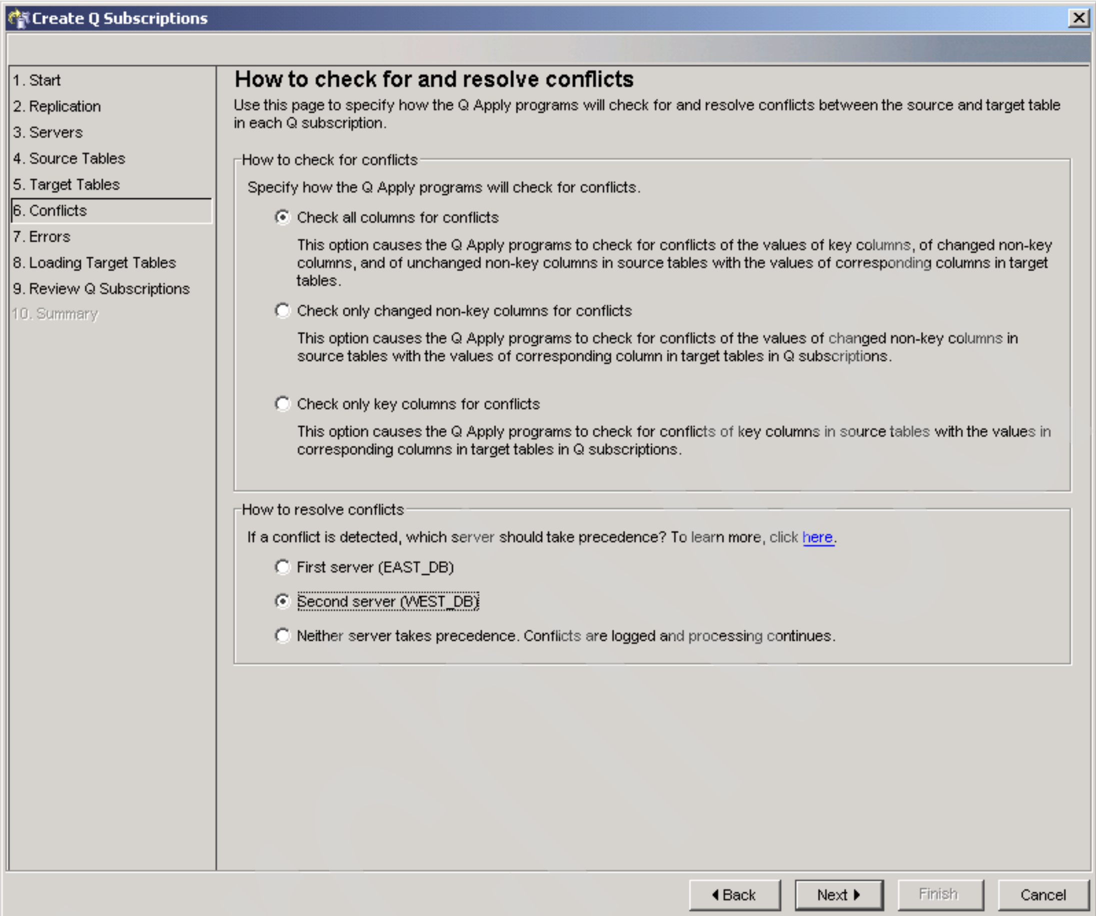


Figure 5-54 How to check for and resolve conflicts

In Figure 5-54, select **Check all columns for conflicts** and **Second server (WEST_DB)** to take precedence when a conflict is detected. Click **Next** to specify the action to be taken in the event of errors, as shown in Figure 5-55.

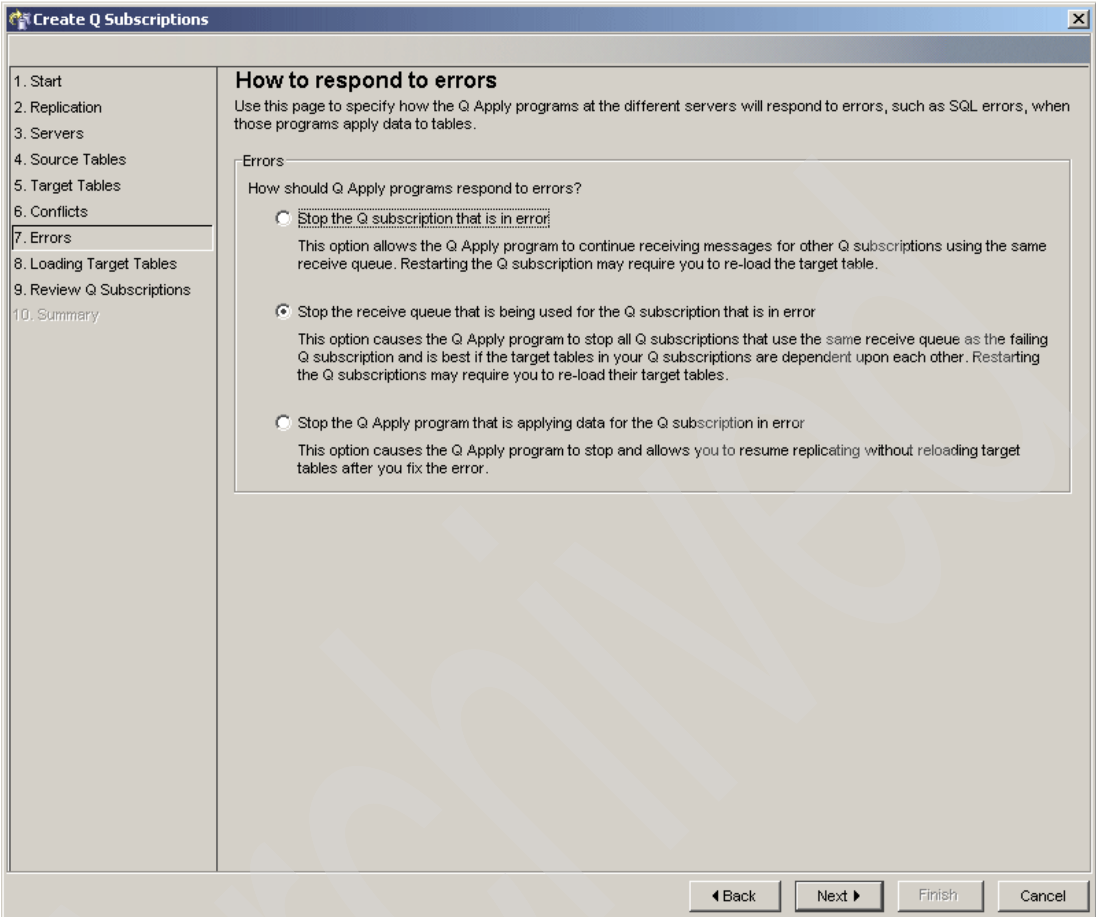


Figure 5-55 How to respond to errors

We selected the **Stop the receive queue that is being used by the Q Subscription in error** setting. Click **Next** to specify how the target tables should be loaded in Figure 5-56.

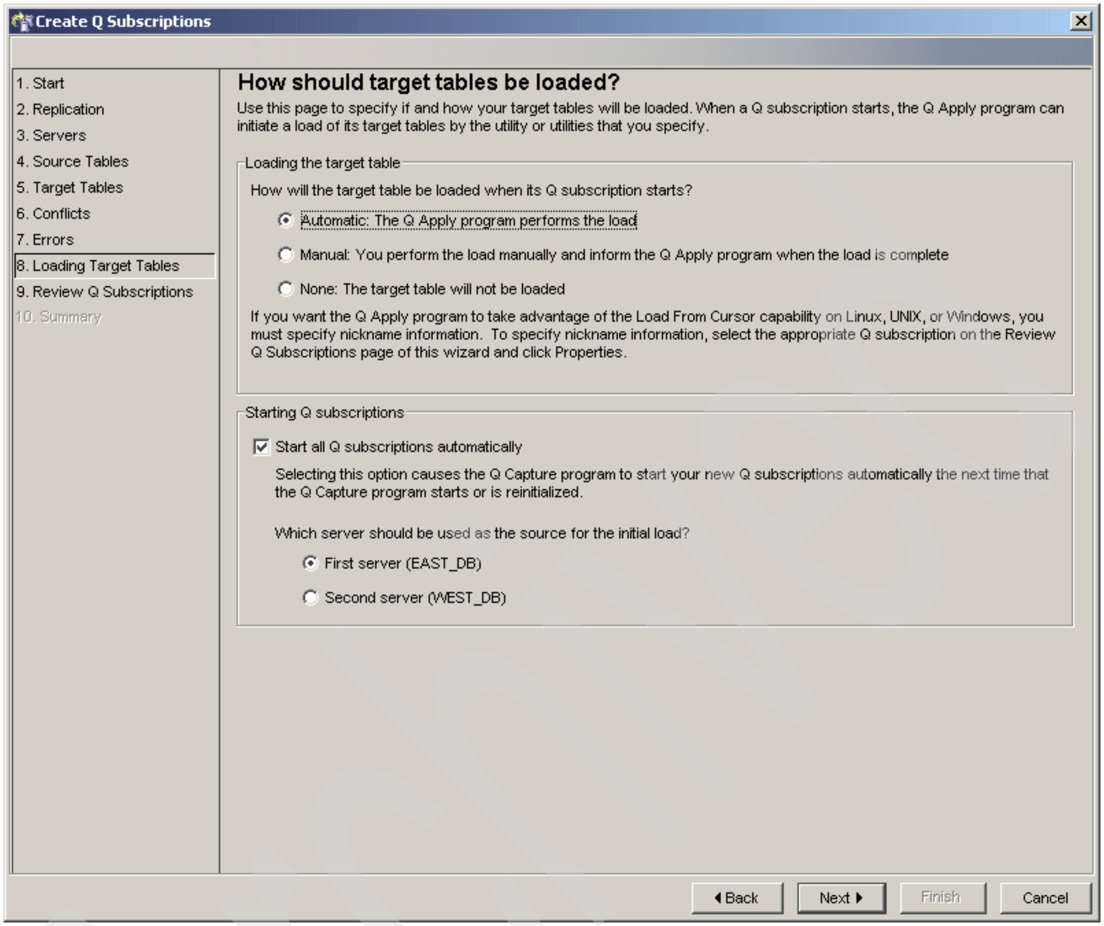


Figure 5-56 How should target tables be loaded

We chose the automatic options for both loading and starting and subscriptions, and selected EAST_DB to be the source for the initial load, as shown in Figure 5-56.

Important: Since automatic loading of the tables was chosen, a password file (asnpwd.aut) *must 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 table. Example 5-33 on page 531 creates the password file for user ID qrepladm on node EAST to access the source tables on WEST_DB, while Example 5-34 on page 531 does the same on node WEST.

Click **Next** to review the subscriptions defined, as shown in Figure 5-57.

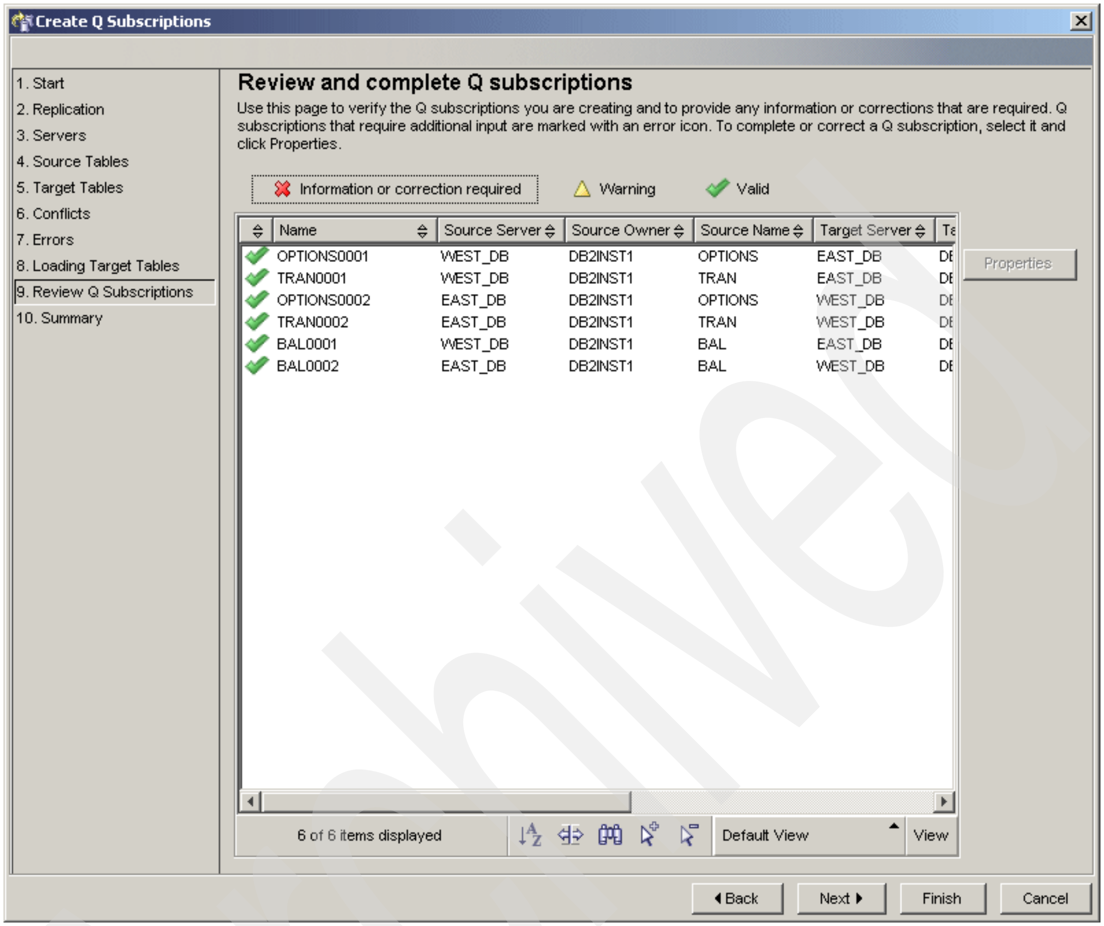


Figure 5-57 Review and complete Q subscriptions

Figure 5-57 shows six subscriptions in all—one for each of the three tables on each server. Click **Next** to complete the definition of the subscriptions and view the summary, as shown in Figure 5-58.

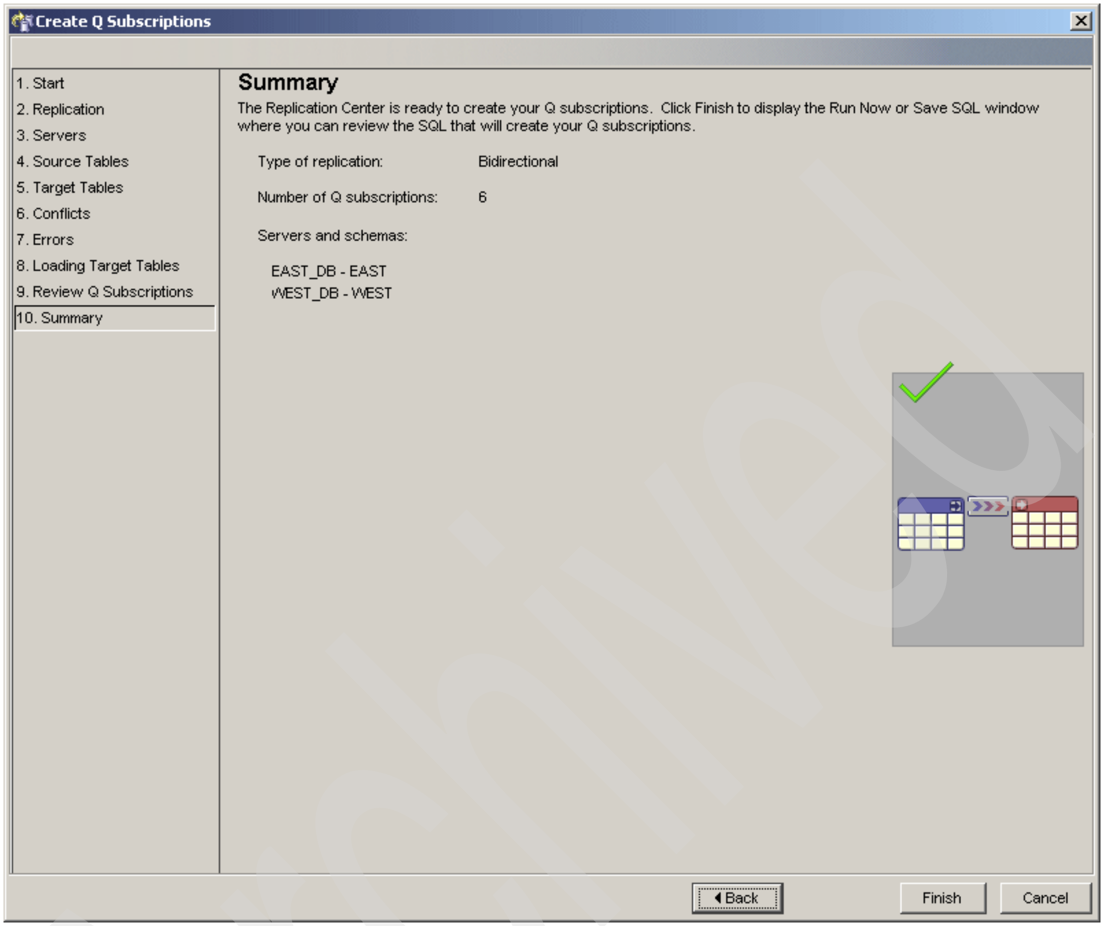


Figure 5-58 Create Q Subscriptions - Summary

Review the summary in Figure 5-58 and click **Finish** to generate the scripts for creating the various objects on the two servers, as shown in Figure 5-59.

Click the **Back** button to go back and correct any errors.

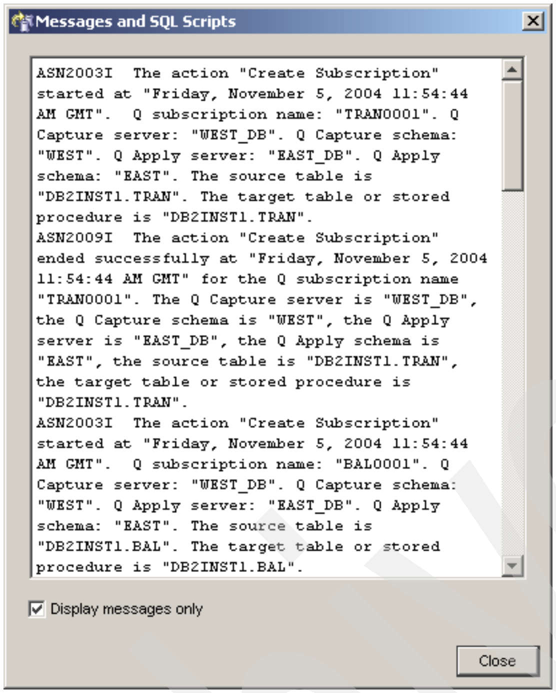


Figure 5-59 Messages and SQL Scripts

Click **Close** to decide whether to run the scripts right away or save them for later execution, as shown in Figure 5-60.

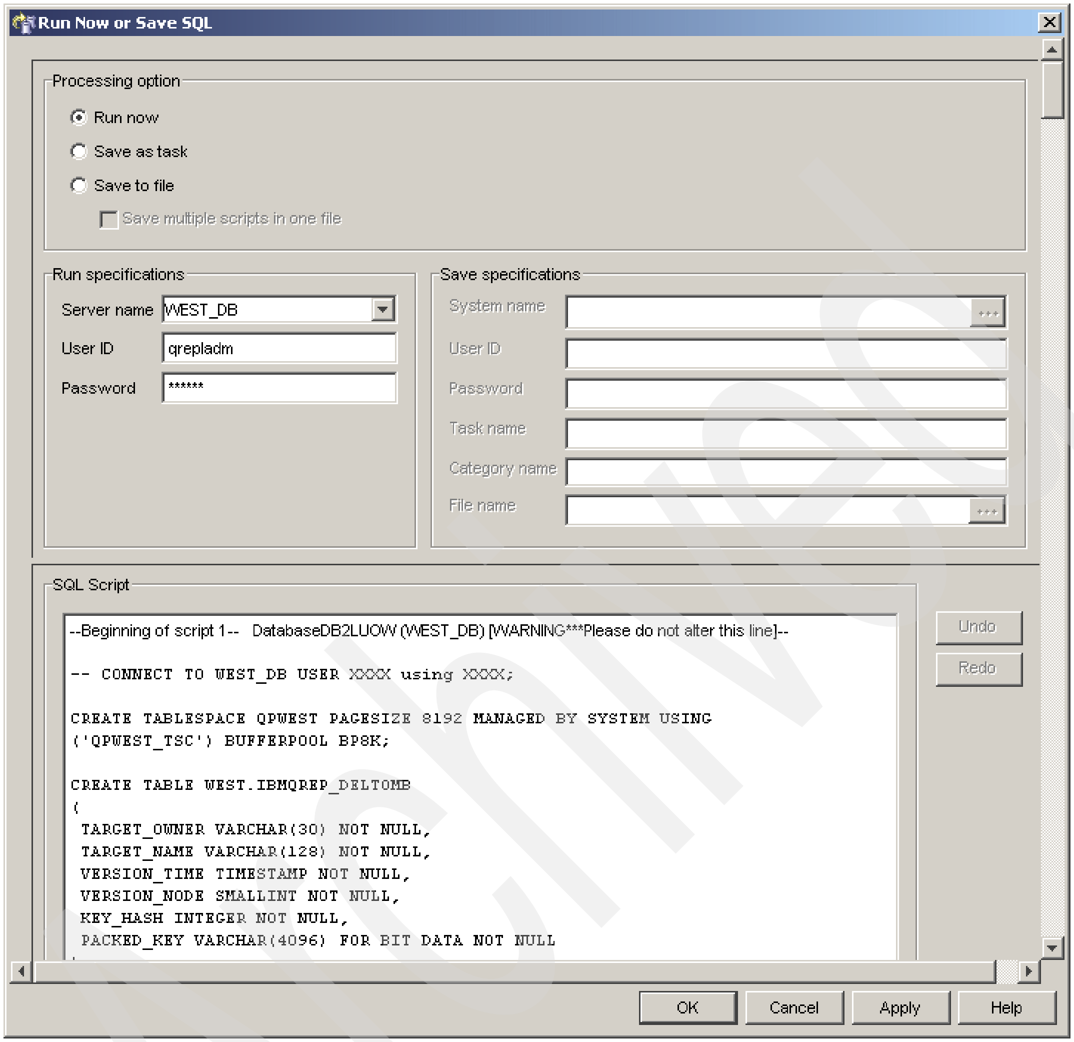


Figure 5-60 Run Now or Save SQL

We selected **Run now** and clicked **OK** to execute the scripts immediately. Figure 5-61 shows that the scripts ran successfully.

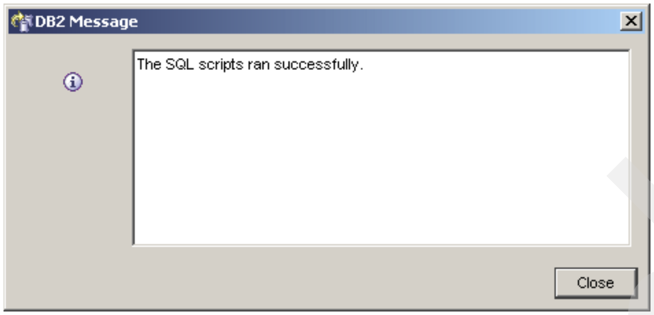


Figure 5-61 SQL scripts ran successfully message

With the completion of the subscriptions, the Q Capture program needs to be started on both servers.

Attention: Q Capture should be started before Q Apply to ensure that all messages needed for activating subscriptions are processed correctly.

Step 6f: Start the Q Capture on both servers

Click **Start Q Capture Program** in Figure 5-62 to start Q Capture on the appropriate servers.

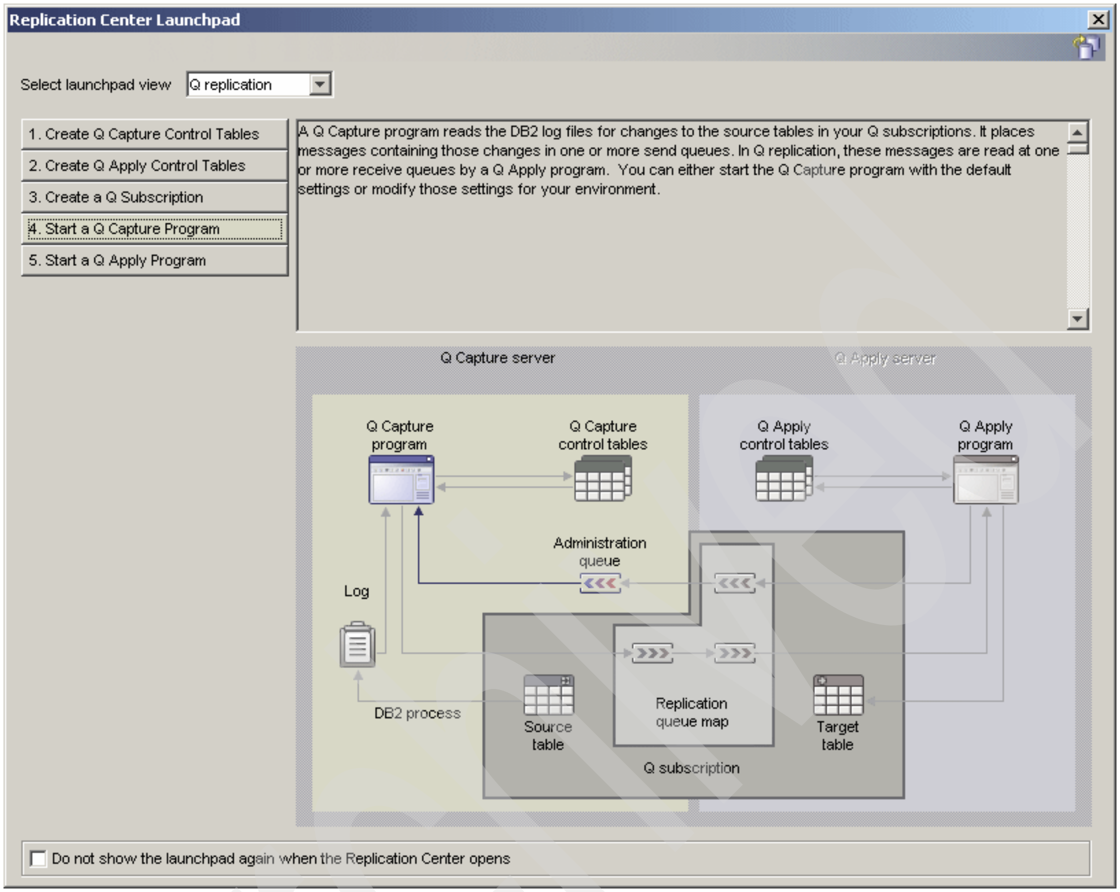


Figure 5-62 Five steps in setting up Q replication infrastructure

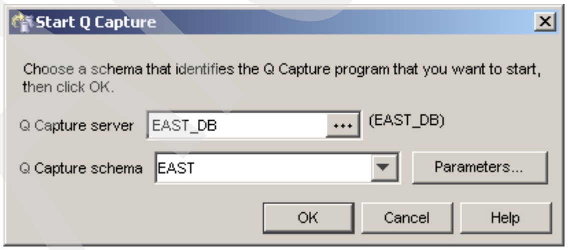


Figure 5-63 Start Q Capture

Specify the Q Capture server EAST_DB and Q Capture schema EAST, and click **Parameters...** (Figure 5-64) to specify the Q Capture startup parameters in Figure 5-65.

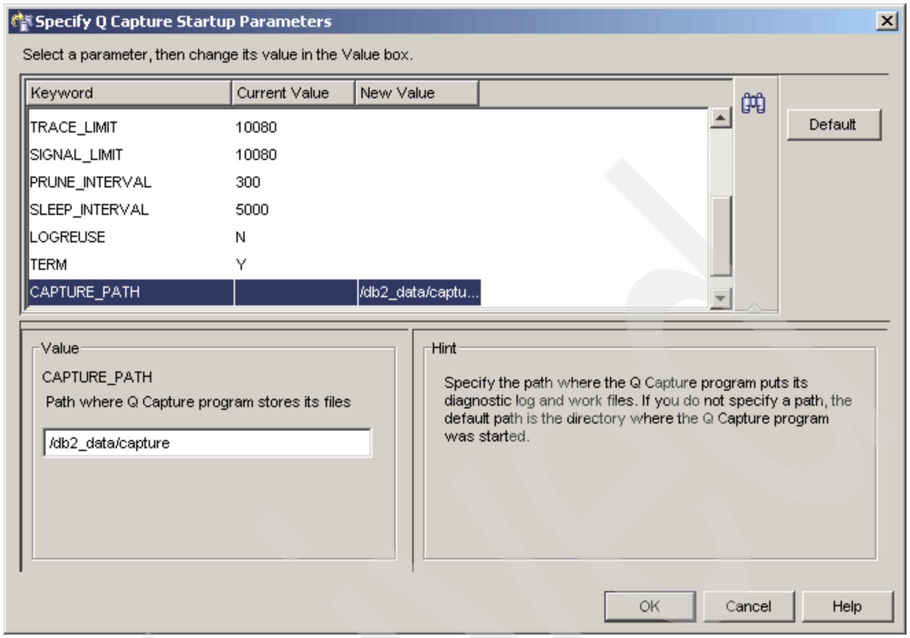


Figure 5-64 Specify Q Capture Startup Parameters

We chose the defaults for all parameters, and specified the CAPTURE_PATH parameter to be /db2_data/capture (reference A.24 in Table 5-3 on page 428). Click **OK** to generate the command for immediate execution or to be saved for later execution, as shown in Figure 5-65.

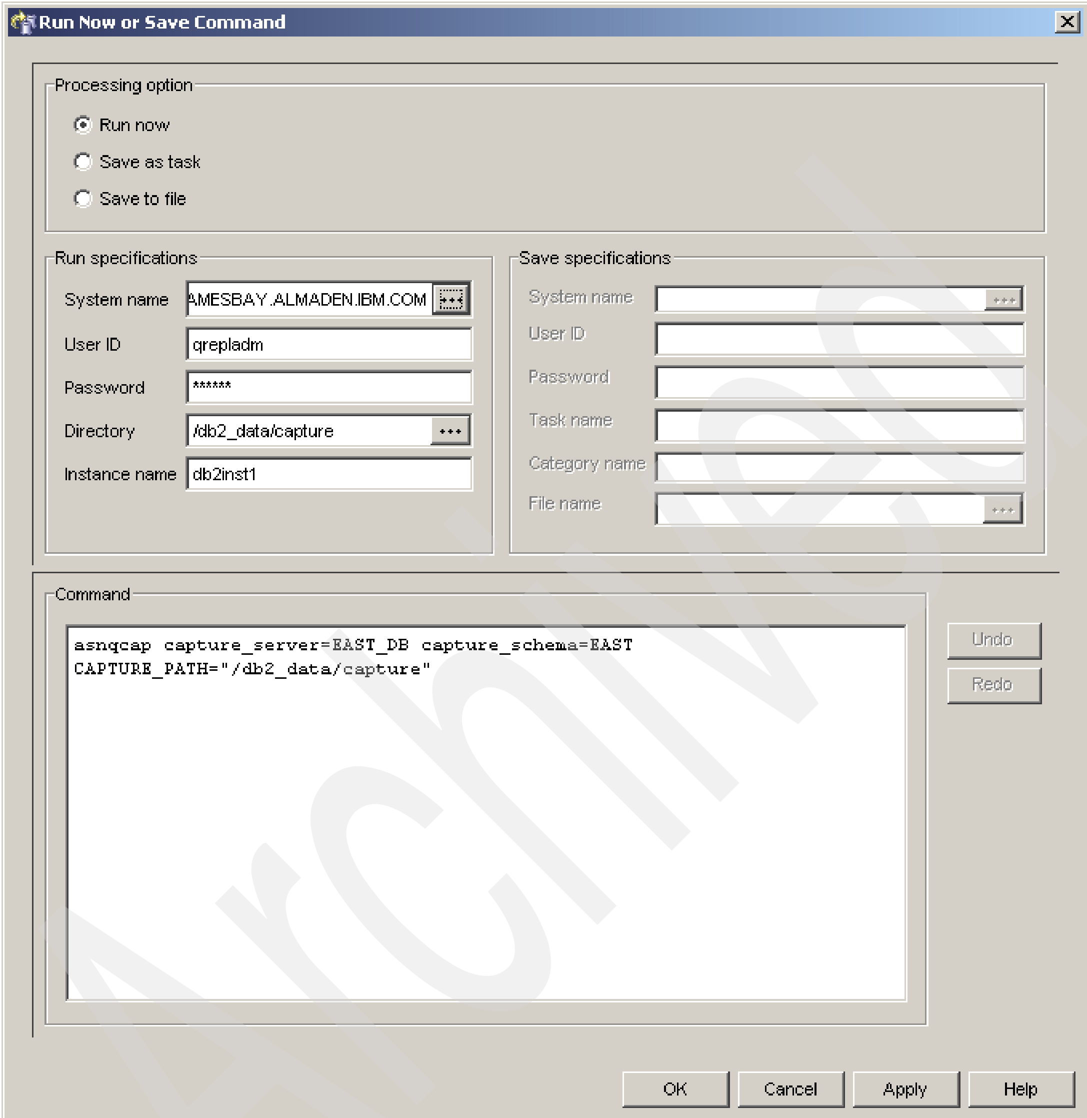


Figure 5-65 Run Now or Save Command

We selected **Run now** and clicked **OK** for the command to be executed immediately. Figure 5-66 displays the message indicating that the command was submitted.

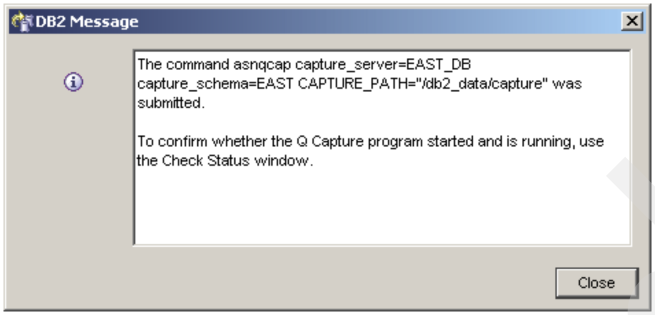


Figure 5-66 Command submitted message

> **Note:** We checked the status of the Q Capture program as described in "Step 6h: Verify status of Q Capture and Q Apply processes" on page 519.

Click **Close** to continue with the starting of Q Capture on the second server.

Repeat the steps described in Figure 5-62 on page 504 through Figure 5-66 on page 507 for the second server, as shown in Figure 5-67 through Figure 5-71 on page 511, using the appropriate reference IDs for the second server (Azov) corresponding to those of the first server (Jamesbay).

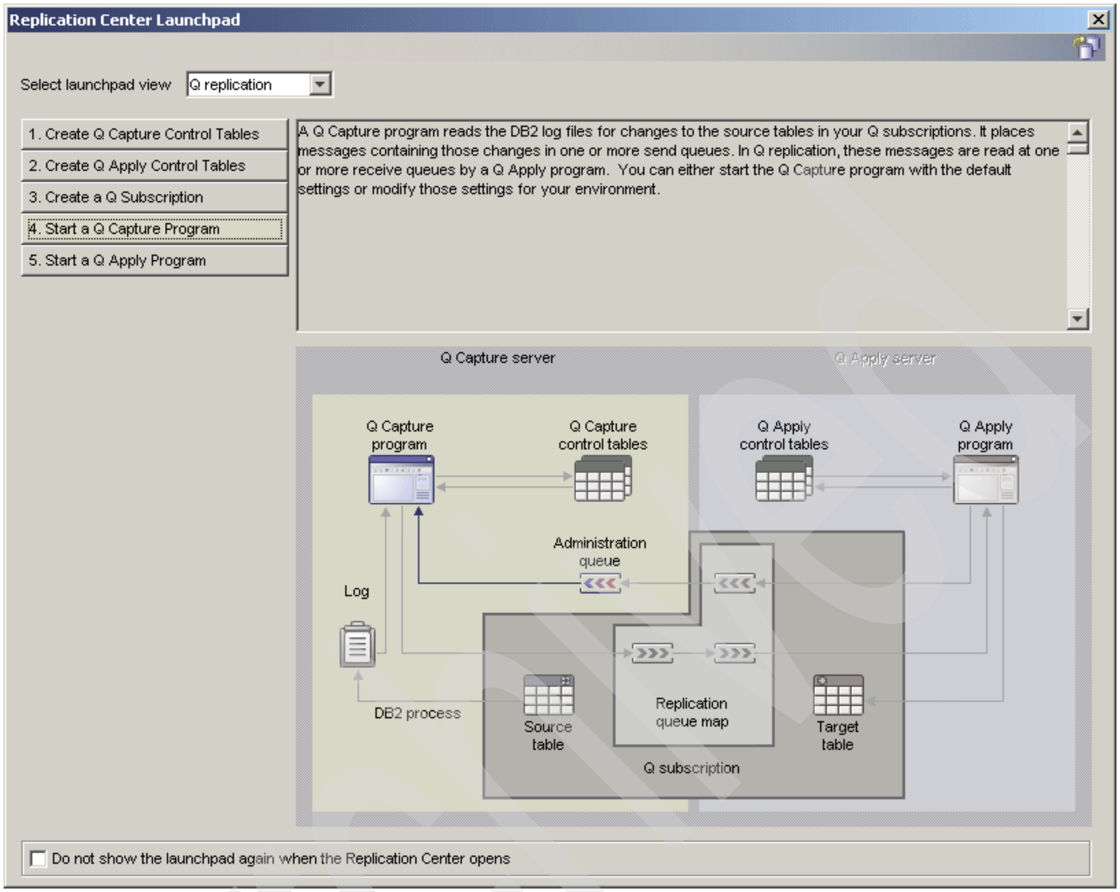


Figure 5-67 Five steps in setting up Q replication infrastructure

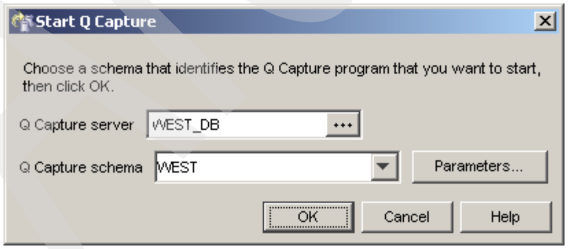


Figure 5-68 Start Q Capture

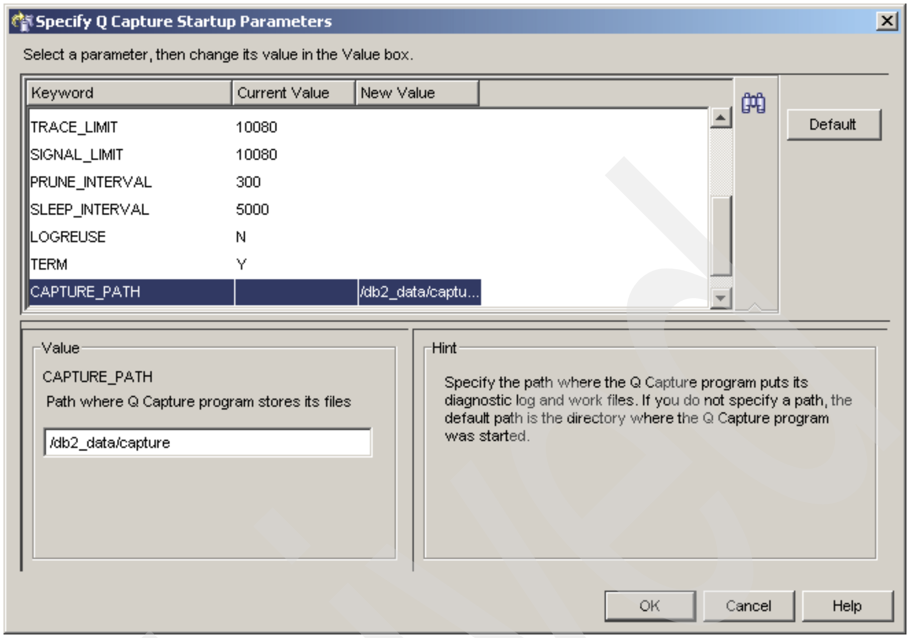


Figure 5-69 Specify Q Capture Startup Parameters

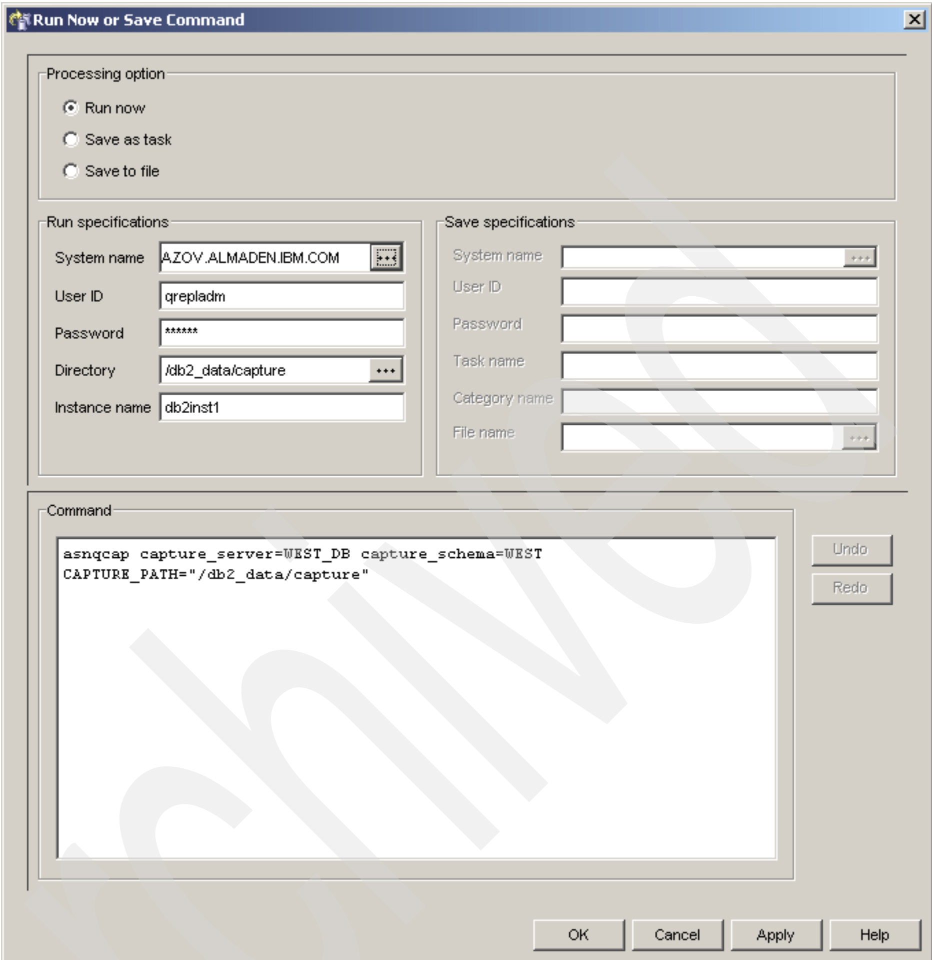


Figure 5-70 Run Now or Save Command

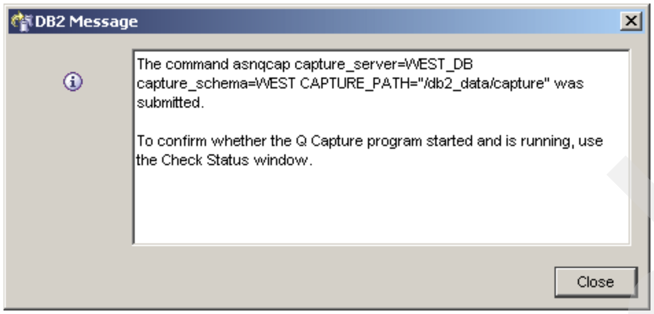


Figure 5-71 Command submitted message

The Q Apply program next needs to be started on both servers.

Step 6g: Start the Q Apply on both servers

Click **Start Q Apply Program** in Figure 5-72 to start Q Apply on the appropriate servers.

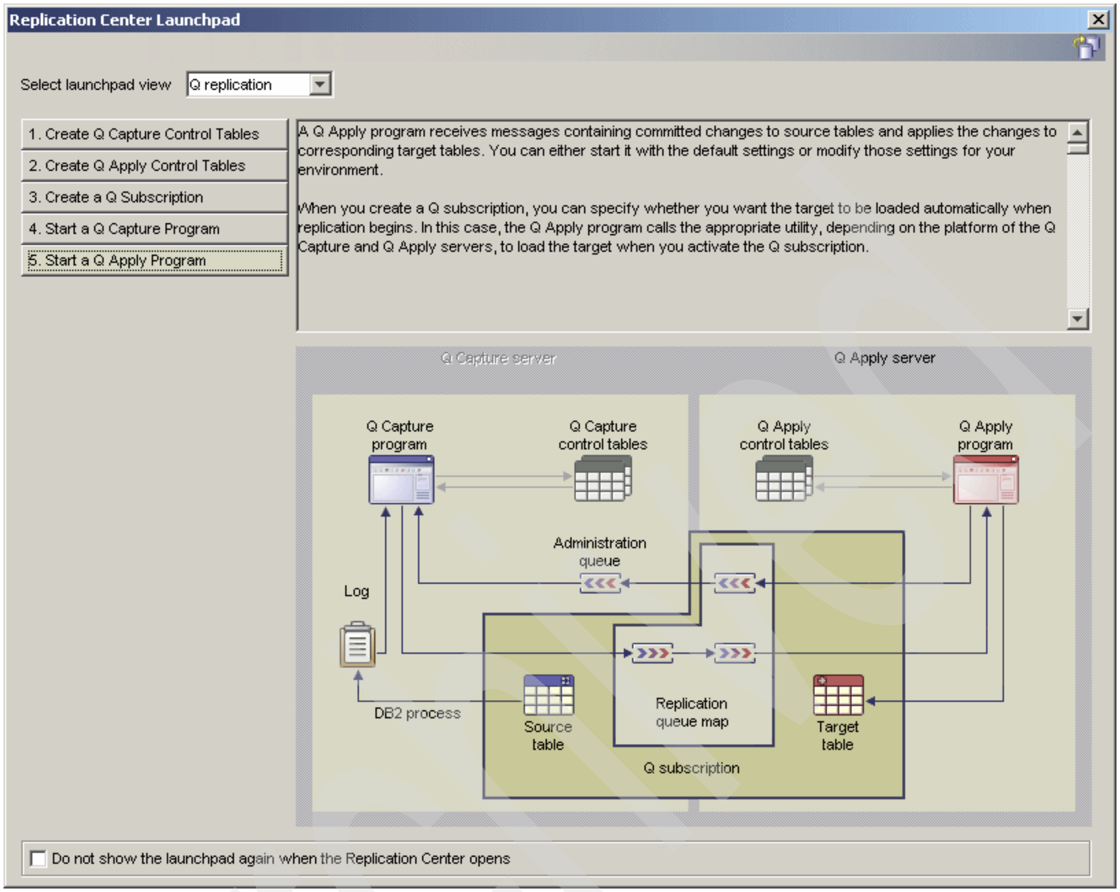


Figure 5-72 Five steps in setting up Q replication infrastructure

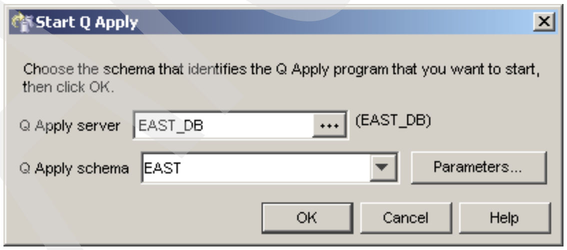


Figure 5-73 Start Q Apply

Specify the Q Apply server EAST_DB and Q Apply schema EAST, and click **Parameters...** in Figure 5-74 to specify the Q Apply startup parameters in Figure 5-74.

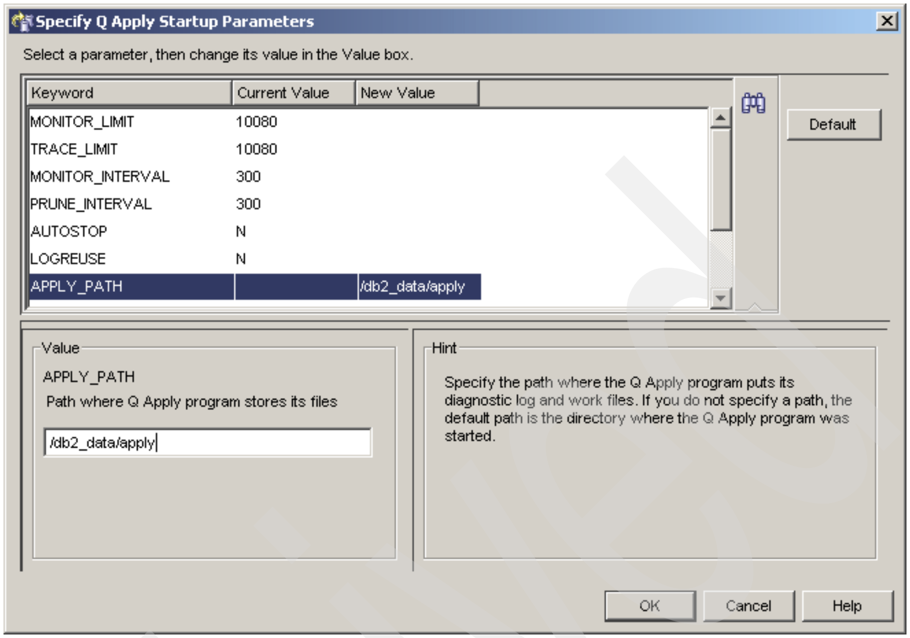


Figure 5-74 Specify Q Apply Startup Parameters

We chose the defaults for all parameters, and specified the APPLY_PATH parameter to be /db2_data/apply (reference A.25 in Table 5-3 on page 428). Click **OK** to generate the command for immediate execution or to be saved for later execution, as shown in Figure 5-75.

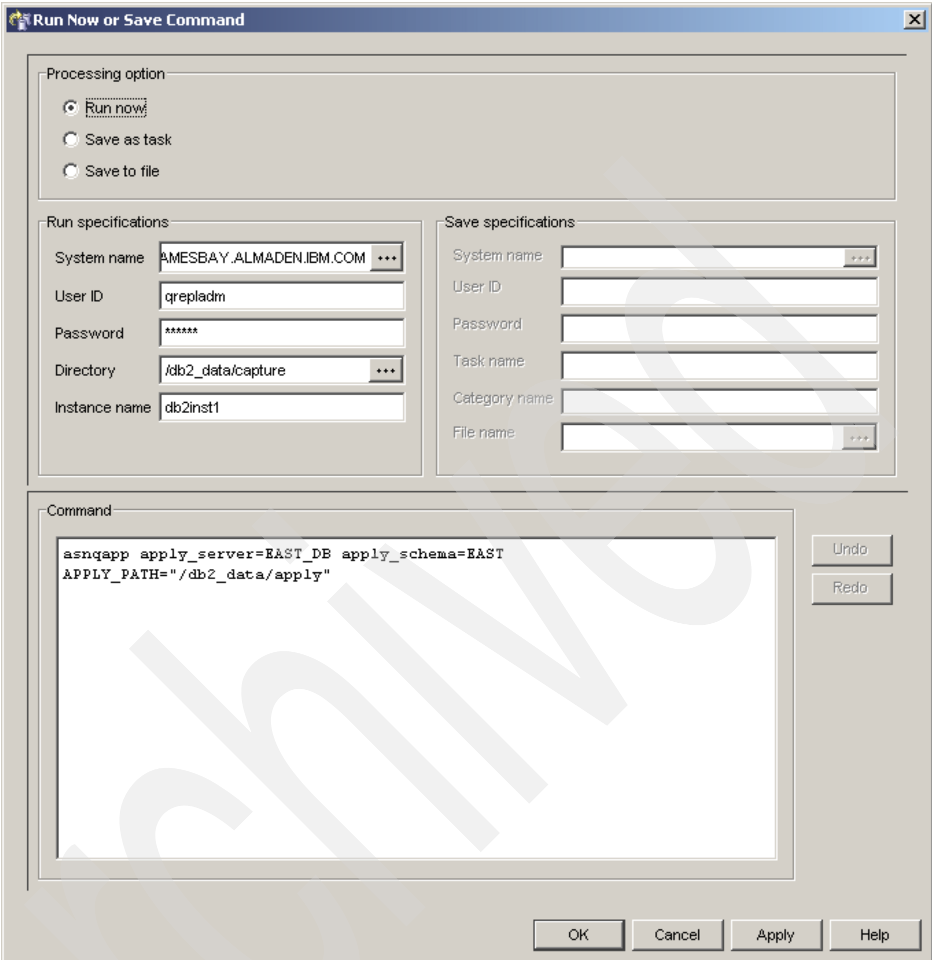


Figure 5-75 Run Now or Save Command

We selected **Run now** and clicked **OK** for the command to be executed immediately. Figure 5-76 displays the message indicating that the command was submitted.

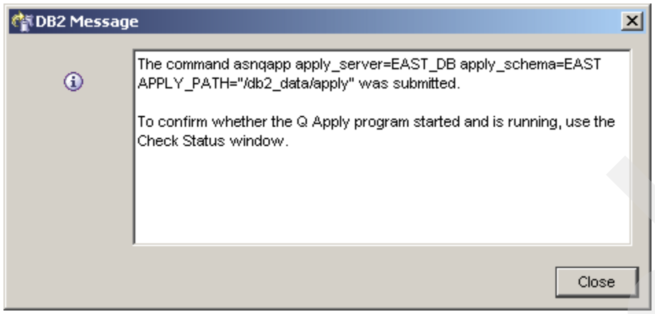


Figure 5-76 Command submitted message

> **Note:** We checked the status of the Q Apply program, as described in "Step 6h: Verify status of Q Capture and Q Apply processes" on page 519.

Click **Close** to continue with the starting of Q Apply on the second server.

Repeat the steps described in Figure 5-72 on page 512 through Figure 5-76 for the second server, as shown in Figure 5-77 through Figure 5-81 on page 519, using the appropriate reference IDs for the second server (Azov) corresponding to those of the first server (Jamesbay).

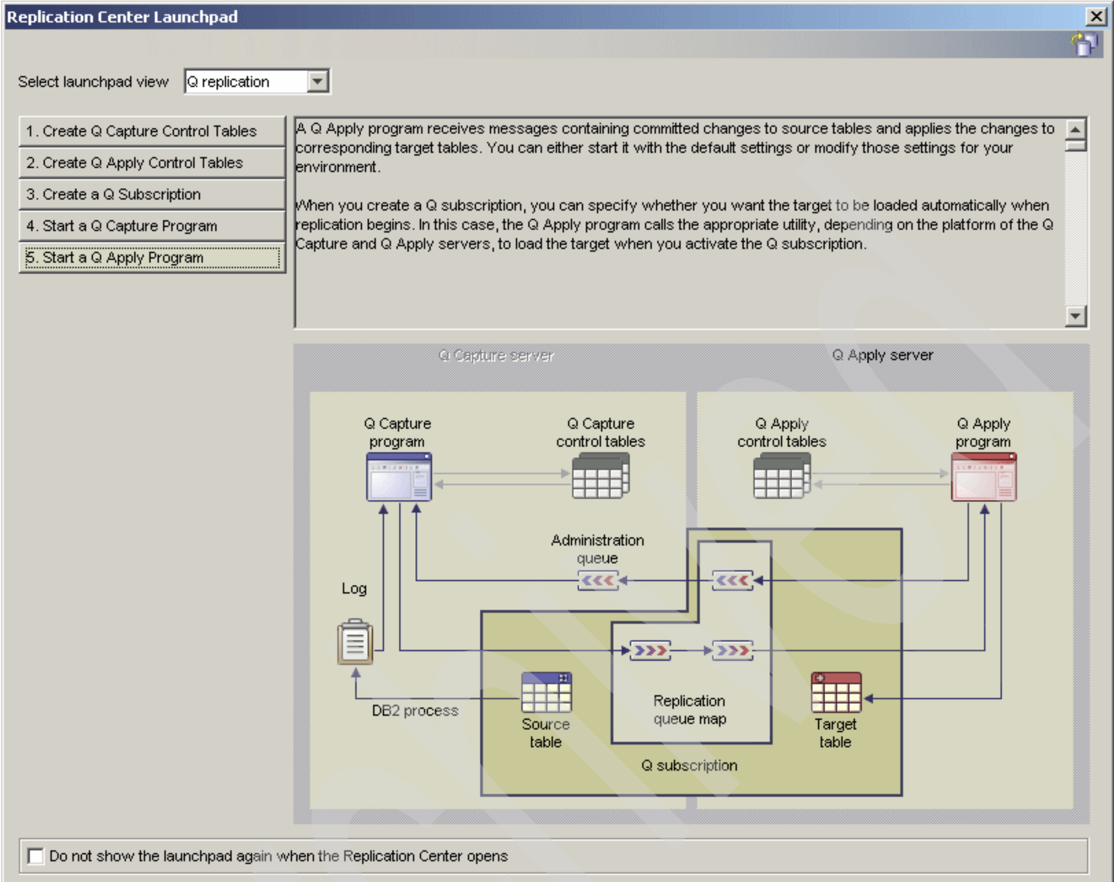


Figure 5-77 Five steps in setting up Q replication infrastructure

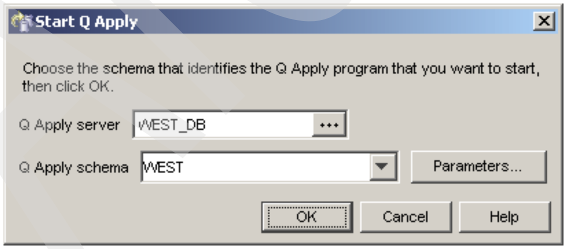


Figure 5-78 Start Q Apply

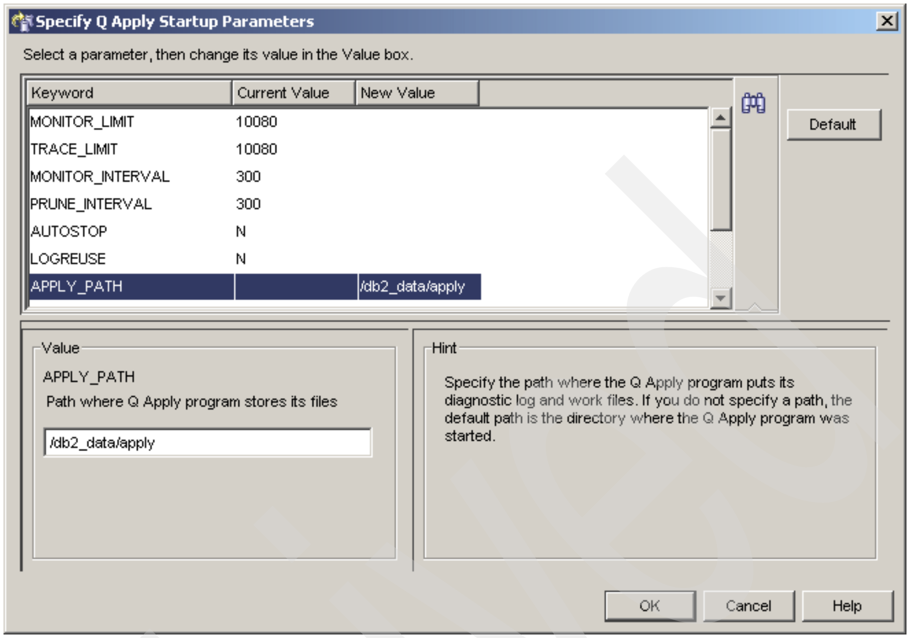


Figure 5-79 Specify Q Apply Startup Parameters

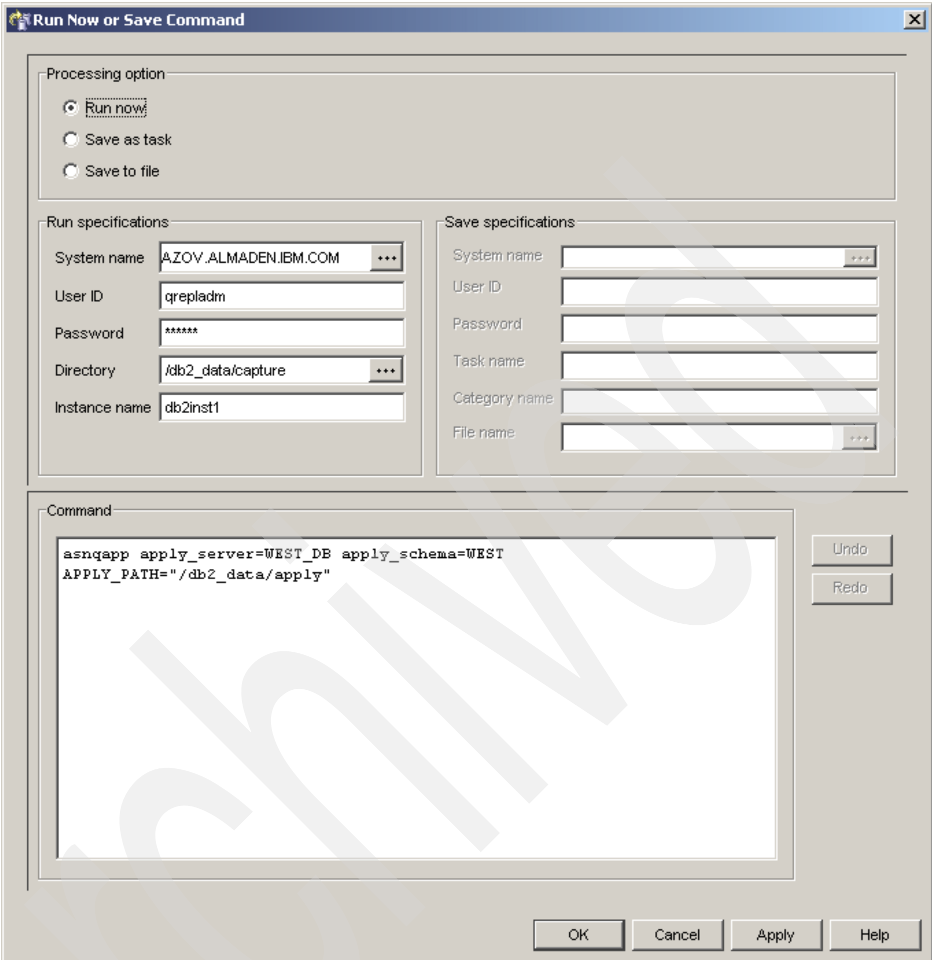


Figure 5-80 Run Now or Save Command

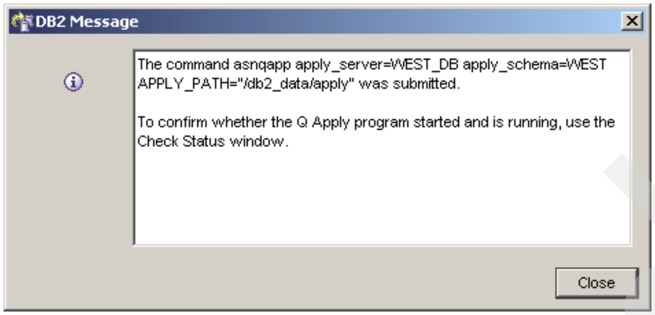


Figure 5-81 Command submitted message

Step 6h: Verify status of Q Capture and Q Apply processes

Here we review the status of the Q Capture and Q Apply processes started earlier.

A quick way of determining whether the Q Capture and Q Apply processes are running on a server is to issue the AIX **ps** command, as shown in Example 5-23.

Example 5-23 Checking for Q Capture and Q Apply processes

```
$ ps -ef | grep asn
qrepladm 49936   1   0   Nov 17      -  3:22 asnqcap capture_server=STOCK_W capture_schema=PEER2
qrepladm 51964   1   0   Nov 17      -  1:58 asnqapp apply_server=STOCK_W apply_schema=PEER2
```

The recommended approach is to select the **Q Capture Servers** folder in the navigation pane of the Replication Center, as shown in Figure 5-82. This lists EAST_DB on Jamesbay and WEST_DB on Azov as the two Q Capture Servers in the right pane.

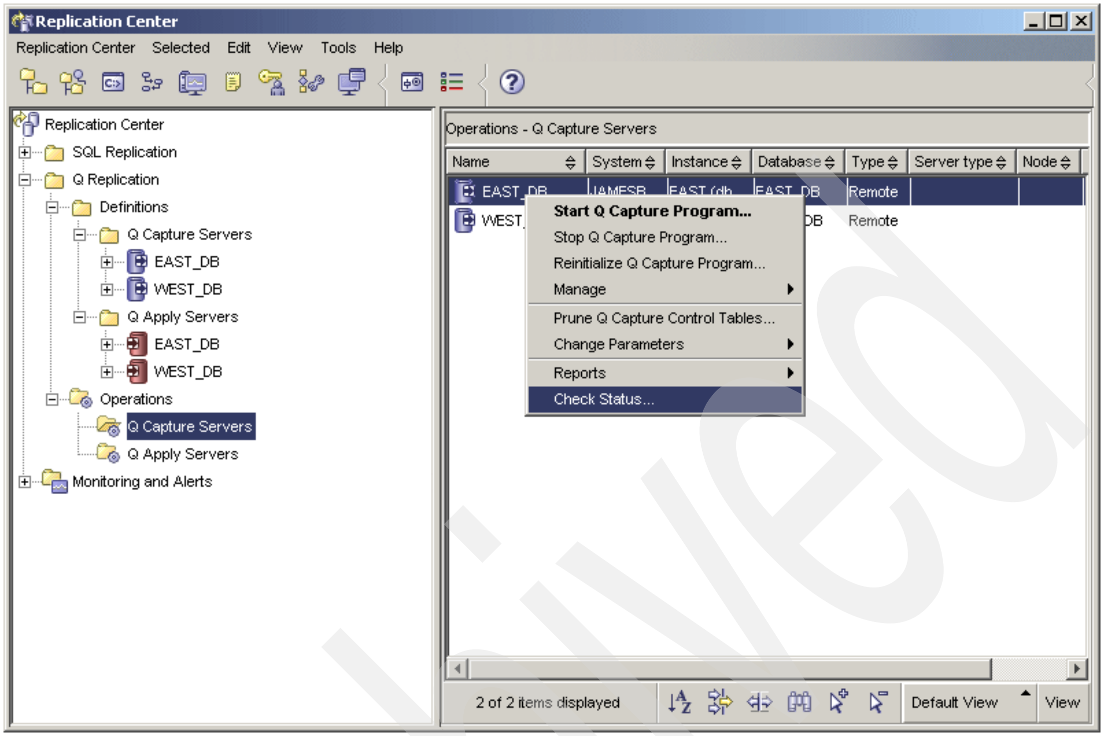


Figure 5-82 Check Status of the EAST_DB Q Capture Server

Select the row with EAST_DB in the Name column and right-click to obtain a drop-down list. Select **Check Status...**, as shown in Figure 5-82, which displays Figure 5-83.

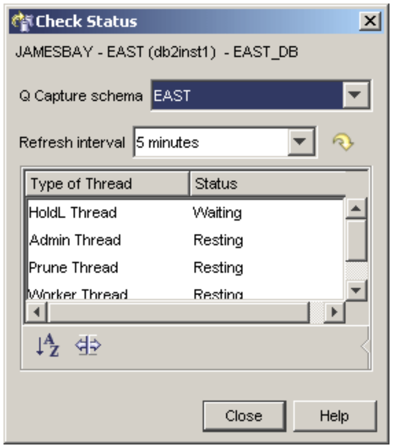


Figure 5-83 Check Status of EAST_DB Q Capture Server threads

Figure 5-83 shows four threads—Holdl (Waiting), Admin (Resting). Prune (Resting), and Worker (Resting)—indicating that the Q Capture program is operating properly.

To check the status of the Q Apply program, select the **Q Apply Servers** folder in the navigation pane of the Replication Center, as shown in Figure 5-84. This lists EAST_DB on Jamesbay and WEST_DB on Azov as the two Q Apply Servers in the right pane.

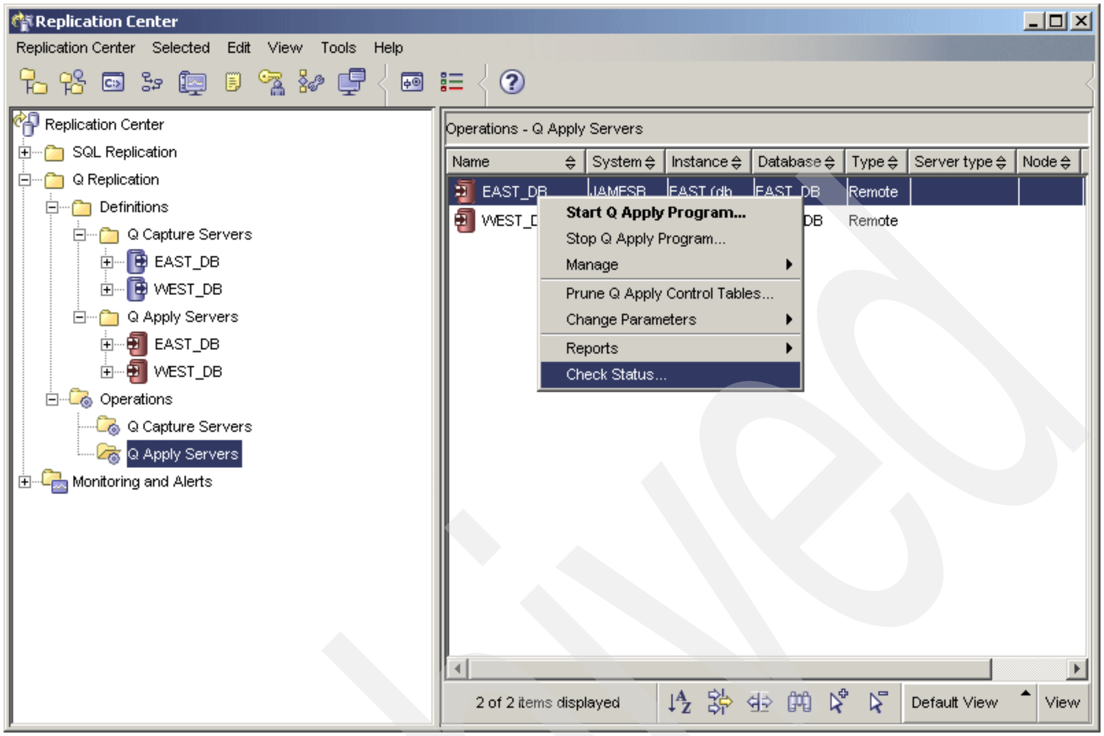


Figure 5-84 Check Status of the EAST_DB Q Apply Server

Select the row with EAST_DB in the Name column and right-click to obtain a drop-down list. Select **Check Status...** (as shown in Figure 5-84), which displays Figure 5-85.

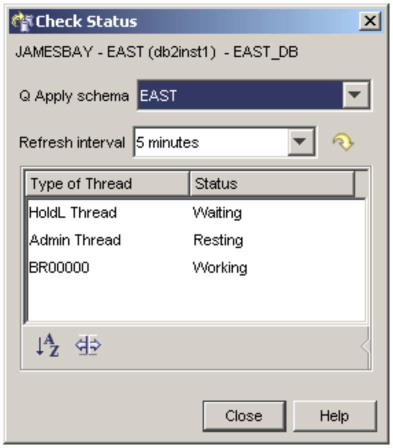


Figure 5-85 Check Status of the EAST_DB Q Apply Server threads

Figure 5-85 shows three threads—Holdl (Waiting), Admin (Resting). and browser BR00000 (Working)—indicating proper operation of the Q Apply program.

Step 6i: Perform manual load if appropriate

Since we chose automatic loading of the tables involved in the subscriptions, this step does not apply in our scenario.

Figure 5-86 shows the three subscriptions defined for the EAST_DB server by selecting the Q Subscriptions folder in the navigation pane for EAST.

This completes the setup of the bidirectional Q replication implementation for Corona.

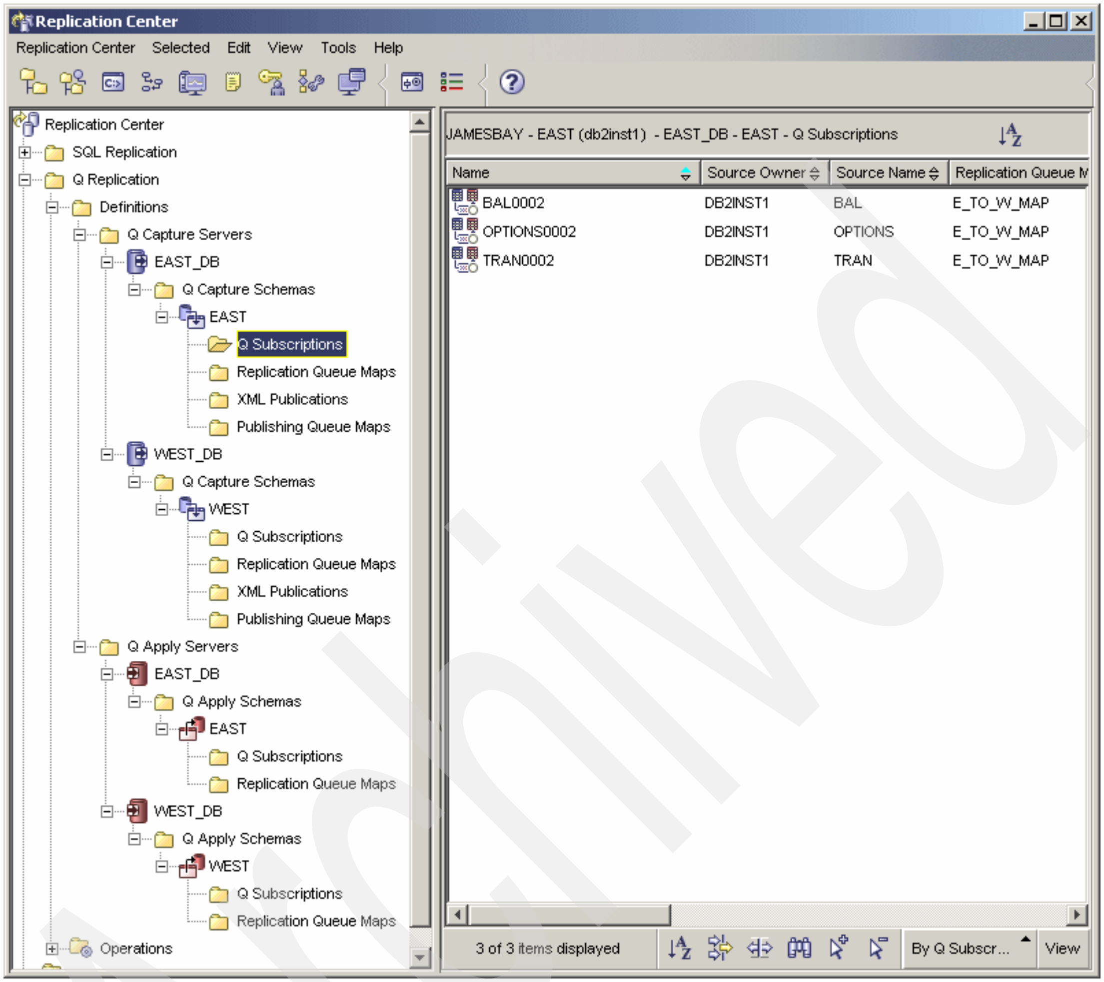


Figure 5-86 List of Q Subscriptions on EAST_DB server

5.5.7 Step 6: Configure and activate Q replication using commands

WebSphere II Q replication configuration may be performed using GUI as described in 5.5.6, “Step 6: Configure and activate Q replication using GUI” on page 448, or by using ASNCLP commands from the Unix command line. This section describes the configuration of the same bidirectional Q replication topology solution 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 5-24.

Example 5-24 Set up Java environment for user ID qrepladm

```
PATH=$PATH:/usr/java14/bin

CP=/db2_data/db2inst1/sqllib/java/Common.jar
CP=$CP:/db2_data/db2inst1/sqllib/tools/db2cmn.jar
CP=$CP:/db2_data/db2inst1/sqllib/tools/db2replapis.jar
CP=$CP:/db2_data/db2inst1/sqllib/tools/db2qreplapis.jar
CP=$CP:/db2_data/db2inst1/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 setup was successful. Enter `quit` to exit the Repl> command line, as shown in Example 5-25.

Example 5-25 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 5-26.

Example 5-26 Execute commands from a file

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

The ASNCLP commands in the following sections are categorized according to the following tasks:

1. Create the Q Capture and Q Apply control tables.
2. Create the replication queue maps.

3. Create the subscriptions.
4. Create a password file.
5. Update the CAPTURE_PATH and APPLY_PATH.
6. Start the Q Capture and Q Apply programs.
7. Verify successful implementation.

Create the Q Capture and Q Apply control tables

Example 5-27 lists the ASNCLP commands for creating the Q Capture and Q Apply control tables on node EAST (Jamesbay), while Example 5-28 does the same on node WEST (Azov).

Example 5-27 Create the Q Capture and Q Apply control tables on EAST

```
####################################################
# The following commands create the control
# tables in an existing tablespace USERSPACE1. If
# a tablespaces needs to be created automatically
# the user ID (qrepladm) must have SYSCTRL
# authority.
####################################################
ASNCLP SESSION SET TO Q REPLICATION;
SET OUTPUT MULTIDIR;
#############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 5-1 on page 425, Table 5-2 on page 426 and Table 5-3 on page 428
############################################################
##EAST_DB from reference A.7a in Table 5-1 on page 425
##QREPLADM from reference A.7b in Table 5-1 on page 425
##QM_EAST from reference A.11 in Table 5-2 on page 426
##Capture schema EAST from reference A.22 in Table 5-3 on page 428
##EAST_RESTARTQ from reference A.16 in Table 5-2 on page 426
##EAST_ADMINQ from reference A.17 in Table 5-2 on page 426
#############################################################
SET SERVER CAPTURE TO DB EAST_DB ID QREPLADM PASSWORD "xxxxxx";
SET QMANAGER "QM_EAST" FOR CAPTURE SCHEMA;
SET CAPTURE SCHEMA SOURCE EAST;
SET RUN SCRIPT NOW STOP ON SQL ERROR ON;
CREATE CONTROL TABLES FOR CAPTURE SERVER USING RESTARTQ "EAST_RESTARTQ" ADMINQ
"EAST_ADMINQ" STARTMODE WARMSI IN UW TBSPACE USERSPACE1;
############################################################
##EAST_DB from reference A.7a in Table 5-1 on page 425
##QREPLADM from reference A.7c in Table 5-1 on page 425
##QM_EAST from reference A.11 in Table 5-2 on page 426
##Apply schema EAST from reference A.23 in Table 5-3 on page 428
#############################################################
SET SERVER TARGET TO DB EAST_DB ID QREPLADM PASSWORD "xxxxxx";
SET QMANAGER "QM_EAST" FOR APPLY SCHEMA;
```

```
SET APPLY SCHEMA EAST;
SET RUN SCRIPT NOW STOP ON SQL ERROR ON;
CREATE CONTROL TABLES FOR APPLY SERVER IN UW TBSPACE USERSPACE1;
```

Example 5-28 Create the Q Capture and Q Apply control tables on WEST

```
######################################################
# The following commands create the control
# tables in an existing tablespace USERSPACE1.  If
# a tablespaces needs to be created automatically
# the user ID (qrepladm) must have SYSCTRL
# authority.
#####################################################
ASNCLP SESSION SET TO Q REPLICATION;
SET OUTPUT MULTIDIR;
#############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 5-1 on page 425, Table 5-2 on page 426 and Table 5-3 on page 428
############################################################
##WEST_DB from reference B.7a in Table 5-1 on page 425
##QREPLADM from reference B.7b in Table 5-1 on page 425
##QM_WEST from reference B.11 in Table 5-2 on page 426
##Capture schema WEST from reference B.22 in Table 5-3 on page 428
##WEST_RESTARTQ from reference B.16 in Table 5-2 on page 426
##WEST_ADMINQ from reference B.17 in Table 5-2 on page 426
#############################################################
SET SERVER CAPTURE TO DB WEST_DB ID QREPLADM PASSWORD "xxxxxx";
SET QMANAGER "QM_WEST" FOR CAPTURE SCHEMA;
SET CAPTURE SCHEMA SOURCE WEST;
SET RUN SCRIPT NOW STOP ON SQL ERROR ON;
CREATE CONTROL TABLES FOR CAPTURE SERVER USING RESTARTQ "WEST_RESTARTQ" ADMINQ
"WEST_ADMINQ" STARTMODE WARMSI IN UW TBSPACE USERSPACE1;
############################################################
##WEST_DB from reference B.7a in Table 5-1 on page 425
##QREPLADM from reference B.7c in Table 5-1 on page 425
##QM_WEST from reference B.11 in Table 5-2 on page 426
##Apply schema WEST from reference B.23 in Table 5-3 on page 428
#############################################################
SET SERVER TARGET TO DB WEST_DB ID QREPLADM PASSWORD "xxxxxx";
SET QMANAGER "QM_WEST" FOR APPLY SCHEMA;
SET APPLY SCHEMA WEST;
SET RUN SCRIPT NOW STOP ON SQL ERROR ON;
CREATE CONTROL TABLES FOR APPLY SERVER IN UW TBSPACE USERSPACE1;
```

Create the replication queue maps

Example 5-29 lists the ASNCLP commands for creating the replication queue maps on node EAST (Jamesbay), while Example 5-30 does the same on node WEST (Azov).

Example 5-29 Create the replication queue maps on EAST

```
#############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 5-1 on page 425, Table 5-2 on page 426 and Table 5-3 on page 428
############################################################
##EAST_DB from reference A.7a in Table 5-1 on page 425
##QREPLADM from reference A.7b in Table 5-1 on page 425
##WEST_DB from reference B.7a in Table 5-1 on page 425
##QREPLADM from reference B.7c in Table 5-1 on page 425
##Capture schema EAST from reference A.22 in Table 5-3 on page 428
##Apply schema WEST from reference B.23 in Table 5-3 on page 428
##E_TO_W_MAP from reference A.27 in Table 5-3 on page 428
##EAST_ADMINQ from reference A.17 in Table 5-2 on page 426
##RECVQ EAST_TO_WEST_Q from reference B.19 in Table 5-2 on page 426
##SENDQ EAST_TO_WEST_Q from reference A.18 in Table 5-2 on page 426
#############################################################

ASNCLP SESSION SET TO Q REPLICATION;
SET OUTPUT MULTIDIR;

SET SERVER CAPTURE TO DB EAST_DB ID "QREPLADM" PASSWORD "xxxxxx";
SET SERVER TARGET  TO DB WEST_DB ID "QREPLADM" PASSWORD "xxxxxx";

SET CAPTURE SCHEMA SOURCE EAST;
SET APPLY   SCHEMA WEST;
SET RUN SCRIPT NOW STOP ON SQL ERROR ON;
CREATE REPLQMAP E_TO_W_MAP USING ADMINQ "EAST_ADMINQ" RECVQ "EAST_TO_WEST_Q"
SENDQ "EAST_TO_WEST_Q" NUM APPLY AGENTS 4 MEMORY LIMIT 30 ERROR ACTION I MAX
MESSAGE SIZE 2048;
```

Example 5-30 Create the replication queue maps on WEST

```
############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 5-1 on page 425, Table 5-2 on page 426 and Table 5-3 on page 428
############################################################
##EAST_DB from reference A.7a in Table 5-1 on page 425
##QREPLADM from reference A.7b in Table 5-1 on page 425
##WEST_DB from reference B.7a in Table 5-1 on page 425
##QREPLADM from reference B.7c in Table 5-1 on page 425
##Capture schema WEST from reference B.22 in Table 5-3 on page 428
```

```
##Apply schema EAST from reference A.23 in Table 5-3 on page 428
##W_TO_E_MAP from reference B.27 in Table 5-3 on page 428
##WEST_ADMINQ from reference B.17 in Table 5-2 on page 426
##RECVQ WEST_TO_EAST_Q from reference A.19 in Table 5-2 on page 426
##SENDQ WEST_TO_EAST_Q from reference B.18 in Table 5-2 on page 426
#############################################################
ASNCLP SESSION SET TO Q REPLICATION;
SET OUTPUT MULTIDIR;

SET SERVER CAPTURE TO DB WEST_DB ID "QREPLADM" PASSWORD "xxxxxx";
SET SERVER TARGET  TO DB EAST_DB ID "QREPLADM" PASSWORD "xxxxxx";

SET CAPTURE SCHEMA SOURCE WEST;
SET APPLY   SCHEMA EAST;
SET RUN SCRIPT NOW STOP ON SQL ERROR ON;
CREATE REPLQMAP W_TO_E_MAP USING ADMINQ "WEST_ADMINQ" RECVQ "WEST_TO_EAST_Q"
SENDQ "WEST_TO_EAST_Q" NUM APPLY AGENTS 4 MEMORY LIMIT 30 ERROR ACTION I MAX
MESSAGE SIZE 2048;
```

Create the subscriptions

Example 5-31 and Example 5-32 list the ASNCLP commands for creating the subscriptions on node EAST (Jamesbay) and node WEST (Azov).

Example 5-31 Create the subscriptions

```
#####################################################
# Create subscriptions on EAST
#####################################################
ASNCLP SESSION SET TO Q REPLICATION;
SET OUTPUT MULTIDIR;
SET RUN SCRIPT NOW STOP ON SQL ERROR ON;
LOAD MULTIDIR REPL SCRIPT "subs1";
```

Example 5-32 Content of the "subs1" file

```
#############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 5-1 on page 425, Table 5-2 on page 426 and Table 5-3 on page 428
############################################################
##EAST_DB from reference A.7a in Table 5-1 on page 425
##QREPLADM from reference A.7b in Table 5-1 on page 425
##WEST_DB from reference B.7a in Table 5-1 on page 425
##QREPLADM from reference B.7c in Table 5-1 on page 425
##Capture schema EAST from reference A.22 in Table 5-3 on page 428
##Apply schema WEST from reference B.23 in Table 5-3 on page 428
```

```
##E_TO_W_MAP from reference A.27 in Table 5-3 on page 428
##W_TO_E_MAP from reference B.27 in Table 5-3 on page 428
##EAST_DB.EAST.QREPLADM.BAL & WEST_DB.WEST.QREPLADM.BAL table pair
##EAST_DB.EAST.QREPLADM.TRAN & WEST_DB.WEST.QREPLADM.TRAN table pair
##EAST_DB.EAST.QREPLADM.OPTIONS & WEST_DB.WEST.QREPLADM.OPTIONS table pair
#############################################################
SET SUBGROUP "BIDI001";

SET SERVER MULTIDIR TO DB EAST_DB ID "QREPLADM" PASSWORD "xxxxxx";
SET SERVER MULTIDIR TO DB WEST_DB ID "QREPLADM" PASSWORD "xxxxxx";

SET MULTIDIR SCHEMA EAST_DB.EAST;
SET MULTIDIR SCHEMA WEST_DB.WEST;

SET CONNECTION SOURCE EAST_DB.EAST TARGET WEST_DB.WEST REPLQMAP E_TO_W_MAP;
SET CONNECTION SOURCE WEST_DB.WEST TARGET EAST_DB.EAST REPLQMAP W_TO_E_MAP;

set tables (EAST_DB.EAST.QREPLADM.BAL, WEST_DB.WEST.QREPLADM.BAL);
create qsub subtype b
from node WEST_DB.WEST
source has load phase N
target conflict rule C
       conflict action F
from node EAST_DB.EAST
source has load phase I
target conflict rule A
       conflict action I
       load type 0;

set tables (EAST_DB.EAST.QREPLADM.TRAN, WEST_DB.WEST.QREPLADM.TRAN);
create qsub subtype b
from node WEST_DB.WEST
source has load phase N
target conflict rule C
       conflict action F
from node EAST_DB.EAST
source has load phase I
target conflict rule A
       conflict action I
       load type 0;

set tables (EAST_DB.EAST.QREPLADM.OPTIONS, WEST_DB.WEST.QREPLADM.OPTIONS);
create qsub subtype b
from node WEST_DB.WEST
source has load phase N
target conflict rule C
       conflict action F
from node EAST_DB.EAST
source has load phase I
```

```
target conflict rule A
       conflict action I
       load type 0;
```

Create a password file

Since automatic loading of the tables was chosen, 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 table.

Example 5-33 creates the password file for user ID qrepladm on node EAST to access the source tables on WEST_DB, while Example 5-34 does the same on node WEST.

Example 5-33 Create the password file on EAST

```
cd ~qrepladm/apply
asnpwd init
asnpwd add alias WEST_DB id QREPLADM password "xxxxxx"
```

Example 5-34 Create the password file on WEST

```
cd ~qrepladm/apply
asnpwd init
asnpwd add alias EAST_DB id QREPLADM password "xxxxxx"
```

Update the CAPTURE_PATH and APPLY_PATH

The Replication Control Tables IBMQREP_CAPPARMS and IBMQREP_APPLYPARMS contain a column for CAPTURE_PATH and APPLY_PATH, respectively. Prior to starting the Q Capture and Q Apply programs, these columns need to be set to the values /db2_data/capture and /db2_data/apply on EAST (Jamesbay) and WEST (Azov), as shown in Example 5-35 and Example 5-36, respectively.

Example 5-35 Update the Capture and Apply paths on EAST

```
#############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 5-1 on page 425, Table 5-2 on page 426 and Table 5-3 on page 428
############################################################
##EAST_DB from reference A.7a in Table 5-1 on page 425
##/db2_data/capture from reference A.24 in Table 5-3 on page 428
##/db2_data/apply from reference A.25 in Table 5-3 on page 428
#############################################################

CONNECT TO EAST_DB ;
```

```
UPDATE EAST.IBMQREP_APPLYPARMS SET APPLY_PATH='/db2_data/apply',
PWDFILE='asnpwd.aut' ;

UPDATE EAST.IBMQREP_CAPPARMS SET CAPTURE_PATH='/db2_data/capture' ;

CONNECT RESET ;
```

Example 5-36 Update the Capture and Apply paths on WEST

```
#############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 5-1 on page 425, Table 5-2 on page 426 and Table 5-3 on page 428
############################################################
##WEST_DB from reference B.7a in Table 5-1 on page 425
##/db2_data/capture from reference B.24 in Table 5-3 on page 428
##/db2_data/apply from reference B.25 in Table 5-3 on page 428
#############################################################
CONNECT TO WEST_DB;

UPDATE WEST.IBMQREP_APPLYPARMS SET APPLY_PATH='/db2_data/apply',
PWDFILE='asnpwd.aut' ;

UPDATE WEST.IBMQREP_CAPPARMS SET CAPTURE_PATH='/db2_data/capture' ;

CONNECT RESET ;
```

Start the Q Capture and Q Apply programs

The only way to verify that the Q replication objects have been created properly is to start replication. The Q Capture and Q Apply programs can be started through the Replication Center or via the Unix command line, as shown in Example 5-37 through Example 5-40.

Example 5-37 Start Q Capture on EAST

```
############################################################
##EAST_DB from reference A.7a in Table 5-1 on page 425
##Capture schema EAST from reference A.22 in Table 5-3 on page 428
#############################################################
asnqcap capture_server=EAST_DB capture_schema=EAST
```

Example 5-38 Start Q Capture on WEST

```
############################################################
##WEST_DB from reference B.7a in Table 5-1 on page 425
##Capture schema WEST from reference B.22 in Table 5-3 on page 428
#############################################################
```

```
asnqcap capture_server=WEST_DB capture_schema=WEST
```

Example 5-39 Start Q Apply on EAST

```
#############################################################
##EAST_DB from reference A.7a in Table 5-1 on page 425
##Apply schema EAST from reference A.23 in Table 5-3 on page 428
##############################################################
asnqapp apply_server=EAST_DB apply_schema=EAST
```

Example 5-40 Start Q Apply on WEST

```
#############################################################
##WEST_DB from reference B.7a in Table 5-1 on page 425
##Apply schema WEST from reference B.23 in Table 5-3 on page 428
##############################################################
asnqapp apply_server=WEST_DB apply_schema=WEST
```

Verify successful implementation

Since all subscriptions were created to activate automatically and load automatically, the start of Q Capture and Q Apply programs loads the target tables on WEST, and marks all subscriptions as active.

> **Note:** To determine whether the load succeeded, use SQL[a] to verify that the number of rows in the source and target tables are identical or close to if the system is active. If the system is not active, **asntdiff** may be used to verify that the rows got loaded correctly.
>
> a. The SELECT COUNT(*) FROM table should have a minimum impact with an index available and UR isolation level.

Example 5-41 and Example 5-42 show the commands that display the status of the Q Capture and Q Apply programs on EAST. Similar commands may be executed on WEST to view the status of Q Capture and Q Apply. The state of the threads indicates that the Q Capture and Q Apply processes are operating correctly.

Example 5-41 Status of Q Capture on EAST

```
$ asnqccmd capture_server=EAST_DB capture_schema=EAST status
2004-11-10-13.27.18.535547 ASN0520I  "AsnQCcmd" : "EAST" : "Initial" : The
STATUS command response:"HoldLThread" thread is in the "is waiting" state.
2004-11-10-13.27.18.536952 ASN0520I  "AsnQCcmd" : "EAST" : "Initial" : The
STATUS command response:"AdminThread" thread is in the "is resting" state.
2004-11-10-13.27.18.537004 ASN0520I  "AsnQCcmd" : "EAST" : "Initial" : The
STATUS command response:"PruneThread" thread is in the "is resting" state.
```

```
2004-11-10-13.27.18.537047 ASN0520I  "AsnQCcmd" : "EAST" : "Initial" : The
STATUS command response:"WorkerThread" thread is in the "is resting" state.
```

Example 5-42 Status of Q Apply on EAST

```
$ asnqacmd apply_server=WEST_DB apply_schema=WEST status
2004-11-10-13.31.30.170914 ASN0520I  "AsnQAcmd" : "WEST" : "Initial" : The
STATUS command response:"HoldLThread" thread is in the "is waiting" state.
2004-11-10-13.31.30.172589 ASN0520I  "AsnQAcmd" : "WEST" : "Initial" : The
STATUS command response:"AdminThread" thread is in the "is resting" state.
2004-11-10-13.31.30.172648 ASN0520I  "AsnQAcmd" : "WEST" : "Initial" : The
STATUS command response:"BR00000" thread is in the "is doing work" state.
```

Example 5-43 displays the status of the subscriptions using SQL on EAST. Similar SQL may be executed on WEST to view the status of its subscriptions. The state of the subscriptions and queues shows `A` for active, indicating that Q replication is operating correctly in this environment.

Example 5-43 Status of subscriptions using SQL on EAST

```
-- CONNECT TO EAST_DB ;
-- SELECT substr(subname, 1, 20) as SUBNAME, STATE, HAS_LOADPHASE FROM
-- EAST.IBMQREP_SUBS ;
-- SELECT substr(subname, 1, 20) as SUBNAME, SUBGROUP, STATE, HAS_LOADPHASE,
-- LOAD_TYPE FROM EAST.IBMQREP_TARGETS ;
-- SELECT substr(sendq,1,20) as SENDQ, substr(recvq,1,20) as RECVQ, STATE FROM
-- EAST.IBMQREP_SENDQUEUES;
-- SELECT substr(sendq,1,20) as SENDQ, substr(recvq,1,20) as RECVQ, STATE FROM
-- EAST.IBMQREP_RECVQUEUES;
-- CONNECT RESET ;

CONNECT TO EAST_DB

   Database Connection Information

 Database server        = DB2/6000 8.2.0
 SQL authorization ID   = QREPLADM
 Local database alias   = EAST_DB


SELECT substr(subname, 1, 20) as SUBNAME, STATE, HAS_LOADPHASE FROM
EAST.IBMQREP_SUBS

SUBNAME              STATE HAS_LOADPHASE
-------------------- ----- -------------
TRAN0002             A     I
BAL0002              A     I
OPTIONS0002          A     I
```

```
  3 record(s) selected.


SELECT substr(subname, 1, 20) as SUBNAME, SUBGROUP, STATE, HAS_LOADPHASE,
LOAD_TYPE FROM EAST.IBMQRE
P_TARGETS

SUBNAME          SUBGROUP                 STATE HAS_LOADPHASE LOAD_TYPE
------------------------------------------ ----- ------------- ---------
TRAN0001         000001                   A     I                     0
BAL0001          000001                   A     I                     0
OPTIONS0001      000001                   A     I                     0

  3 record(s) selected.


SELECT substr(sendq,1,20) as SENDQ, substr(recvq,1,20) as RECVQ, STATE FROM
EAST.IBMQREP_SENDQUEUES

SENDQ                RECVQ                STATE
-------------------- -------------------- -----
EAST_TO_WEST_Q       EAST_TO_WEST_Q       A

  1 record(s) selected.


SELECT substr(sendq,1,20) as SENDQ, substr(recvq,1,20) as RECVQ, STATE FROM
EAST.IBMQREP_RECVQUEUES

SENDQ                RECVQ                STATE
-------------------- -------------------- -----
WEST_TO_EAST_Q       WEST_TO_EAST_Q       A

  1 record(s) selected.


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

5.5.8 Step 7: Set up the Q replication monitoring environment

We recommend setting up the monitoring environment with appropriate alert conditions for the bidirectional replication environment to ensure a stable operating environment,

Figure 5-87 lists the steps involved in setting up the monitoring environment.

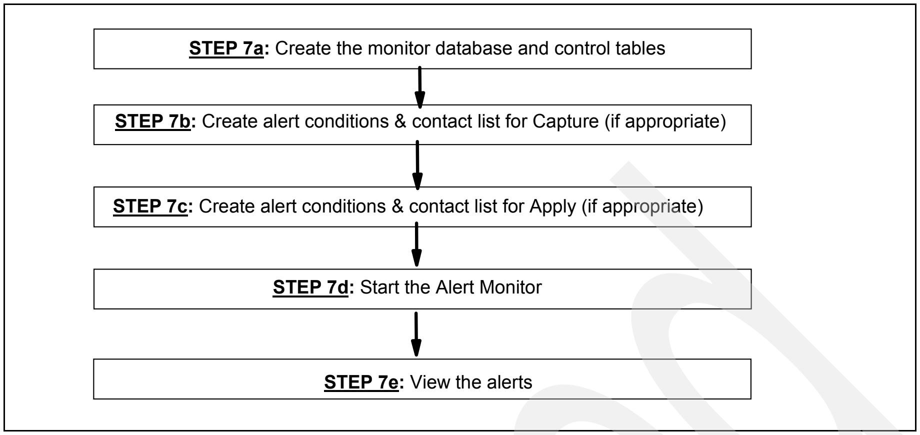


Figure 5-87 Steps to set up monitoring environment using GUI

Section 6.5.8, “Step 7: Set up monitoring environment using the GUI” on page 665; and 6.5.9, “Step 7: Set up monitoring environment using commands” on page 708, describe in detail the steps to set up a Q replication monitoring environment using a GUI and commands, respectively, for the peer-to-peer replication environment. The same series of steps applies to monitoring the bidirectional replication environment as well.

5.6 Failover considerations

Corona’s Q replication bidirectional topology involves two AIX servers, Jamesbay and Azov, where:

- Jamesbay is the primary server that is available for read and write applications.
- Azov is the non-dedicated secondary server that is available only for read applications against the replica.

The requirement is for the secondary server (Azov) to take over the update processing of applications in the event of failure of the primary server (Jamesbay). When the primary server is restored, the requirement is for the secondary server to revert back to read only (switchback). Until the primary server is restored, updates occurring at the secondary server are accumulated to be applied later to the primary server.

Attention: This section provides a very high-level overview of some of the considerations involved with failover processing associated with bidirectional replication. Depending upon the particular environment, the process involved in ensuring satisfactory failover processing can get quite complex. A full discussion of these considerations is beyond the scope of this book.

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 got written to the WebSphere MQ queue (item 1 in Figure 5-88), or messages in the WebSphere MQ queue that did not get transmitted to the secondary server (item 2 in Figure 5-88). These un-replicated changes should be considered to be data loss at the secondary server (at least until the primary server is restored). If there are messages in the receive queue on the secondary server that have not been drained (item 4 in Figure 5-88) 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.

Important: Identifying the data loss suffered can be critical for certain types of applications. Approaches for identifying the data loss suffered is beyond the scope of this book.

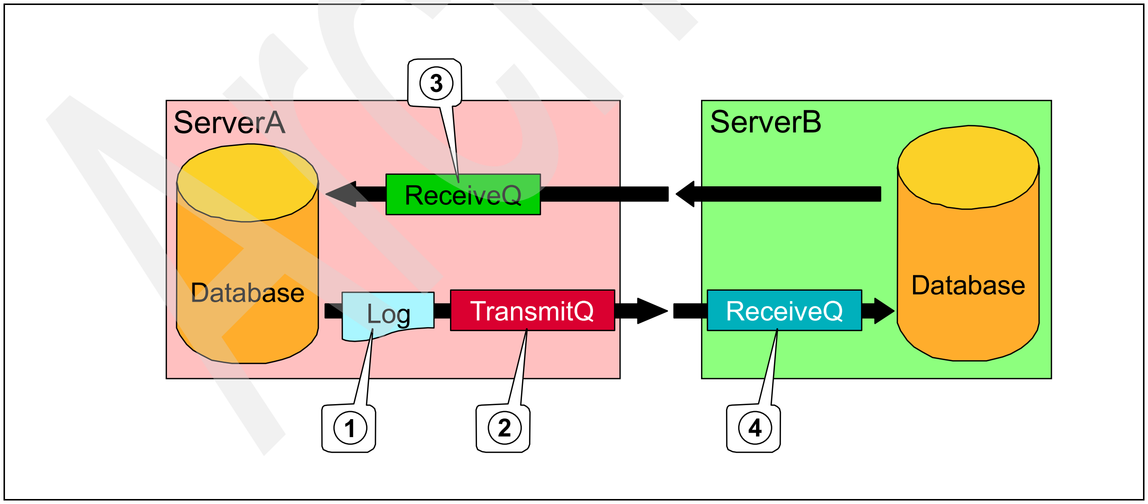


Figure 5-88 Potential sources of conflicts

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 5-88). 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 5-88).

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

To support the Corona 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.

> **Note:** The triggering event for failover is assumed to originate external to the Q replication environment, and enable updating applications on the secondary server. Therefore, 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. The current depth of the queue may be determined by executing the command shown in Example 5-44.

Example 5-44 Determine current depth of a queue

```
$ runmqsc QM_WEST
dis ql(QM_WEST) curdepth

1 : dis ql(QM_WEST) curdepth
AMQ8409: Display Queue details.
   QUEUE(QM_WEST)                   CURDEPTH(0)
```

- Upon the transmit queue reaching its capacity (MAXDEPTH), the Q Capture program will deactivate all subscriptions or Q Capture will shut down depending on ERROR_ACTION 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 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 the:

- 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, that may cause the transmit queue to fill up. Then Q Capture must be shut down before the transmit queue fills up. Example 5-45 shows how the states of the subscriptions can be determined so that Q Capture is turned off before subscriptions are deactivated.

Example 5-45 Check the subscriptions states

```
$ CONNECT TO WEST_DB

$ SELECT SUBNAME, STATE, HAS_LOADPHASE FROM WEST.IBMQREP_SUBS

SUBNAME           STATE HAS_LOADPHASE
-------------------------------------------
BAL0001             A     E
TRAN0001            A     E
OPTIONS0001         A     E

  3 record(s) selected.

$ CONNECT RESET
```

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 the updates on the DB2 log (after a Q Capture shutdown) 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.

5.7 Switchback 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 Corona's bidirectional topology this involves restoring Jamesbay as the primary server with updating applications, and the secondary server reverting back to a read-only environment against the replica. Switchback should involve minimum data loss.

Important: Identifying the data loss suffered can be critical for certain types of applications. Approaches for identifying the data loss suffered is beyond the scope of this book.

Attention: This section provides a very 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. A full discussion of these considerations is beyond the scope of this book.

The effort involved in restoring the normal operating environment depends upon the expected duration of the outage of the primary server.

- Short outage

  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 resynchronize and resume normal operations simply by restarting the failed server, and warm starting Q Capture on the primary server.

- Extended outage

  If the duration of the primary server outage was estimated to be so long that Q Capture was shut down to prevent the queues from filling up, then after restarting the failed server, adequate time must be provided so that the messages already stored in the WebSphere MQ queues can be processed by Q Apply on the primary server. This may be determined by starting Q Apply with the command shown in Example 5-46, which causes the Q Apply program to terminate after all receive queues are emptied once. After the messages have been processed, Q Capture can be restarted with the command shown in Example 5-47, which causes Q Capture to stop after reaching the end of the DB2 active log.

Example 5-46 Determine that Q Apply has finished processing messages

```
asnqapp apply_server=.... autostop=y .....
```

Example 5-47 Determine that Q Capture has finished reading the log

```
asnqcap capture_server=.... autostop=y .....
```

If, despite adequate planning, Q Capture on the secondary server deactivated the subscriptions, then a careful analysis is required to determine the appropriate recovery action, such as:

- If the Q Subscription was configured for automatic load, then reactivation of the subscription will cause the target table to be emptied and reloaded by the method shown in the subscription, as described in Figure 5-56 on page 498. For a large table this could take significant time and resources. The CAPSTART signal must be issued to the Q Capture program on the surviving side to designate source and target tables.
- If the Q Subscription was configured for a manual load, then a CAPSTART signal must be issued. Reconfirm that the tables are synchronized (using the TDIFF / TREPAIR utility, or a method of ones choice), and then a LOADDONE signal is issued to complete subscription activation.
- If appropriate, the existing subscriptions can be dropped and recreated.

Note: It is assumed that the primary server with WebSphere II is restarted successfully to the point-of-failure.

6

Peer-to-peer Q replication on AIX platforms

In this chapter we describe a step-by-step approach to implementing a peer-to-peer queue replication financial industry solution on AIX platforms.

The topics covered are:

- Business requirements
- Rationale for the peer-to-peer solution
- Environment configuration
- Step-by-step setup
- Monitoring considerations
- Failover considerations
- Switchback considerations

6.1 Introduction

Peer-to-peer 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 or more servers where updates are expected to occur at any of the servers as part of normal business activity. Scenarios that may be appropriate for a peer-to-peer replication solution include the following:

- Simultaneous updates to all the replicated tables are expected to occur, with a high potential for the same data to be updated on the different servers concurrently.
- Application users can connect to either server to accomplish their tasks, since all the servers are considered to be peers.

In this chapter we describe a high-availability business solution implementation cycle involving peer-to-peer 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 are documented as follows:

- Business requirements
- Rationale for choosing the peer-to-peer solution
- Environment configuration
- Step-by-step setup
- Failover considerations
- Switchback considerations

6.2 Business requirements

Our fictitious company Luna is a financial organization that must provide 24x7 access to its customers. Given the critical nature of the business, the implementation must support a high-availability scenario involving two servers. In the event of a failure of one of the servers, the remaining server must take over the workload of the failing site transparently and instantaneously. In addition, for scalability reasons Luna requires the two servers to act as peers—both for queries as well as updates, and present a single system image to applications. Therefore, in the event of simultaneous updates to the same data on the two servers, the more recent update must prevail.

These requirements may be summarized as follows:

- Provide a single system image to applications accessing the two servers—which means maintaining a peer relationship between the servers.

- When a server fails, there must be no outage perceived by the user, since the the remaining server must take over the workload of the failing server transparently and instantaneously.
- Replicated tables on both servers are peers, and updates may occur at either server—there is no concept of primary or secondary server.
- Update conflicts on the two servers must be resolved in favor of the most recent update.
- The failing server must be resynchronized with the active server after it is restored to service.

6.3 Rationale for the peer-to-peer solution

Table 2-1 on page 39 lists the evaluation criteria for choosing between unidirectional, bidirectional, and peer-to-peer replication topologies.

Since Luna's business requirement is support for a high availability peer environment, the choice is a peer-to-peer replication topology.

The peer-to-peer replication topology is appropriate for Luna for the following reasons:

- Totally transparent and instantaneous takeover in the event of failure of any server (in seconds)
- Expected conflicts[1] due to simultaneous updates to be resolved in favor of the most recent update
- Adequate computing resources available since either server is meant to pick up the workload of the failing server

Luna's requirement for a controlled switchback to the failing server from the active server is handled well by a peer-to-peer replication topology.

6.4 Environment configuration

Figure 6-1 shows the configuration used in the Luna peer-to-peer replication topology.

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

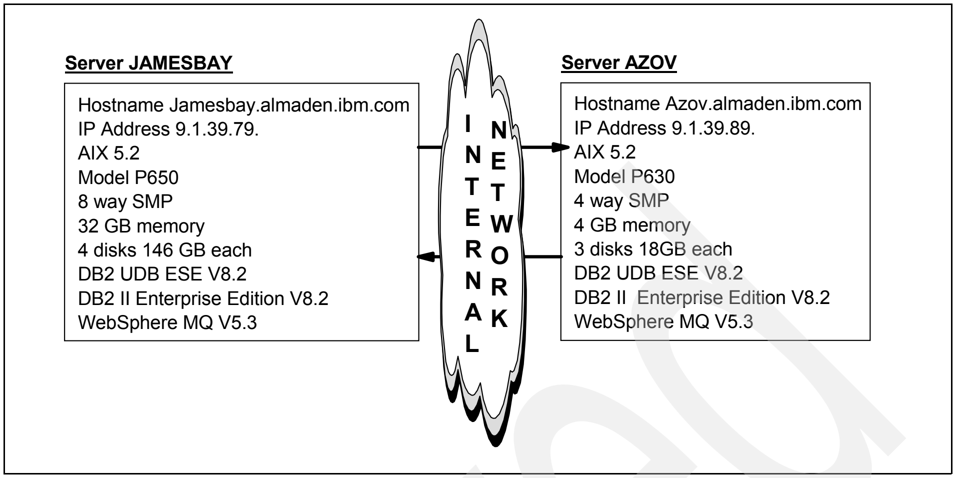


Figure 6-1 Peer-to-peer replication topology configuration for Luna

We installed a very simple set of three tables with referential integrity constraints defined between them. These tables included LOB data that were replicated as well. Full details of these tables are documented in Appendix , "Tables used in the Q replication scenarios on AIX" on page 782.

6.5 Step-by-step setup

In this section we document the step-by-step setup of the peer-to-peer replication topology in our fictitious company. Figure 6-2 lists the main steps involved in setting up the environment. Each of these steps is described in detail in the following subsections.

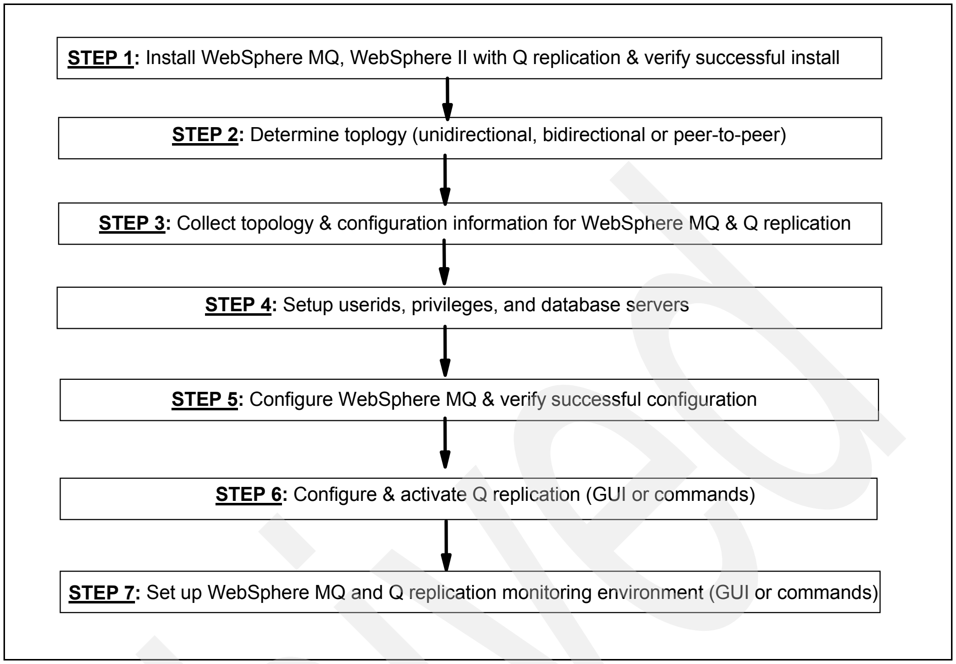


Figure 6-2 Overview of setup steps

6.5.1 Step 1: Install WebSphere MQ, WebSphere II with Q replication

Refer to *IBM WebSphere MQ for AIX Quick Beginnings Version 5.3*, GC34-6076-02; and *IBM DB2 Information Integrator Installation Guide for Linux, Unix and Windows Version 8.2,* GC18-7036-01, for details on prerequisites and steps for installing WebSphere MQ and WebSphere II with Q replication, respectively.

6.5.2 Step 2: Determine topology

We chose the peer-to-peer replication topology to address the Luna business requirement, as described in 6.3, "Rationale for the peer-to-peer solution" on page 545.

6.5.3 Step 3: Collect topology and configuration information

Implementing peer-to-peer replication is a complex task involving effective coordination of the configuration settings of the operating system, database management system, WebSphere MQ and WebSphere Information Integrator Q replication offerings.

Important: To ensure successful and error-free implementation, we strongly encourage systematic and careful planning that involves identifying and gathering all required information prior to commencing implementation.

Towards this end, we have developed a template that identifies all the information required to implement a peer-to-peer replication topology, and the cross-relationships between the information elements to ensure a smooth implementation.

Figure 6-3 provides a high-level overview of the various objects involved in implementing a peer-to-peer replication topology involving two servers, and serves as a reference for the host and DB2 system information template (Table 6-1 on page 549), WebSphere MQ configuration information template (Table 6-2 on page 550), Q replication configuration information template (Table 6-3 on page 552), and Replication Alert Monitor configuration information template (Table 6-4 on page 553). Each of the information elements for each server is associated with a reference identification such as A.7a or B.7b, where A represents one of the servers and B the other server. These reference IDs are then cross-referenced in the template itself as well as the WebSphere II Q replication configuration screens and scripts. For example, in Table 6-2 on page 550 on WebSphere MQ information, for the XMITQ parameter (reference ID A.18a for Server A) in the SendQ, we require reference ID A.13 (which is the name of the TransmitQ), and that value is QM_WEST.

Attention: In Figure 6-3, there appear to be two sets of transmission queues, and sender and receiver channels, on each server. However, there is only one set on each server, as can be deduced from the identical names. Figure 6-3 has the appearance of two sets so that the flow of data and messages between the two servers is easily understood.

Once all the information identified in the template had been collected, we can proceed with the actual implementation.

Note: The template shows certain parameter values that are required for Q replication (such as parameter DEFPSIST *must be* set to YES for the TransmitQ [reference ID A.13] and does *not* have a reference ID), while others can be customized for a particular environment (such as reference ID A.13d parameter MAXDEPTH *may be* set to any value).

We have collected all the identified information for the Luna peer-to-peer replication implemented and recorded them in the templates Table 6-1 on

page 549, Table 6-2 on page 550, Table 6-3 on page 552, and Table 6-4 on page 553. We are now ready to proceed with configuring the various resources.

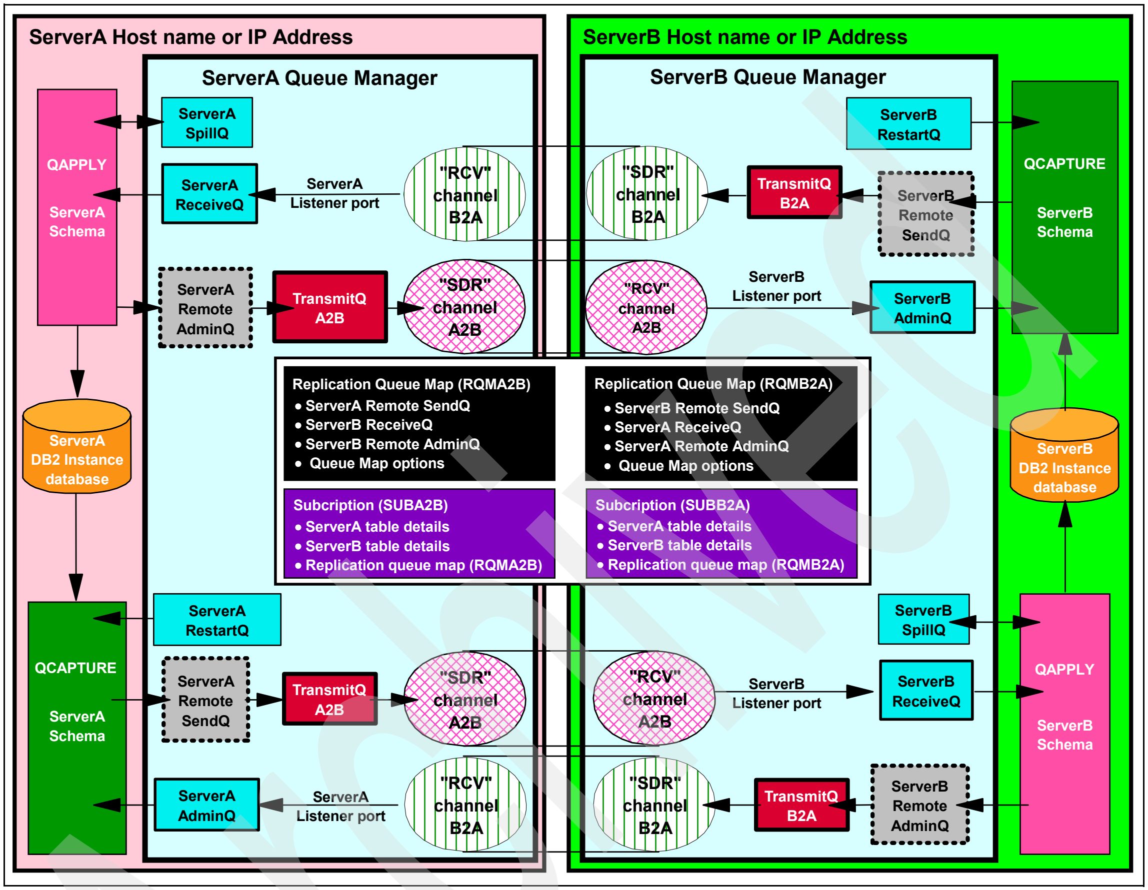


Figure 6-3 Peer-to-peer replication topology objects overview

Table 6-1 Host and DB2 system information

Description	**Server A**	**Server B**
Host name	(A.1) JAMESBAY	(B.1) AZOV
Host IP address	(A.2) 9.1.39.79	(B.2) 9.1.39.89
DB2 subsystem name or instance owner	(A.3) db2inst1	(B.3) db2inst1
DB2 group name (z/OS only)	(A.4)	(B.4)
DB2 location name (z/OS only)	(A.5)	(B.5)

Description	Server A	Server B
DB2 tcp port	(A.6) 50000	(B.6) 50000
Database server/alias information ▸ Database server/alias ▸ Q Capture user ID and password ▸ Q Apply user ID and password	 ▸ (A.7a) STOCK_E ▸ (A.7b) qrepladm/xxx ▸ (A.7c) qrepladm/xxx	 ▸ (B.7a) STOCK_W ▸ (B.7b) qrepladm/xxx ▸ (B.7c) qrepladm/xxx
User ID group (Unix only)	(A.8) qreplgrp	(B.8) qreplgrp
Other configuration user IDs requiring access	(A.9)	(B.9)
Logical database for control tables (z/OS only)	(A.10)	(B.10)

Table 6-2 WebSphere MQ information

Description	Server A	Server B
Queue Manager ▸ DEADQ (optional) ▸ MAXUMSGS (or use default) ▸ MAXMSGL (or use default)	(A.11) QM_PEER1 ▸ (A.11a) ▸ (A.11b) 10000 ▸ (A.11c) 4194304	(B.11) QM_PEER2 ▸ (B.11a) ▸ (B.11b) 10000 ▸ (B.11c) 4194304
Listener port	(A.12) 1421	(B.12) 1420
TransmitQ ▸ USAGE ▸ PUT ▸ GET ▸ DEFPSIST ▸ TRIGGER ▸ TRIGTYPE ▸ TRIGDATA ▸ INITQ ▸ MAXMSGL (or use default) ▸ MAXDEPTH (or use default)	(A.13) QM_PEER2 ▸ XMITQ ▸ ENABLED ▸ ENABLED ▸ YES ▸ YES ▸ FIRST ▸ (A.13a)=(A14) PEER1_TO_PEER2 ▸ (A.13b) SYSTEM.CHANNEL.INITQ ▸ (A.13c) 4194304 ▸ (A.13d) 5000	(B.13) QM_PEER1 ▸ XMITQ ▸ ENABLED ▸ ENABLED ▸ YES ▸ YES ▸ FIRST ▸ (B.13a)=(B14) PEER2_TO_PEER1 ▸ (B.13b) SYSTEM.CHANNEL.INITQ ▸ (B.13c) 4194304 ▸ (B.13d) 5000
SDR channel ▸ CHLTYPE ▸ TRPTYPE ▸ XMITQ ▸ CONNAME ▸ HBINT (or use default)	(A.14) PEER1_TO_PEER2 ▸ SDR ▸ (A.14a) TCP ▸ (A.14b)=(A.13) QM_PEER2 ▸ (A.14c) 9.1.39.89 (1420) ▸ (A.14d) 300	(B.14) PEER2_TO_PEER1 ▸ SDR ▸ (B.14a) TCP ▸ (B.14b)=(B.13) QM_PEER1 ▸ (B.14c) 9.1.39.79 (1421) ▸ (B.14d) 300

Description	**Server A**	**Server B**
RCV channel ► CHLTYPE ► TRPTYPE ► HBINT (or use default)	(A.15)=(B.14) PEER2_TO_PEER1 ► RCVR ► (A.15a) TCP ► (A.15b) 300	(B.15)=(A.14) PEER1_TO_PEER2 ► RCVR ► (B.15a) TCP ► (B.15b) 300
RestartQ ► PUT ► GET ► SHARE ► DEFSOPT ► DEFPSIST ► MAXMSGL (or use default) ► MAXDEPTH (or use default)	(A.16) PEER1_RESTARTQ ► ENABLED ► ENABLED ► YES ► SHARED ► YES ► (A.16a) 4194304 ► (A.16b) 5000	(B.16) PEER2_RESTARTQ ► ENABLED ► ENABLED ► YES ► SHARED ► YES ► (B.16a) 4194304 ► (B.16b) 5000
AdminQ ► PUT ► GET ► SHARE ► DEFSOPT ► DEFPSIST ► MAXMSGL (or use default) ► MAXDEPTH (or use default)	(A.17) PEER1_ADMINQ ► ENABLED ► ENABLED ► YES ► SHARED ► YES ► (A.17a) 4194304 ► (A.17b) 5000	(B.17) PEER2_ADMINQ ► ENABLED ► ENABLED ► YES ► SHARED ► YES ► (B.17a) 4194304 ► (B.17b) 5000
SendQ (remote) ► PUT ► DEFPSIST ► XMITQ ► RNAME ► RQMNAME	(A.18) PEER1_TO_PEER2_Q ► ENABLED ► YES ► (A.18a)=(A.13) QM_PEER2 ► (A.18b)=(B.19) PEER1_TO_PEER2_Q ► (A.18c)=(B.11) QM_PEER2	(B.18) PEER2_TO_PEER1_Q ► ENABLED ► YES ► (B.18a)=(B.13) QM_PEER1 ► (B.18b)=(A.19) PEER2_TO_PEER1_Q ► (B.18c)=(A.11) QM_PEER1
ReceiveQ (local) ► PUT ► GET ► SHARE ► DEFSOPT ► DEFPSIST ► INDXTYPE ► MSGDLVSQ ► MAXMSGL (or use default) ► MAXDEPTH (or use default)	(A.19) = (B.18b) PEER2_TO_PEER1_Q ► ENABLED ► ENABLED ► YES ► SHARED ► YES ► MSGID ► PRIORITY ► (A.19a) 4194304 ► (A.19b) 5000	(B.19) = (A.18b) PEER1_TO_PEER2_Q ► ENABLED ► ENABLED ► YES ► SHARED ► YES ► MSGID ► PRIORITY ► (B.19a) 4194304 ► (B.19b) 5000

Description	Server A	Server B
SpillQ ► DEFTYPE ► DEFSOPT ► MSGDLVSQ ► MAXMSGL (or use default) ► MAXDEPTH (or use default)	IBMQREP.SPILL.MODELQ ► PERMDYN ► SHARED ► FIFO ► (A.20a) 4194304 ► (A.20b) 5000	IBMQREP.SPILL.MODELQ ► PERMDYN ► SHARED ► FIFO ► (B.20a) 4194304 ► (B.20b) 5000
AdminQ (remote) ► PUT ► DEFPSIST ► XMITQ ► RNAME ► RQMNAME	(A.21) = (B.17) PEER2_ADMINQ ► ENABLED ► YES ► (A.21a)=(A.13) QM_PEER2 ► (A.21b)=(B.17) PEER2_ADMINQ ► (A.21c)=(B.11) QM_PEER2	(B.21) = (A.17) PEER1_ADMINQ ► ENABLED ► YES ► (B.21a)=(B.13) QM_PEER1 ► (B.21b)=(A.17) PEER1_ADMINQ ► (B.21c)=(A.11) QM_PEER1

Table 6-3 Q replication configuration information

Description	Server A	Server B
QCAPTURE Schema	(A.22) PEER1	(B.22) PEER2
QAPPLY Schema	(A.23) PEER1	(B.23) PEER2
Q Capture path (log files)	(A.24) /db2_data/capture	(B.24) /db2_data/capture
Q Apply path (log files)	(A.25) /db2_data/apply	(B.25) /db2_data/apply
Q Replication type (U/B/P)	(A.26) P	(B.26) P
Replication Queue Map name ► Max message length (KB) ► Error handling ► Num apply agents ► Memory buffer for Recvq (MB) ► Allow QCapture to send heartbeat ► Heartbeat interval (sec)	(A.27) P1_TO_P2_MAP ► (A.27a) 64 ► (A.27b) Stop Q Capture ► (A.27c) 16 ► (A.27d) 2 ► (A.27e) yes ► (A.27f) 60	(B.27) P2_TO_P1_MAP ► (B.27a) 64 ► (B.27b) Stop Q Capture ► (B.27c) 16 ► (B.27d) 2 ► (B.27e) yes ► (B.27f) 60
Q Subscriptions parameters ► Target table schema/creator ► Target table index schema/name ► Target tablespaces ► Check for conflicts setting ► Conflict resolution action ► Error response action ► Initial load option ► Source server for initial load ► Qsub start automatic	► (A.28a) same ► (A.28b) same ► (A.28c) each gets own ► (A.28d) all columns ► (A.28e) STOCK_E ► (A.28f) Stop receive queue ► (A.28g) Automatic ► (A.28h) STOCK_E ► (A.28i) yes	► (B.28a) same ► (B.28b) same ► (B.28c) each gets own ► (B.28d) all columns ► (B.28e) STOCK_W ► (B.28f) Stop receive queue ► (B.28g) Automatic ► (B.28h) STOCK_E ► (B.28i) yes
Q Subscriptions list	(A.29)	(B.29)

Table 6-4 Replication Alert Monitor configuration information

Description	Designated server
Host name	(M.1) KANAGA
Host IP Address	(M.2) 9.1.39.90
DB2 subsystem name or instance owner	(M.3) db2inst1
DB2 tcp port	(M.4) 50000
Monitor database server/alias information ▸ Database server/alias ▸ db2instance ▸ User ID/password ▸ Contact name ▸ Contact e-mail address ▸ Monitor qualifier ▸ Schema name	 ▸ (M.5a) MONDB ▸ (M.5b) DB2 ▸ (M.5c) qrepladm/xxx ▸ (M.5d) Qrepl Admin ▸ (M.5e) qrepladm@kanaga.almaden.ibm.com ▸ (M.5f) MON ▸ ASN
Monitor log path	(M.6) /db2_data/MON/log
DB2 group name (z/OS only)	(M.7)
DB2 location name (z/OS only)	(M.8)

6.5.4 Step 4: Set up user IDs, privileges, and database servers

The user IDs that run the Q replication and event publishing programs need authority to connect to servers, access or update tables, put and get from WebSphere MQ queues, and perform other operations such as write to directories. In addition, appropriate databases to be used in replication need to be created and cataloged for access by the Replication Center. The steps involved are described here.

User IDs and privileges

User IDs need to be defined for executing Q Capture (reference A.7b Table 6-1 on page 549) and Q Apply (reference A.7c Table 6-1 on page 549).

- Q Capture

  All user IDs that run a Q Capture program must have authorization to access the DB2® system catalog, access and update all Q Capture control tables, read the DB2 log, and run the Q Capture program packages. The following list summarizes the DB2 UDB requirements and operating system requirements for the UNIX® platform.

  – DBADM or SYSADM authority on the databases involved.

In a massively parallel processing (MPP) configuration, the user IDs must be able to connect to database partitions and read the password file.

- WRITE privilege on the directory that is specified by the CAPTURE_PATH parameter. The Q Capture program creates diagnostic files in this directory.
- MQM[2] privileges on the WebSphere MQ queues or necessary privileges to operate each queue and queue manager for Q Replication.

► Q Apply

All user IDs that run a Q Apply program must have authorization to access the DB2® system catalog, access and update targets, access and update the Q Apply control tables, run the Q Apply program packages, and read the Q Apply password file.

The following list summarizes the DB2 UDB requirements, operating system requirements, and WebSphere MQ requirements for the UNIX® platform.

- DBADM or SYSADM authority.
- SELECT privilege for the source tables if the Q Apply program will be used to load target tables.

  If the Q Apply program uses the LOAD from CURSOR option of the LOAD utility to load target tables, the Q Apply server must be a federated server, and you must create nicknames, server definitions, and user mappings on the Q Apply server. The user ID that is supplied in the user mappings must have privilege to read from nicknames on the federated Q Apply server and read from the source tables.
- WRITE privilege on the directory that is specified by the APPLY_PATH parameter. The Q Apply program creates diagnostic files in this directory.
- MQM privileges on the WebSphere MQ queues or necessary privileges to operate each queue and queue manager for Q Replication.

The following steps created the user IDs and assigned them the required system and database privileges:

1. We created a single user ID qrepladm (references A.7b, A.7c, B.7b, and B.7c) and assigned it to group qreplgrp (references A.8 and B.8), as shown in Example 6-1, on each of the servers Jamesbay and Azov.

Example 6-1 Create group qreplgrp and user ID qrepladm and associate them

```
############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 6-1 on page 549, and Table 6-2 on page 550
```

[2] The MQM privilege in WebSphere MQ is equivalent to having the SYSADM privilege in DB2.

```
############################################################
##QREPLGRP from reference A.8 and B.8 in Table 6-2 on page 550
##QREPLADM from reference A.7b and B.7b in Table 6-1 on page 549
#############################################################
mkgroup qreplgrp
mkuser qrepladm
chuser groups=qreplgrp, qrepladm
```

2. We created directories /db2_data/capture (reference A.24 and B.24) and /db2_data/apply (reference A.25 and B.25) where the Q Capture and Q Apply programs will operate and need to have write permissions, as shown in Example 6-2, on each of the servers Jamesbay and Azov.

   > **Note:** Since we planned to automatically load the tables for our subscriptions, we needed to ensure that the DB2 instance owner db2inst1 (reference A.3 and B.3) has full permissions in the APPLY_PATH directory so that it can create subdirectories for all the data files.

   We therefore chose to allow public read/write access to these directories.

Example 6-2 Create Q Capture and Q Apply directories and give write permission to it

```
#############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 6-3 on page 552
############################################################
##/db2_data/capture from reference A.24 and B.24 in Table 6-3 on page 552
##/db2_data/apply from reference A.25 and B.25 in Table 6-3 on page 552
#############################################################
mkdir /db2_data/capture
mkdir /db2_data/apply
chmod 777 /db2_data/capture
chmod 777 /db2_data/apply
```

3. We then granted user ID qrepladm on each server DBADM and SYSCTRL authority on the respective database—on server Jamesbay, it is on STOCK_E, while on server Azov, it is on STOCK_W, as shown in Example 6-3 and Example 6-4, respectively.

   > **Note:** SYSADM privilege may be granted instead of DBADM and SYSCTRL.

Example 6-3 Grant DBADM and SYSCTRL privilege on STOCK_E to qrepladm

```
#############################################################
```

```
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 6-1 on page 549
############################################################
##STOCK_E from reference A.7a in Table 6-1 on page 549
##QREPLADM from reference A.7b and A.7c in Table 6-1 on page 549
##QREPLGRP from reference A.8 in Table 6-1 on page 549
#############################################################
db2 connect to STOCK_E
db2 grant DBADM on DATABASE to USER qrepladm
db2 update dbm cfg using SYSCTRL_GROUP qreplgrp
```

Example 6-4 Grant DBADM and SYSCTRL privilege on STOCK_W to qrepladm

```
#############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 6-1 on page 549
############################################################
##STOCK_W from reference B.7a in Table 6-1 on page 549
##QREPLADM from reference B.7b and A.7c in Table 6-1 on page 549
##QREPLGRP from reference B.8 in Table 6-1 on page 549
#############################################################
db2 connect to STOCK_W
db2 grant DBADM on DATABASE to USER qrepladm
db2 update dbm cfg using SYSCTRL_GROUP qreplgrp
```

4. The user IDs that will be used to run the Q replication Apply and Capture programs need to have adequate MQ authorities for all WebSphere MQ objects. This may be accomplished by either:

   – Adding the user IDs to the mqm group.

     We granted mqm privilege to qrepladm on each of the servers Jamesbay and Azov, as shown in Example 6-5, to allow the Q Capture and Q Apply programs to operate the WebSphere MQ queues. This grants the IDs full administrative authority over all MQ objects.

Example 6-5 Add user ID qrepladm to mqm group

```
#############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 6-1 on page 549
############################################################
##QREPLGRP from reference A.8 and B.8 in Table 6-1 on page 549
##QREPLADM from reference A.7b and B.7b in Table 6-1 on page 549
#############################################################
chuser groups=qreplgrp,mqm qrepladm
```

– Granting individual authorities to the specific objects required

For environments where granting the mqm privilege may not be acceptable, Example 6-6 and Example 6-7 provide details on granting individual privileges instead of mqm privileges.

> **Note:** This assumes that the WebSphere MQ objects referenced in Example 6-6 and Example 6-7 already exist. In our case, these WebSphere MQ objects are defined in 6.5.5, “Step 5: Configure WebSphere MQ” on page 561.

For full details on these privileges refer to Chapter 10 of the *WebSphere MQ System Administration Guide*, SC34-6068-02.

Example 6-6 Grant individual privileges on node PEER1 on server Jamesbay

```
#############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 6-1 on page 549, and Table 6-2 on page 550
############################################################
##QM_PEER1 from reference A.11 in Table 6-2 on page 550
##QREPLADM from reference A.7b in Table 6-1 on page 549
##PEER2_TO_PEER1_Q from reference A.19 in Table 6-2 on page 550
##PEER1_TO_PEER2_Q from reference A.18 in Table 6-2 on page 550
##PEER1_ADMINQ from reference A.17 in Table 6-2 on page 550
##PEER1_RESTARTQ from reference A.16 in Table 6-2 on page 550
##QM_PEER2 from reference B.11 in Table 6-2 on page 550
##PEER2_ADMINQ from reference B.17 in Table 6-2 on page 550
##QREPLGRP from reference A.8 in Table 6-1 on page 549
#############################################################
setmqaut -m QM_PEER1 -t qmgr -p qrepladm +connect +inq
setmqaut -m QM_PEER1 -t queue -n PEER2_TO_PEER1_Q -p qrepladm +inq +get +browse
setmqaut -m QM_PEER1 -t queue -n IBMQREP.SPILL.MODELQ -p qrepladm +inq +put
setmqaut -m QM_PEER1 -t queue -n PEER1_TO_PEER2_Q -p qrepladm +inq +put
setmqaut -m QM_PEER1 -t queue -n PEER1_ADMINQ -p qrepladm +inq +get +browse
+put
setmqaut -m QM_PEER1 -t queue -n PEER1_RESTARTQ -p qrepladm +inq +get +put
+browse
setmqaut -m QM_PEER1 -t queue -n QM_PEER2 -p qrepladm +put +inq
setmqaut -m QM_PEER1 -t queue -n PEER2_ADMINQ -p qrepladm +inq +put
setmqaut -m QM_PEER1 -t queue -n IBMQREP.SPILL.MODELQ -g qreplgrp +allmqi +dlt
+chg +dsp +clr
```

Example 6-7 Grant individual privileges on PEER2 on server Azov

```
#############################################################
##The variables in the commands are shown unbold, and
```

```
##these values are substituted from the templates described
##in Table 6-1 on page 549, and Table 6-2 on page 550
############################################################
##QM_PEER2 from reference B.11 in Table 6-2 on page 550
##QREPLADM from reference B.7b in Table 6-1 on page 549
##PEER1_TO_PEER2_Q from reference A.18 in Table 6-2 on page 550
##PEER2_TO_PEER1_Q from reference A.19 in Table 6-2 on page 550
##PEER2_ADMINQ from reference B.17 in Table 6-2 on page 550
##PEER2_RESTARTQ from reference B.16 in Table 6-2 on page 550
##QM_PEER1 from reference A.11 in Table 6-2 on page 550
##PEER1_ADMINQ from reference A.17 in Table 6-2 on page 550
##QREPLGRP from reference B.8 in Table 6-1 on page 549
#############################################################
setmqaut -m QM_PEER2 -t qmgr -p qrepladm +connect +inq
setmqaut -m QM_PEER2 -t queue -n PEER1_TO_PEER2_Q -p qrepladm +inq +get +browse
setmqaut -m QM_PEER2 -t queue -n IBMQREP.SPILL.MODELQ -p qrepladm +inq +put
setmqaut -m QM_PEER2 -t queue -n PEER2_TO_PEER1_Q -p qrepladm +inq +put
setmqaut -m QM_PEER2 -t queue -n PEER2_ADMINQ -p qrepladm +inq +get +browse
+put
setmqaut -m QM_PEER2 -t queue -n PEER2_RESTARTQ -p qrepladm +inq +get +put
+browse
setmqaut -m QM_PEER2 -t queue -n QM_PEER1 -p qrepladm +put +inq
setmqaut -m QM_PEER2 -t queue -n PEER1_ADMINQ -p qrepladm +inq +put
setmqaut -m QM_PEER2 -t queue -n IBMQREP.SPILL.MODELQ -g qreplgrp +allmqi +dlt
+chg +dsp +clr
```

Database servers

The following actions need to be performed to set up the database servers:

1. In the AIX environment, we need to set the EXTSHM environment variable in the DB2 instance owner environment as shown in Example 6-8.

Example 6-8 Set EXTSHM environment variable

```
EXTSHM=ON; export EXTSHM
db2set DB2ENVLIST=EXTSHM
```

2. We assume that the databases used for servers (STOCK_E reference A.7a and STOCK_W reference B.7a) of peer-to-peer Q replication already exist on servers Jamesbay and Azov, respectively. We also assume that these databases are recoverable—they have the database configuration parameter logretain set to recovery. If not, set it for each database, as shown in Example 6-9.

Example 6-9 Setting logretain to recovery

```
db2 update db cfg using logretain recovery;
```

After executing this command, the database needs to be backed up using the command similar to that shown in Example 6-10.

Example 6-10 Backup database

```
#############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 6-1 on page 549
############################################################
##STOCK_E from reference A.7a in Table 6-1 on page 549
#############################################################
db2 backup db STOCK_E to /Backup
```

3. The STOCK_E on server Jamesbay and STOCK_W on server Azov must be cataloged on server Azov and server Jamesbay, respectively, as shown in Example 6-11 and Example 6-12.

Example 6-11 Catalog STOCK_E on server Azov

```
#############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 6-1 on page 549
############################################################
##PEER1 is a madeup name - not referenced in the template
##9.1.39.79 from reference A.2 in Table 6-1 on page 549
##50000 from reference B.6 in Table 6-1 on page 549
##STOCK_E from reference A.7a in Table 6-1 on page 549
#############################################################
db2 catalog tcpip node PEER1 remote 9.1.39.79 server 50000
db2 catalog database STOCK_E at node PEER1
db2 terminate;
```

Example 6-12 Catalog STOCK_W on server Jamesbay

```
#############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 6-1 on page 549
############################################################
##PEER2 is a madeup name - not referenced in the template
##9.1.39.89 from reference B.2 in Table 6-1 on page 549
##50000 from reference A.6 in Table 6-1 on page 549
##STOCK_W from reference B.7a in Table 6-1 on page 549
#############################################################
db2 catalog tcpip node PEER2 remote 9.1.39.89 server 50000
db2 catalog database STOCK_W at node PEER2
```

```
db2 terminate;
```

4. We tested the connectivity to STOCK_W and STOCK_E from servers Jamesbay and Azov, respectively, as shown in Example 6-13 and Example 6-14.

Example 6-13 Test connectivity to STOCK_W from server Jamesbay

```
#############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 6-1 on page 549
############################################################
##STOCK_W from reference B.7a in Table 6-1 on page 549
##QREPLADM from reference B.7b in Table 6-1 on page 549
#############################################################
db2 connect to STOCK_W user qrepladm using xxx
db2 terminate;
```

Example 6-14 Test connectivity to STOCK_E from server Azov

```
#############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 6-1 on page 549
############################################################
##STOCK_E from reference A.7a in Table 6-1 on page 549
##QREPLADM from reference A.7b in Table 6-1 on page 549
#############################################################
db2 connect to STOCK_E user qrepladm using xxx
db2 terminate;
```

Time synchronization

When using peer-to-peer replication it is critical to keep all system times synchronized. The **setclock** AIX command can be used to synchronize system clocks as follows:

```
$ setclock [ TimeServer ]
```

This command synchronizes the system clock on the server it runs on with the system clock of the server specified in the [TimeServer].

6.5.5 Step 5: Configure WebSphere MQ

Several WebSphere MQ objects need to be created in support of Q replication, and configured to suit an organization's particular workload. The WebSphere MQ objects are:

- Queue Manager
- Channels
- Local queues
- Remote queues

There is a hierarchical relationship between these objects—the Queue Manager is created at the highest level, and all objects are managed by the Queue Manager.

To support a typical Q replication bidirectional setup, the following objects must be defined on each system:

- Queue Manager
- Sender channel
- Receiver channel
- Transmit queue (local)
- Capture administration queue (local)
- Capture restart queue (local)
- Capture send queue (remote)
- Apply receive queue (local)
- Apply spill queue (local model)
- Apply administration queue (remote)

Note: *Local* refers to the fact that the queue is defined as a local queue on the system, while *remote* refers to the fact that the queue is defined as a remote queue.

After configuring the WebSphere MQ objects, we verified its success by testing the put and get messages from the local and remote queues.

Configure WebSphere MQ objects

Example 6-15 is the script for creating all the WebSphere MQ objects necessary for peer-to-peer replication on node PEER1 (Jamesbay), while Example 6-16 on page 565 is the script for creating the objects on node PEER2 (Azov).

Important: The scripts should be run by a member of the mqm group.

Note: The values for the names of the WebSphere MQ objects and the configuration parameters are substituted from the templates listed in Table 6-1 on page 549, Table 6-2 on page 550, and Table 6-3 on page 552. The specific references are listed in the script.

Example 6-15 Create necessary MQ objects for peer-to-peer replication on PEER1

```
#################################################################
# Set up the PEER1 server
#################################################################
#
# This section ensures that any existing MQ listener and Queue
# Manager are stopped and deleted before attempting to recreate
# them in the secions below.
#
# Uncomment the following lines if you wish to run this script
# to recreate all objects, after an initial run

#endmqm QM_PEER1
#sleep 30
#endmqlsr
#sleep 20
#dltmqm QM_PEER1

#######################################################################
# 1) Create the Queue Manager on PEER1 (do not make it the default QM)
#######################################################################
##Queue Manager reference A.11
crtmqm QM_PEER1

#######################################################################
# 2) Start the Queue Manager on PEER1
#######################################################################
##Queue Manager reference A.11
strmqm QM_PEER1

#######################################################################
# 3) Start a WebSphere MQ listener
#######################################################################
##WebSphere MQ listener port reference A.12
runmqlsr -m QM_PEER1 -t tcp -p 1421 &

#######################################################################
# 4) Create the queue/channel objects
#######################################################################
##Local AdminQ reference A.17
##Local AdminQ MAXMSGL reference A.17a
```

```
##Local AdminQ MAXDEPTH reference A.17b

runmqsc QM_PEER1 <<!

DEFINE QLOCAL(PEER1_ADMINQ) +
REPLACE +
DESCR('LOCAL DEFN OF ADMINQ FOR PEER1 CAPTURE') +
PUT(ENABLED) +
GET(ENABLED) +
DEFPSIST(YES) +
MAXMSGL(4194403) +
MAXDEPTH(5000)
##
##Local RestartQ reference A.16
##Local RestartQ MAXMSGL reference A.16a
##Local RestartQ MAXDEPTH reference A.16b

DEFINE QLOCAL(PEER1_RESTARTQ) +
REPLACE +
DESCR('LOCAL DEFN OF RESTARTQ FOR PEER1 CAPTURE') +
PUT(ENABLED) +
GET(ENABLED) +
DEFPSIST(YES) +
MAXMSGL(4194403) +
MAXDEPTH(5000)
##
##Local SpillQ reference A.20
##Local SpillQ MAXMSGL reference A.20a
##Local SpillQ MAXDEPTH reference A.20b


DEFINE QMODEL(IBMQREP.SPILL.MODELQ) +
REPLACE +
DEFSOPT(SHARED) +
MAXDEPTH(500000) +
MSGDLVSQ(FIFO) +
DEFTYPE(PERMDYN) +
MAXMSGL(100000)
##
##Remote SendQ QREMOTE reference A.18
##Remote SendQ XMITQ reference A.18a
##Remote SendQ RNAME reference A.18b
##Remote SendQ RQMNAME reference A.18c

DEFINE QREMOTE(PEER1_TO_PEER2_Q) +
REPLACE +
DESCR('REMOTE DEFN OF SEND QUEUE FROM PEER1 TO PEER2') +
PUT(ENABLED) +
XMITQ(QM_PEER2)  +
```

```
RNAME(PEER1_TO_PEER2_Q) +
RQMNAME(QM_PEER2) +
DEFPSIST(YES)
##
##Local ReceiveQ QLOCAL reference A.19
##Local ReceiveQ MAXMSGL reference A.19a
##Local ReceiveQ MAXDEPTH reference A.19b

DEFINE QLOCAL(PEER2_TO_PEER1_Q) +
REPLACE +
DESCR('LOCAL RECEIVE QUEUE FOR PEER1 FROM PEER2') +
GET(ENABLED) +
SHARE +
DEFPSIST(YES) +
MSGDLVSQ(PRIORITY) +
MAXMSGL(4194403) +
MAXDEPTH(5000)
##
##Remote AdminQ QREMOTE reference A.21
##Remote AdminQ XMITQ reference A.21a
##Remote AdminQ RNAME reference A.21b
##Remote AdminQ RQMNAME reference A.21c

DEFINE QREMOTE(PEER2_ADMINQ) +
REPLACE +
DESCR('REMOTE DEFN OF ADMINQ FOR PEER2 CAPTURE') +
PUT(ENABLED) +
XMITQ(QM_PEER2) +
RNAME(PEER2_ADMINQ) +
RQMNAME(QM_PEER2) +
DEFPSIST(YES)
##
##Local TransmitQ QLOCAL reference A.13
##Local TransmitQ TRIGDATA reference A.13a
##Local TransmitQ INITQ reference A.13b
##Local TransmitQ MAXMSGL reference A.13c
##Local TransmitQ MAXDEPTH reference A.13d

DEFINE QLOCAL(QM_PEER2) +
REPLACE +
DESCR('TRANSMISSION QUEUE TO PEER2') +
USAGE(XMITQ) +
DEFPSIST(YES) +
TRIGGER +
TRIGTYPE(FIRST) +
TRIGDATA(PEER1_TO_PEER2) +
INITQ(SYSTEM.CHANNEL.INITQ) +
MAXMSGL(4194403) +
MAXDEPTH(5000)
```

```
##
##Local SDR channel CHANNEL reference A.14
##Local SDR channel TRPTYPE reference A.14a
##Local SDR channel XMITQ reference A.14b
##Local SDR channel CONNAME reference A.14c
##Local SDR channel HBINT reference A.14d

DEFINE CHANNEL(PEER1_TO_PEER2) +
CHLTYPE(SDR) +
REPLACE +
TRPTYPE(TCP) +
DESCR('SENDER CHANNEL TO PEER2') +
XMITQ(QM_PEER2) +
CONNAME('9.1.39.89(1420)') +
HBINT(300)
##
##Local RCV channel CHANNEL reference A.15
##Local RCV channel TRPTYPE reference A.15a
##Local RCV channel HBINT reference A.15b

DEFINE CHANNEL(PEER2_TO_PEER1) +
CHLTYPE(RCVR) +
REPLACE +
TRPTYPE(TCP) +
DESCR('RECEIVER CHANNEL FROM PEER2') +
HBINT(300)
##
##MAXUMSGS reference A.11b
##MAXMSGL reference A.11c

ALTER QMGR +
MAXUMSGS(10000) +
MAXMSGL(4194403)

!

# END OF SCRIPT
```

Note: The same listener port can be used on both host servers, but it may be easier to debug if one uses a different port on each.

Example 6-16 Create necessary MQ objects for peer-to-peer replication on PEER2

```
#################################################################
# Set up the PEER2 server
#################################################################
#
# This section ensures that any existing MQ listener and Queue
```

```
# Manager are stopped and deleted before attempting to recreate
# them in the secions below.
#
# Uncomment the following lines if you wish to run this script
# to recreate all objects, after an initial run

#endmqm QM_PEER2
#sleep 30
#endmqlsr
#sleep 20
#dltmqm QM_PEER2

#################################################################
# 1) Create Queue Manager on PEER2 (do not make it the default QM)
#################################################################
##Queue Manager reference B.11
crtmqm QM_PEER2

#################################################################
# 2) Start the Queue Manager on PEER2
#################################################################
##Queue Manager reference B.11
strmqm QM_PEER2

#################################################################
# 3) Start a WebSphere MQ listener
#################################################################
##WebSphere MQ listener port reference B.12
runmqlsr -m QM_PEER2 -t tcp -p 1420 &

#################################################################
# 4) Create the queue/channel objects
#################################################################
##Local AdminQ reference B.17
##Local AdminQ MAXMSGL reference B.17a
##Local AdminQ MAXDEPTH reference B.17b

runmqsc QM_PEER2 <<!

DEFINE QLOCAL(PEER2_ADMINQ) +
REPLACE +
DESCR('LOCAL DEFN OF ADMINQ FOR PEER2 CAPTURE') +
PUT(ENABLED) +
GET(ENABLED) +
DEFPSIST(YES) +
MAXMSGL(4194403) +
MAXDEPTH(5000)
##
##Local RestartQ reference B.16
```

```
##Local RestartQ MAXMSGL reference B.16a
##Local RestartQ MAXDEPTH reference B.16b

DEFINE QLOCAL(PEER2_RESTARTQ) +
REPLACE +
DESCR('LOCAL DEFN OF RESTARTQ FOR PEER2 CAPTURE') +
PUT(ENABLED) +
GET(ENABLED) +
DEFPSIST(YES) +
MAXMSGL(4194403) +
MAXDEPTH(5000)
##
##Local SpillQ reference B.20
##Local SpillQ MAXMSGL reference B.20a
##Local SpillQ MAXDEPTH reference B.20b

DEFINE QMODEL(IBMQREP.SPILL.MODELQ) +
REPLACE +
DEFSOPT(SHARED) +
MAXDEPTH(500000) +
MSGDLVSQ(FIFO) +
DEFTYPE(PERMDYN) +
MAXMSGL(100000)
##
##Remote SendQ QREMOTE reference B.18
##Remote SendQ XMITQ reference B.18a
##Remote SendQ RNAME reference B.18b
##Remote SendQ RQMNAME reference B.18c

DEFINE QREMOTE(PEER2_TO_PEER1_Q) +
REPLACE +
DESCR('REMOTE DEFN OF SEND QUEUE FROM PEER2 TO PEER1') +
PUT(ENABLED) +
XMITQ(QM_PEER1)  +
RNAME(PEER2_TO_PEER1_Q) +
RQMNAME(QM_PEER1) +
DEFPSIST(YES)
##
##Local ReceiveQ QLOCAL reference B.19
##Local ReceiveQ MAXMSGL reference B.19a
##Local ReceiveQ MAXDEPTH reference B.19b

DEFINE QLOCAL(PEER1_TO_PEER2_Q) +
REPLACE +
DESCR('LOCAL RECEIVE QUEUE FOR PEER2 FROM PEER1') +
GET(ENABLED) +
SHARE +
DEFPSIST(YES) +
MSGDLVSQ(PRIORITY) +
```

```
MAXMSGL(4194403) +
MAXDEPTH(5000)
##
##Remote AdminQ QREMOTE reference B.21
##Remote AdminQ XMITQ reference B.21a
##Remote AdminQ RNAME reference B.21b
##Remote AdminQ RQMNAME reference B.21c

DEFINE QREMOTE(PEER1_ADMINQ) +
REPLACE +
DESCR('REMOTE DEFN OF ADMINQ FOR PEER1 CAPTURE') +
PUT(ENABLED) +
XMITQ(QM_PEER1) +
RNAME(PEER1_ADMINQ) +
RQMNAME(QM_PEER1) +
DEFPSIST(YES)
##
##Local TransmitQ QLOCAL reference B.13
##Local TransmitQ TRIGDATA reference B.13a
##Local TransmitQ INITQ reference B.13b
##Local TransmitQ MAXMSGL reference B.13c
##Local TransmitQ MAXDEPTH reference B.13d

DEFINE QLOCAL(QM_PEER1) +
REPLACE +
DESCR('TRANSMISSION QUEUE TO PEER1') +
USAGE(XMITQ) +
DEFPSIST(YES) +
TRIGGER +
TRIGTYPE(FIRST) +
TRIGDATA(PEER2_TO_PEER1) +
INITQ(SYSTEM.CHANNEL.INITQ) +
MAXMSGL(4194403) +
MAXDEPTH(5000)
##
##Local SDR channel CHANNEL reference B.14
##Local SDR channel TRPTYPE reference B.14a
##Local SDR channel XMITQ reference B.14b
##Local SDR channel CONNAME reference B.14c
##Local SDR channel HBINT reference B.14d

DEFINE CHANNEL(PEER2_TO_PEER1) +
CHLTYPE(SDR) +
REPLACE +
TRPTYPE(TCP) +
DESCR('SENDER CHANNEL TO PEER1') +
XMITQ(QM_PEER1) +
CONNAME('9.1.39.79(1421)') +
HBINT(300) +
```

```
DISCINT(0)
##
##Local RCV channel CHANNEL reference B.15
##Local RCV channel TRPTYPE reference B.15a
##Local RCV channel HBINT reference B.15b

DEFINE CHANNEL(PEER1_TO_PEER2) +
CHLTYPE(RCVR) +
REPLACE +
TRPTYPE(TCP) +
DESCR('RECEIVER CHANNEL FROM PEER1') +
HBINT(300)
##
##MAXUMSGS reference B.11b
##MAXMSGL reference B.11c

ALTER QMGR +
MAXUMSGS(10000) +
MAXMSGL(4194403)

!

# END OF SCRIPT
```

Note: The same listener port can be used on both host servers, but it may be easier to debug if one uses a different port on each.

Verify successful WebSphere MQ configuration

After performing the initial configuration, the channels must be started on each Queue Manager, and then messages may be inserted and retrieved from local and remote queues to verify successful configuration.

Example 6-17 and Example 6-18 show scripts that start the sender channel on node PEER1 and PEER2, respectively.

Example 6-17 Start the sender channel on PEER1

```
##################################################################
# Run on PEER1
##################################################################
#
# Start the sender channel
#
runmqchl -c PEER1_TO_PEER2 -m QM_PEER1 &
```

Example 6-18 Start the sender channel on PEER2

```
#################################################################
# Run on PEER2
#################################################################
#
# Start the sender channel
#
runmqchl -c PEER2_TO_PEER1 -m QM_PEER2 &
```

After the channels have been started, some of the several queue pairs may be tested to ensure that communications are working properly. We tested local as well as remote queues, as follows.

Testing local queues

Example 6-19 lists a local queue on the PEER1 and PEER2 nodes.

Example 6-19 Local queues

```
####################################################
Simple Queues:           amqsput/amqsget
####################################################
PEER2_RESTARTQ           QM_PEER2
PEER1_RESTARTQ           QM_PEER1
```

For each of these queues, we inserted a message (using /usr/mqm/samp/bin/amqsput) and read back the message (using /usr/mqm/samp/bin/amqsget) on the same node, as shown in Example 6-20. Testing local queues ensures that they are set up properly, but does not test the connectivity between the queue managers—for that we need to test the remote queues.

Example 6-20 Test put and get from a local queue

```
####################################################
# Execute the following command to put a message into the queue
####################################################
/usr/mqm/samp/bin/amqsput PEER2_RESTARTQ QM_PEER2
Sample AMQSPUT0 start
target queue is PEER2_RESTARTQ
This is a test message to a local queue

Sample AMQSPUT0 end
#
####################################################
# Execute the following command to get the message from the queue.

# The program should display the text one entered above then pause
# for a few seconds waiting for more messages.
####################################################
```

```
#
/usr/mqm/samp/bin/amqsget PEER2_RESTARTQ QM_PEER2
Sample AMQSGET0 start
message <This is a test message to a local queue>
no more messages
Sample AMQSGET0 end
```

Testing remote queues

Example 6-21 lists remote queues on the PEER1 and PEER2 nodes. The queue manager listed next to each queue in Example 6-21 is the LOCAL side.

Example 6-21 Remote queues

```
####################################################
Queue Pairs:       amqsput side       amqsget side
####################################################
#
PEER2_ADMINQ       QM_PEER1           QM_PEER2
PEER1_TO_PEER2_Q   QM_PEER1           QM_PEER2

PEER1_ADMINQ       QM_PEER2           QM_PEER1
PEER2_TO_PEER1_Q   QM_PEER2           QM_PEER1
```

For each of these queues we inserted a message on the REMOTE side of the pair (using /usr/mqm/samp/bin/amqsput) and read it from the LOCAL side (using /usr/mqm/samp/bin/amqsget), as shown in Example 6-22. Testing these queues ensures that they are set up properly and that communications are operating properly between the queue managers.

Example 6-22 Test put and get from a remote queue

```
#####################################################
# Execute the following command to put a message into the queue
####################################################
#
/usr/mqm/samp/bin/amqsput PEER2_ADMINQ QM_PEER1
Sample AMQSPUT0 start
target queue is PEER2_ADMINTQ
This is a test message to a remote queue

Sample AMQSPUT0 end
#
####################################################
# Execute the following command to get the message from the queue.
# The program should display # the text one entered above then pause
# for a few seconds waiting for more messages.
####################################################
#
```

```
/usr/mqm/samp/bin/amqsget PEER2_ADMINQ QM_PEER2
Sample AMQSGET0 start
message <This is a test message to a remote queue>
no more messages
Sample AMQSGET0 end
```

Note: The Model Queues (IBMQREP.SPILL.MODELQ) cannot be tested—they are only models from which queues are dynamically created during a load operation.

Attention: Ensure that one “gets” all the messages that one “puts”. Do not leave test messages in the queues prior to configuring Q replication.

6.5.6 Step 6: Configure and activate Q replication using GUI

In this section we document the step-by-step configuration of Q replication for the peer-to-peer replication topology in our fictitious company. Figure 6-4 expands “STEP 6: Configure & activate Q replication (GUI or commands)” into a number of substeps involved in configuring Q replication. Each of these substeps is described in detail in the following subsections.

Very important: The Replication Center is typically used to configure and manage an SQL or Q Replication environment because of its ease-of-use GUI interface. In order for a Replication Center client to be aware of the database servers that need to be defined as a first and second server in Figure 6-30 on page 599 in a replication environment, the Q Capture and Q Apply control tables *must* be created from the *same* Replication Center client. If the Q Capture and Q Apply control tables are created by ASNCLP scripts, or using *another* Replication Center client, then those database servers will *not* appear in the list of available servers for the first and second server selection. In such cases, you must catalog them in this Replication Center client using the process described in Appendix H, “Cataloging remote database servers” on page 881.

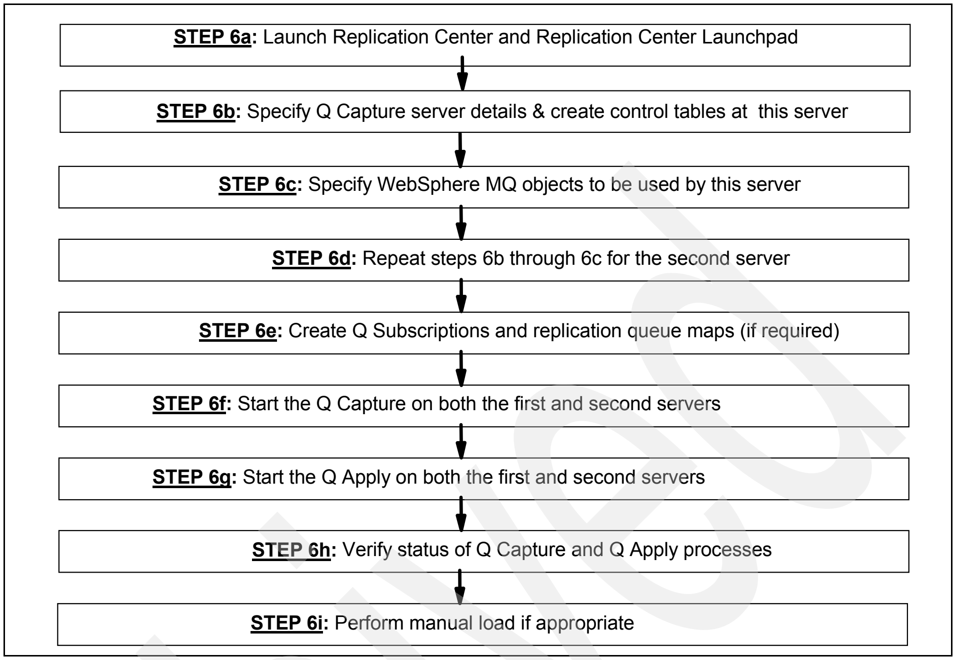


Figure 6-4 Overview of Q replication configuration steps

Step 6a: Launch Replication Center

Figure 6-5 shows the launching of the Replication Center from the DB2 Control Center by clicking the **Tools** tab and selecting **Replication Center**. This displays the Replication Center Launchpad screen shown in Figure 6-6.

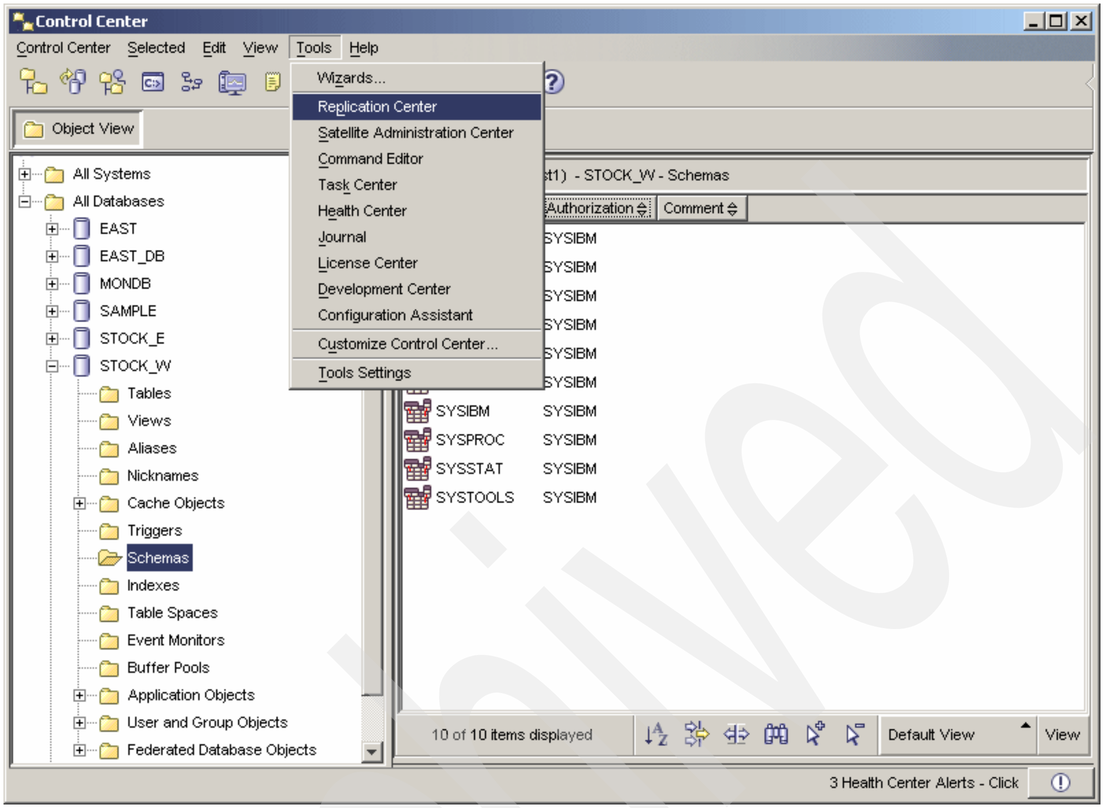


Figure 6-5 Launch Replication Center

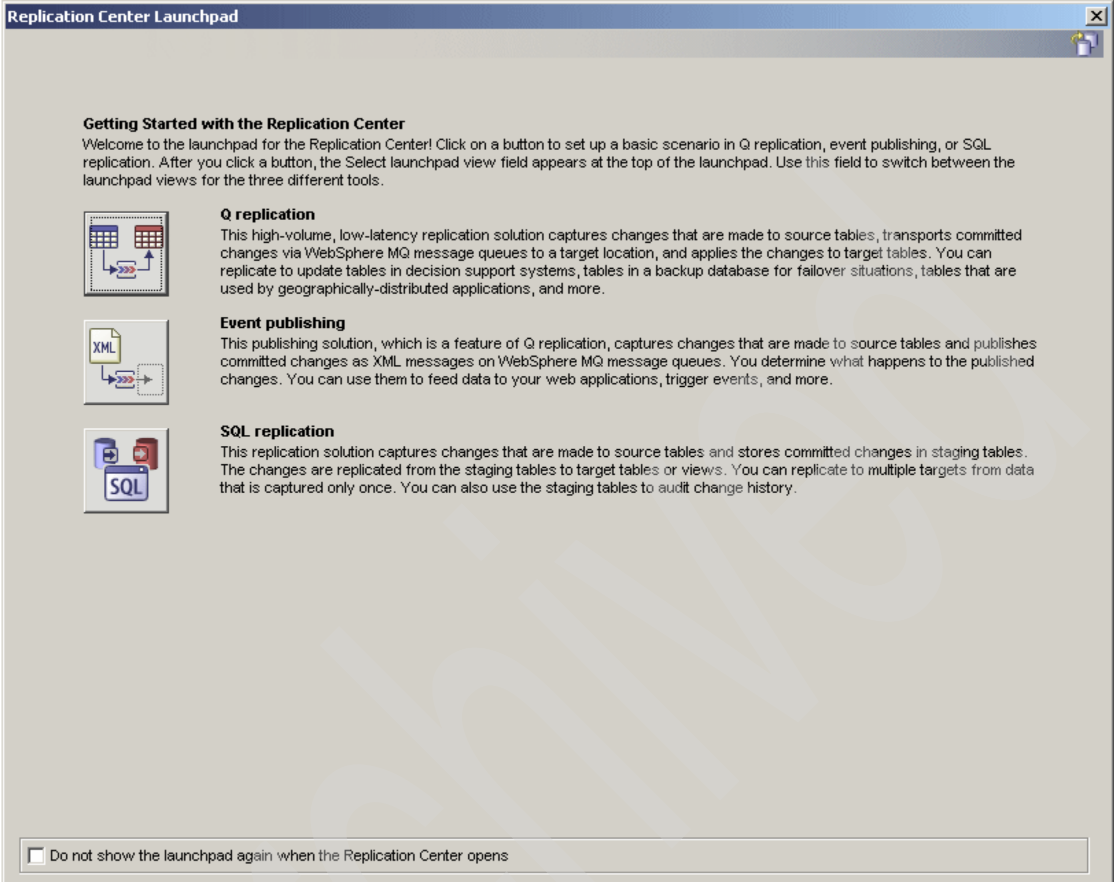


Figure 6-6 Replication Center Launchpad

> **Note:** Figure 6-6 can be skipped in the future by selecting the **Do not show this launchpad again when the Replication Center opens** option.

STEP 6b: Specify Q Capture details

Click **Q replication** in Figure 6-6 to display Figure 6-7, which describes the five steps in setting up and activating Q replication, as follows:

1. Create Q Capture control tables.
2. Create Q Apply control tables.
3. Create a Q Subscription.
4. Start a Q Capture program.
5. Start a Q Apply program.

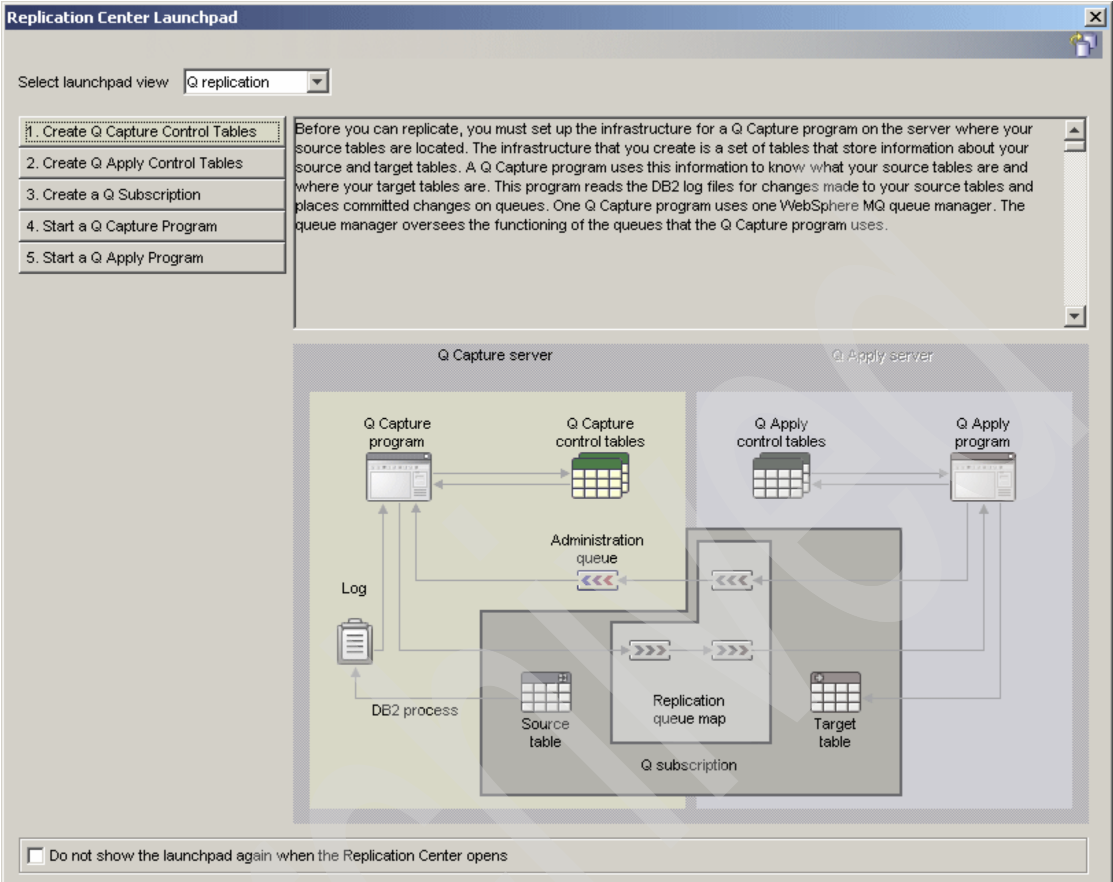


Figure 6-7 Five steps in setting up Q replication infrastructure

Click **Create Q Capture Control Tables** to display Figure 6-8, which enables the specification of details for the Q Capture infrastructure.

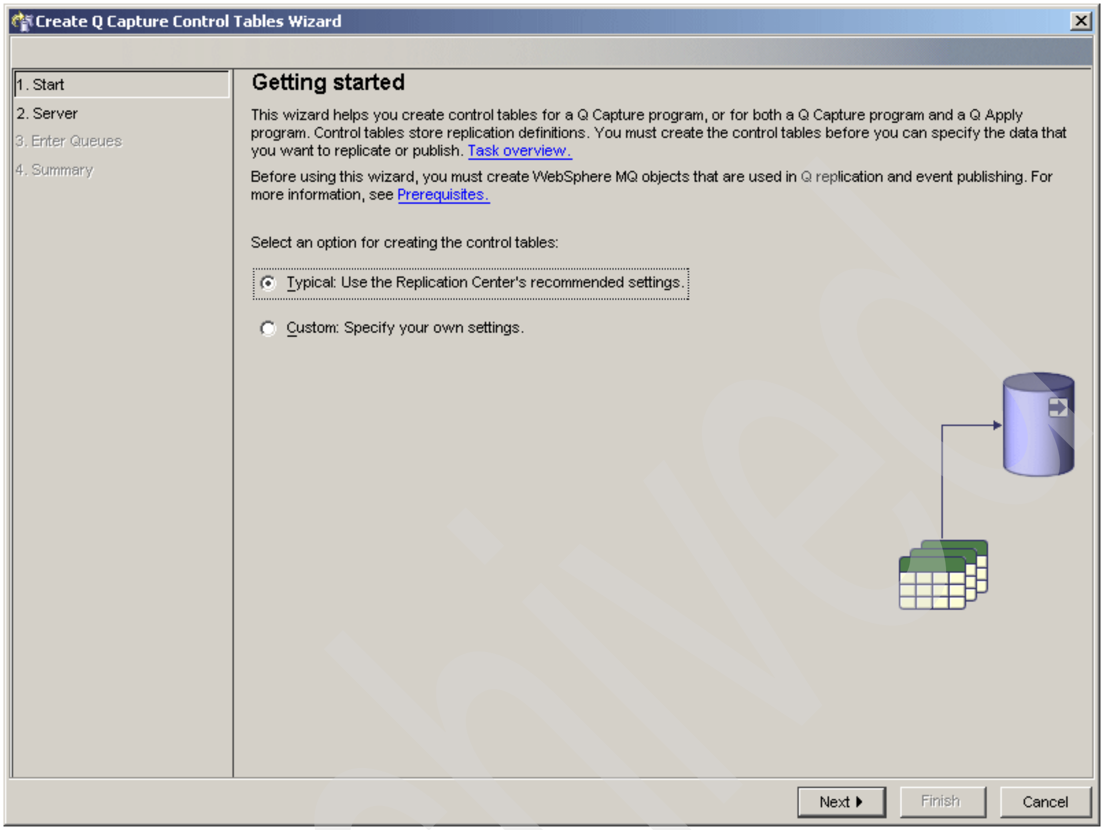


Figure 6-8 Getting Started - Create Q Capture Control Tables Wizard

We selected the **Typical** setting and clicked **Next** to display Figure 6-9, where the Q Capture server details are specified.

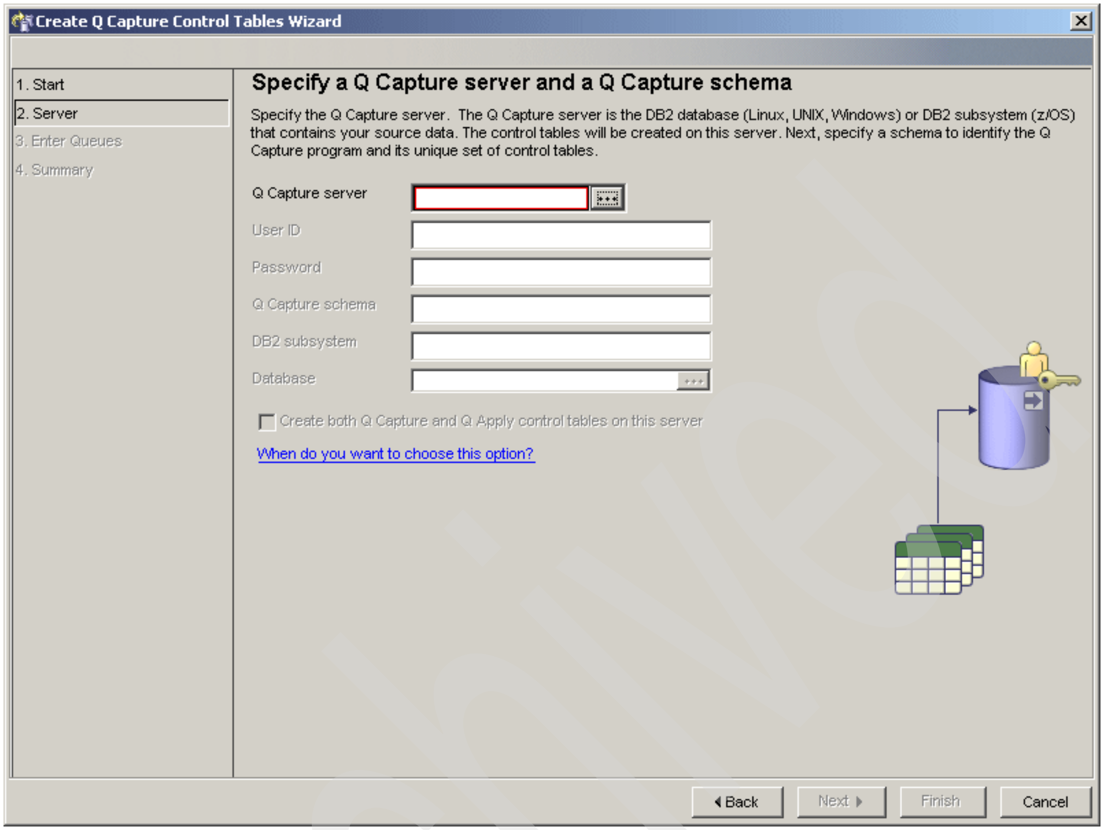


Figure 6-9 Server - Create Q Capture Control Tables Wizard

Click the **...** tab to display the list of available servers, as shown in Figure 6-10.

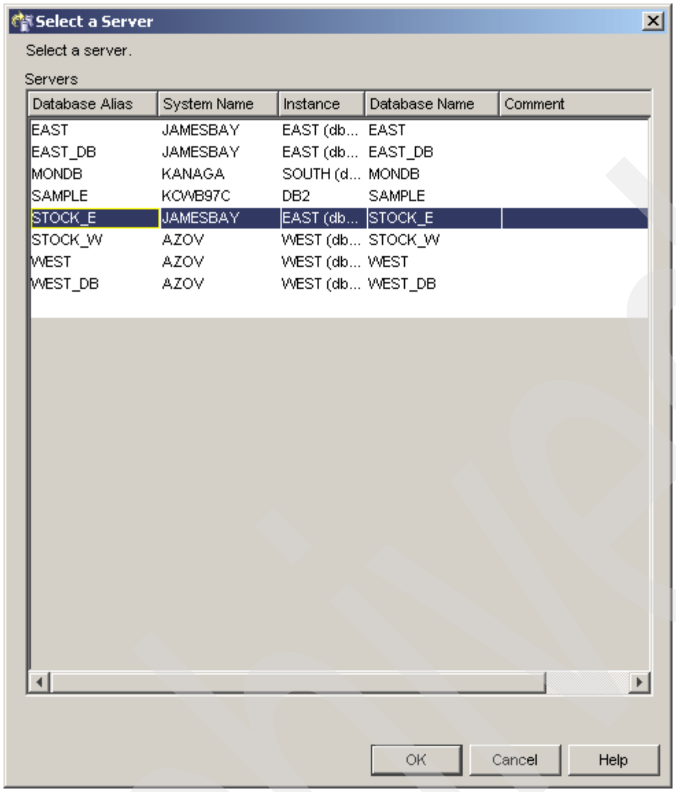


Figure 6-10 Select a Server

Select the **STOCK_E** database alias on Jamesbay as the database server for Q Capture, and click **OK** to display Figure 6-11 for providing Q Capture details.

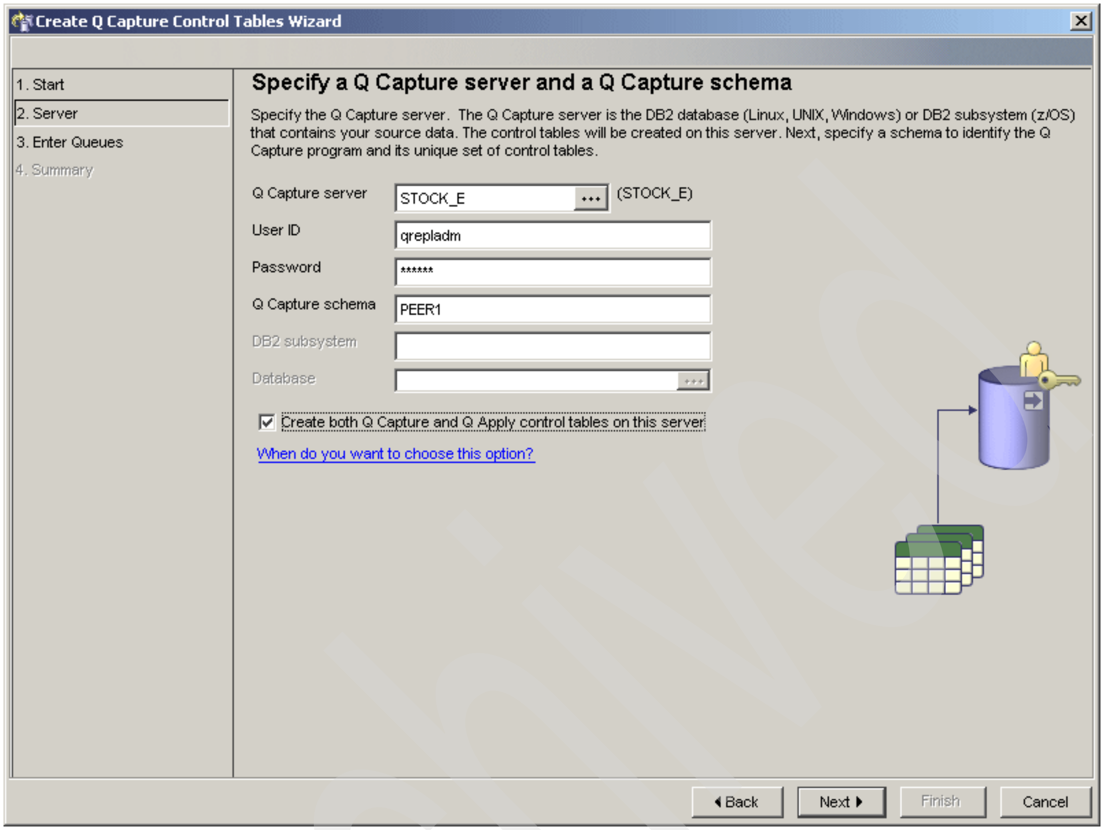


Figure 6-11 Q Capture server and schema details

Supply the Q Capture user ID and password qrepladm/xxx (reference A.7b in Table 6-2 on page 550) and Q Capture schema PEER1 (reference A.22 in Table 6-3 on page 552), and select the **Create both Q Capture and Q Apply control tables on this server** option.

Click **Next** to display Figure 6-12 for specifying WebSphere MQ objects.

Step 6c: Specify WebSphere MQ objects

Supply WebSphere MQ queues for this server. This includes the Queue Manager QM_PEER1 (reference A.11 in Table 6-3 on page 552), Administration queue PEER1_ADMINQ (reference A.17 in Table 6-3 on page 552), and Restart queue PEER1_RESTARTQ (reference A.16 in Table 6-3 on page 552).

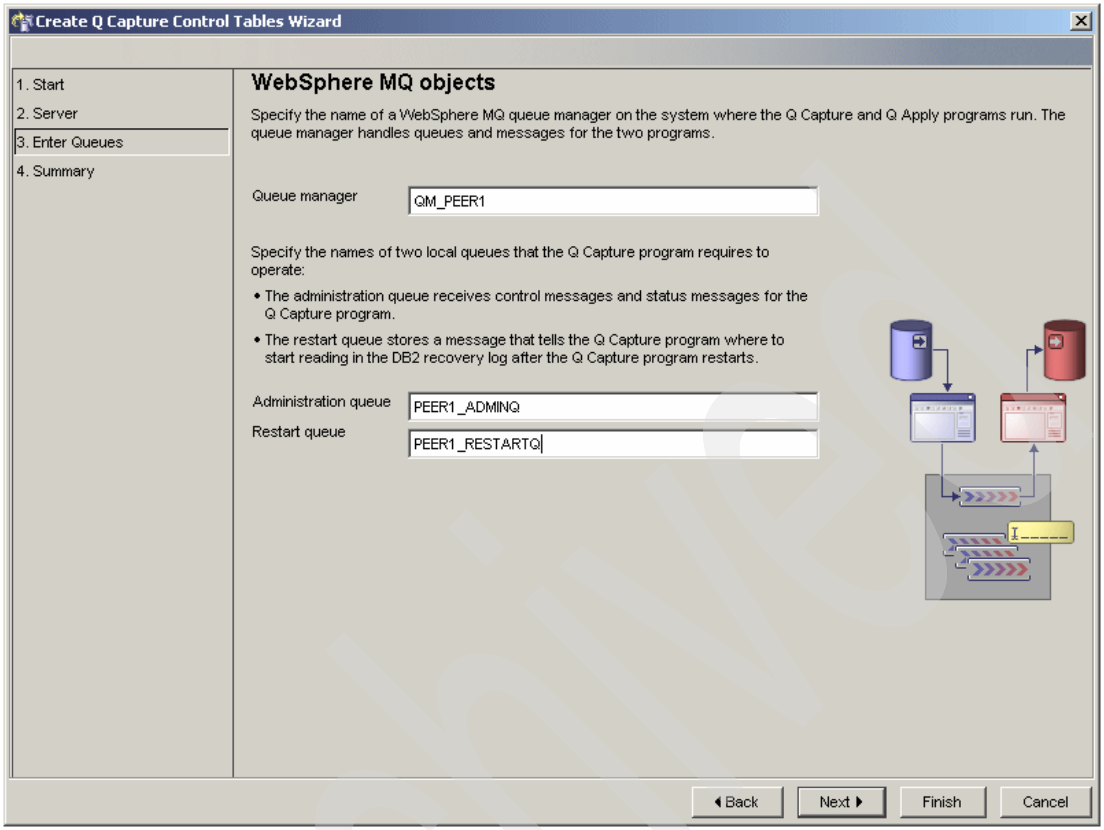


Figure 6-12 Q information for Q Capture server

Click **Next** to continue to Figure 6-13, which summarizes the information provided in the previous screens.

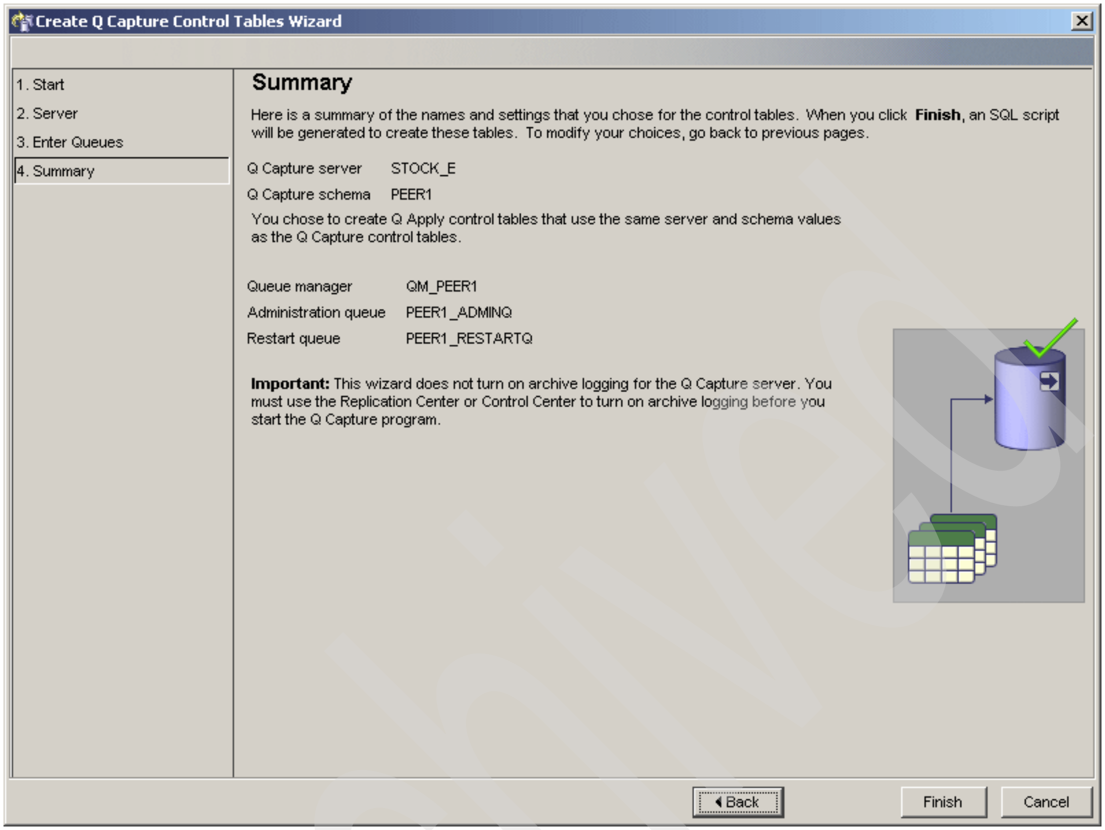


Figure 6-13 Q Capture server summary information

Verify the accuracy of the information supplied, and click **Finish** to generate the scripts that need to be executed on the Q Capture server to define the specified objects, as shown in Figure 6-14.

Click the **Back** button to go back and correct any errors.

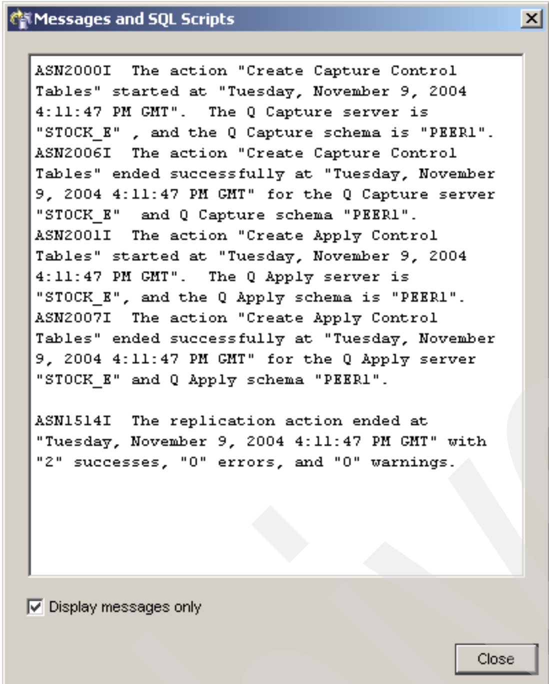


Figure 6-14 Messages and SQL Scripts screen for Q Capture server

Check that the errors are 0. Click **Close** to display Figure 6-15, and decide whether to execute the script or save it for later execution.

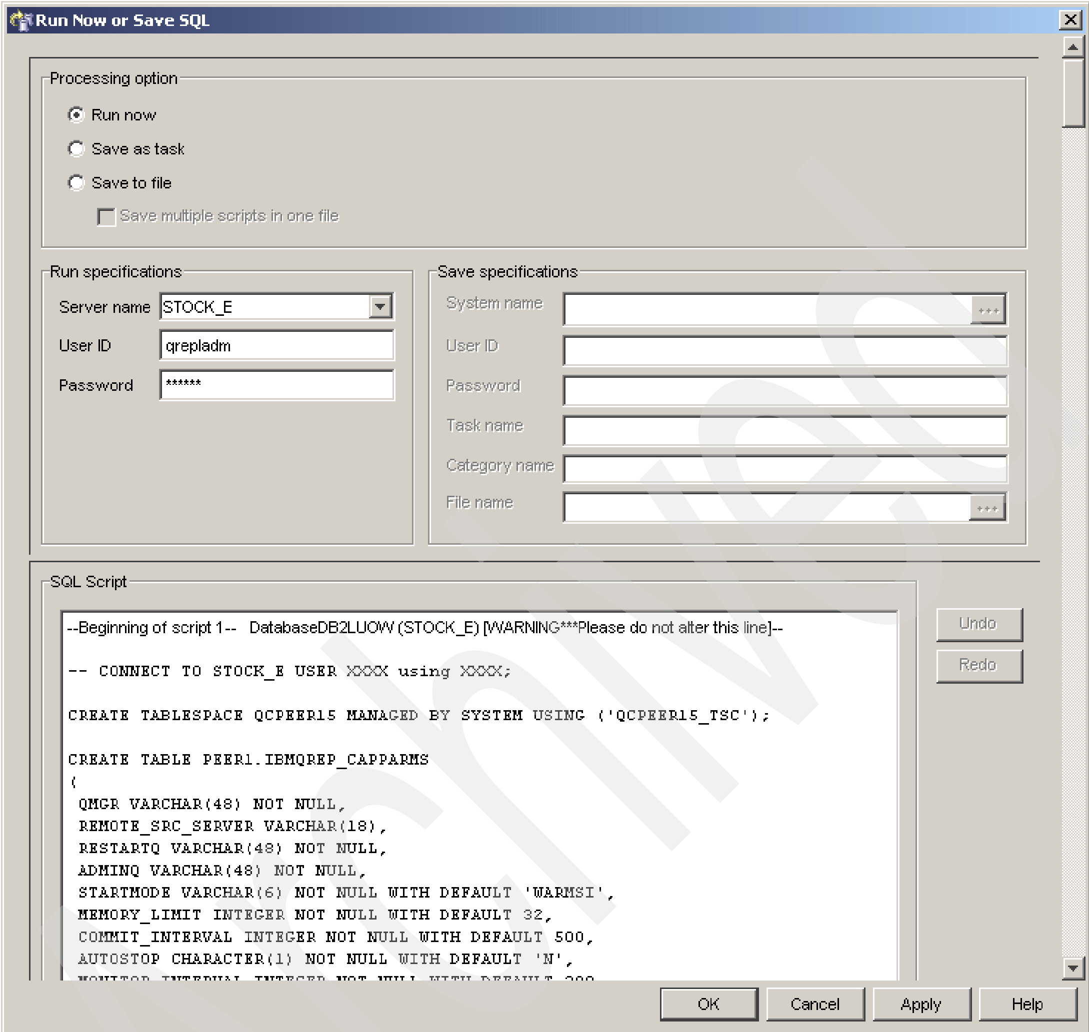


Figure 6-15 Run Now or Save SQL screen

Select the **Run now** option and click **OK** to display the message that the scripts executed successfully, as shown in Figure 6-16.

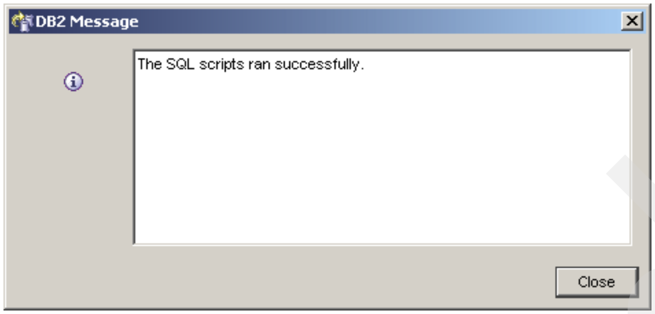


Figure 6-16 SQL scripts ran successfully message

Click **Close** to complete the definition of the Q Capture server infrastructure.

Step 6d: Repeat steps 6b through 6c for second server

The same steps for the Q Capture server on the PEER1 node (Jamesbay) need to be repeated for the PEER2 node (Azov) using the appropriate reference information in Table 6-3 on page 552, as shown in Figure 6-17 through Figure 6-26 on page 595.

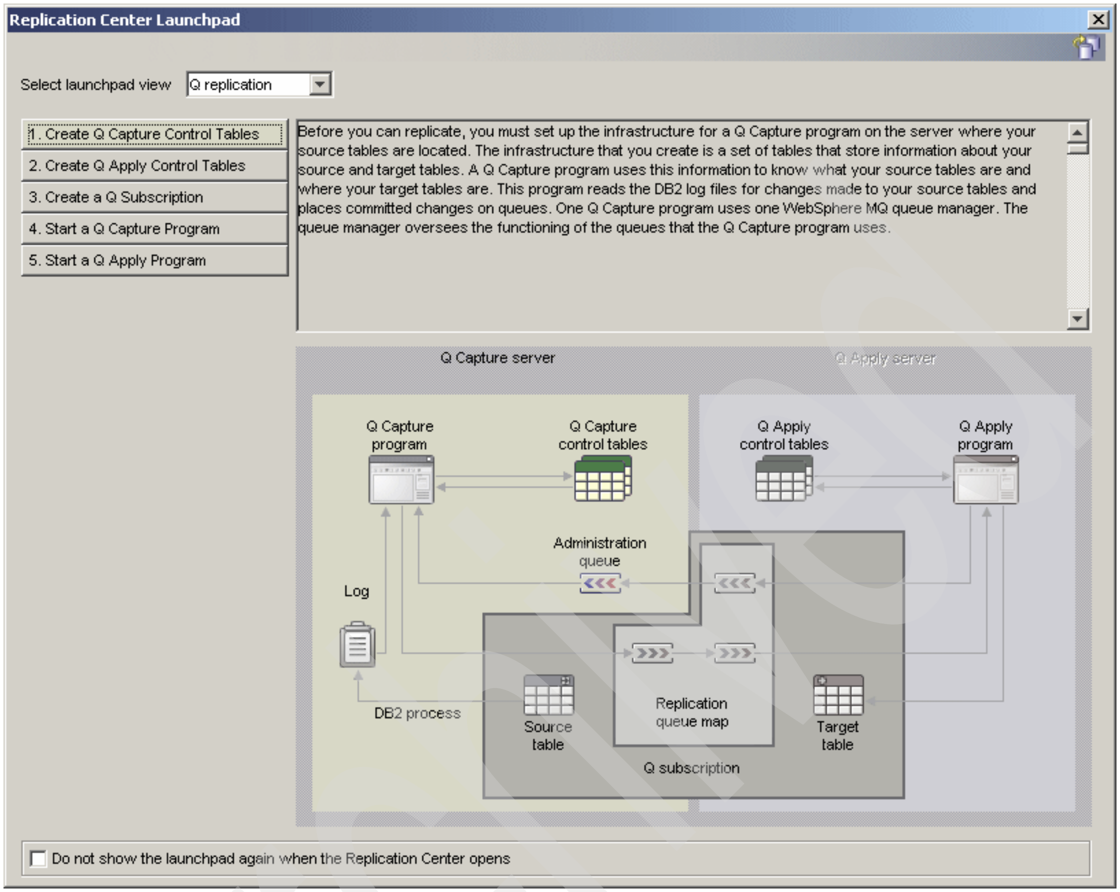


Figure 6-17 Five steps in setting up Q replication infrastructure

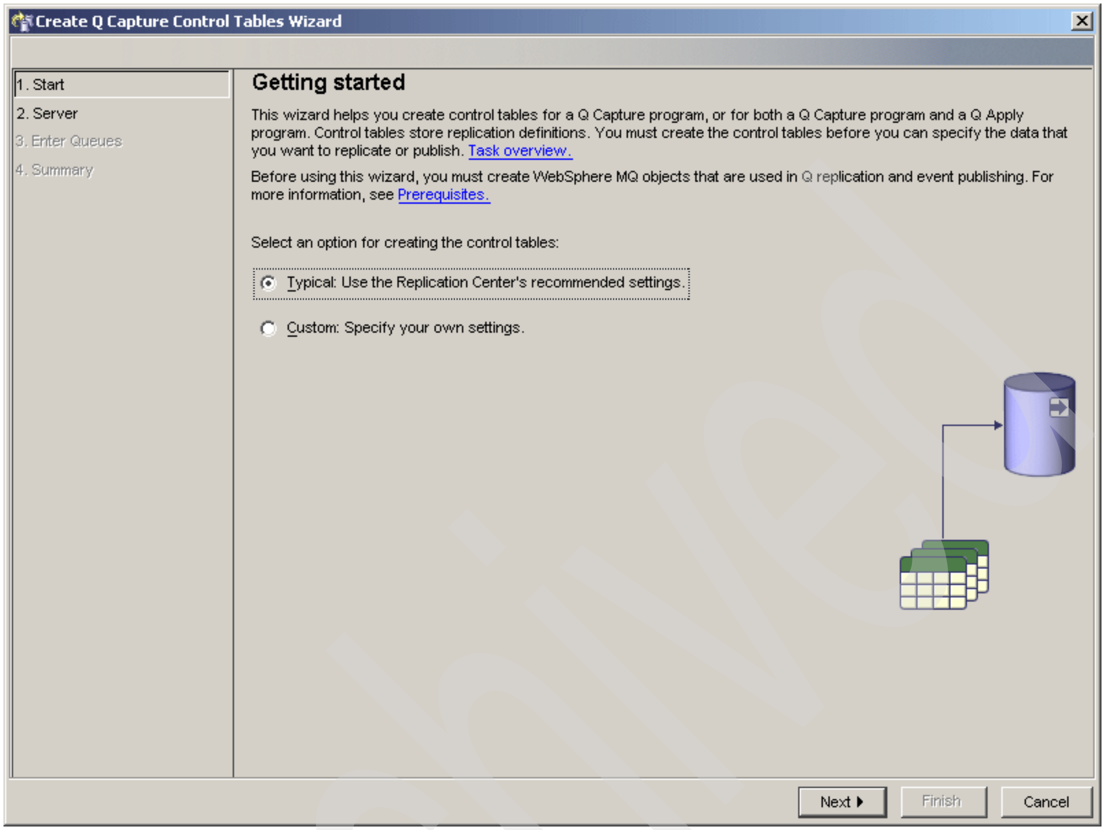


Figure 6-18 Getting started - Create Q Capture Control Tables Wizard

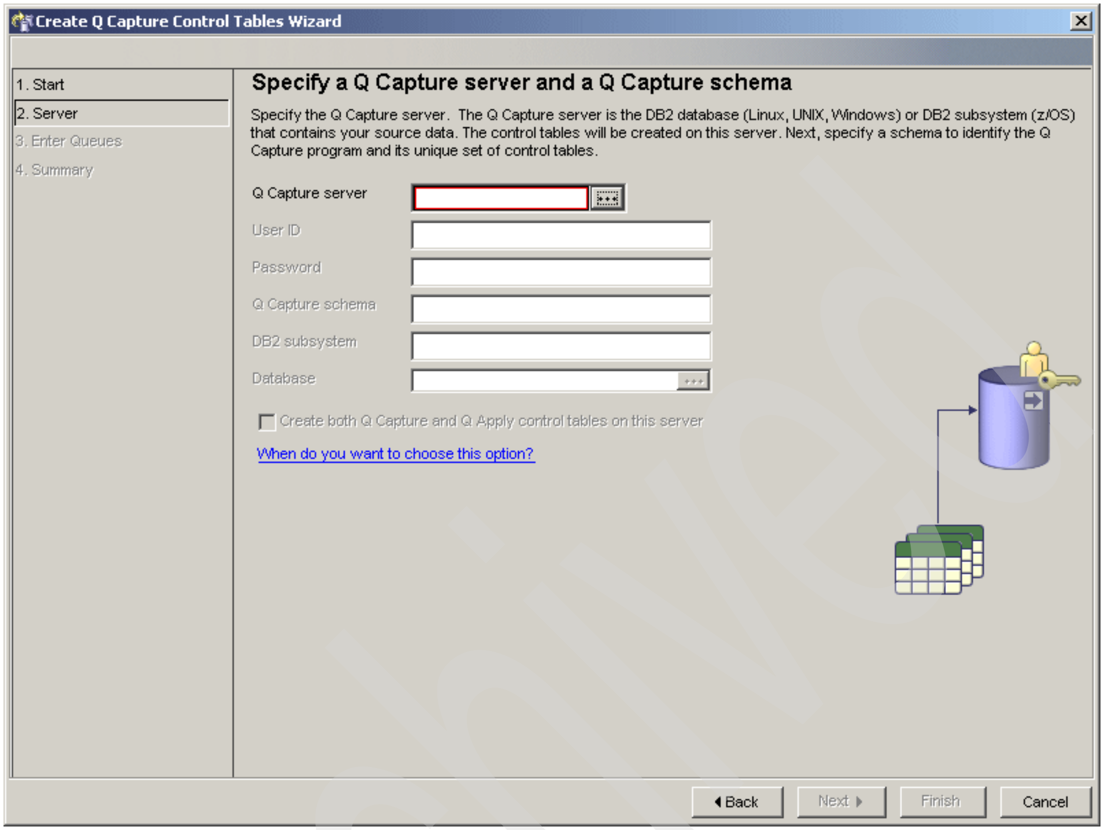


Figure 6-19 Server - Create Q Capture Control Tables Wizard

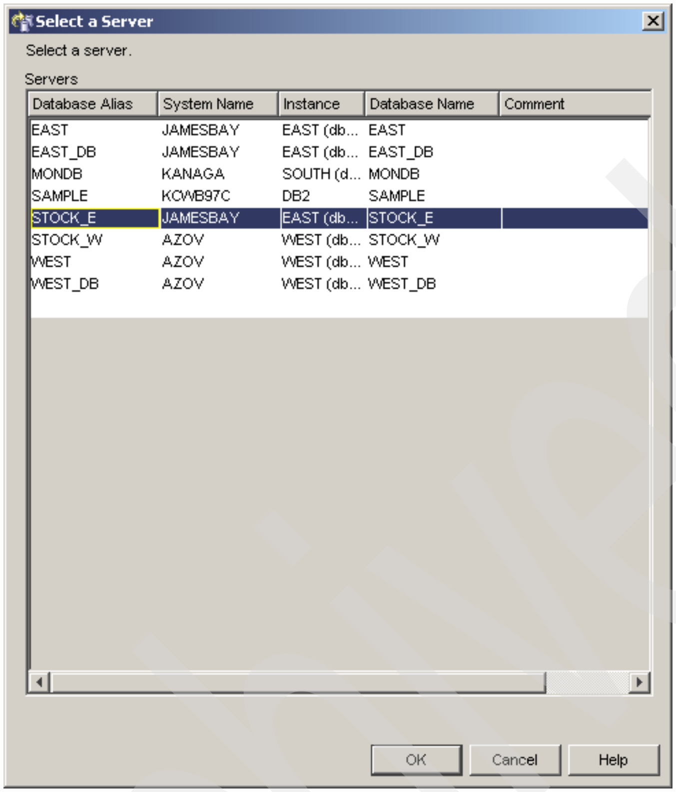


Figure 6-20 Select a Server

Note: We actually selected STOCK_W in Figure 6-20 and clicked **OK** to proceed to the next figure.

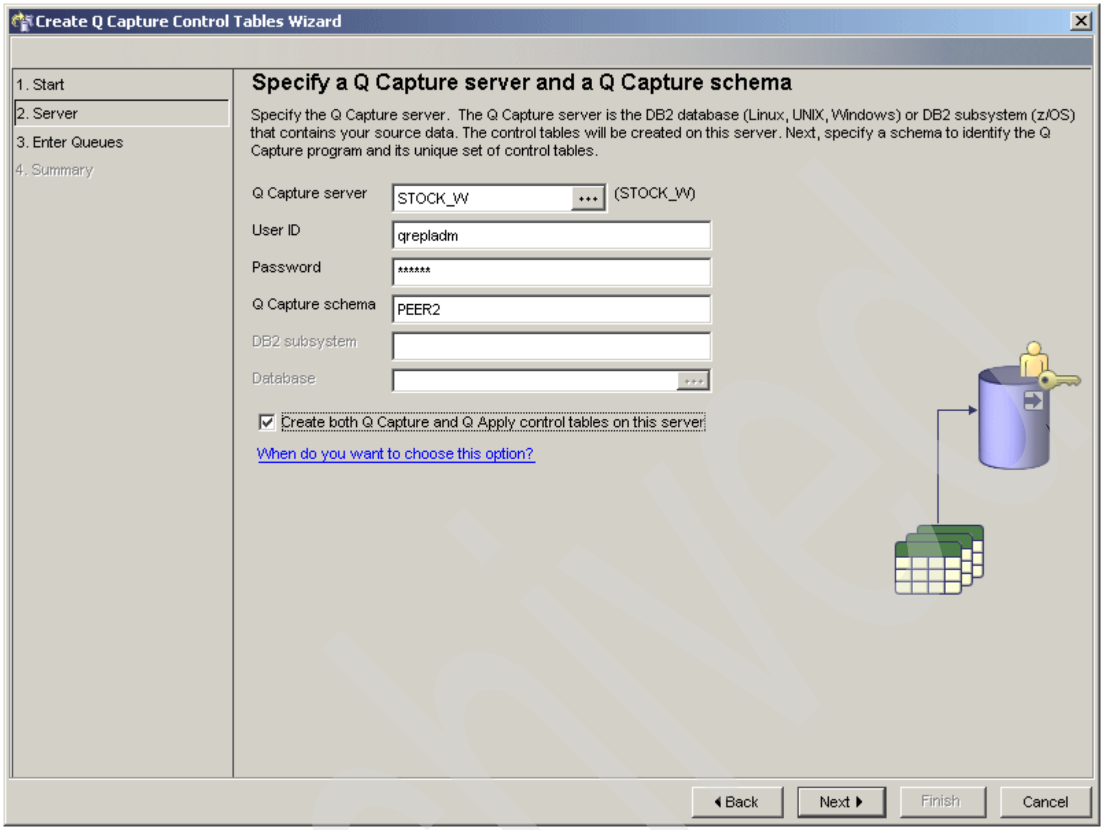


Figure 6-21 Q Capture server and schema details

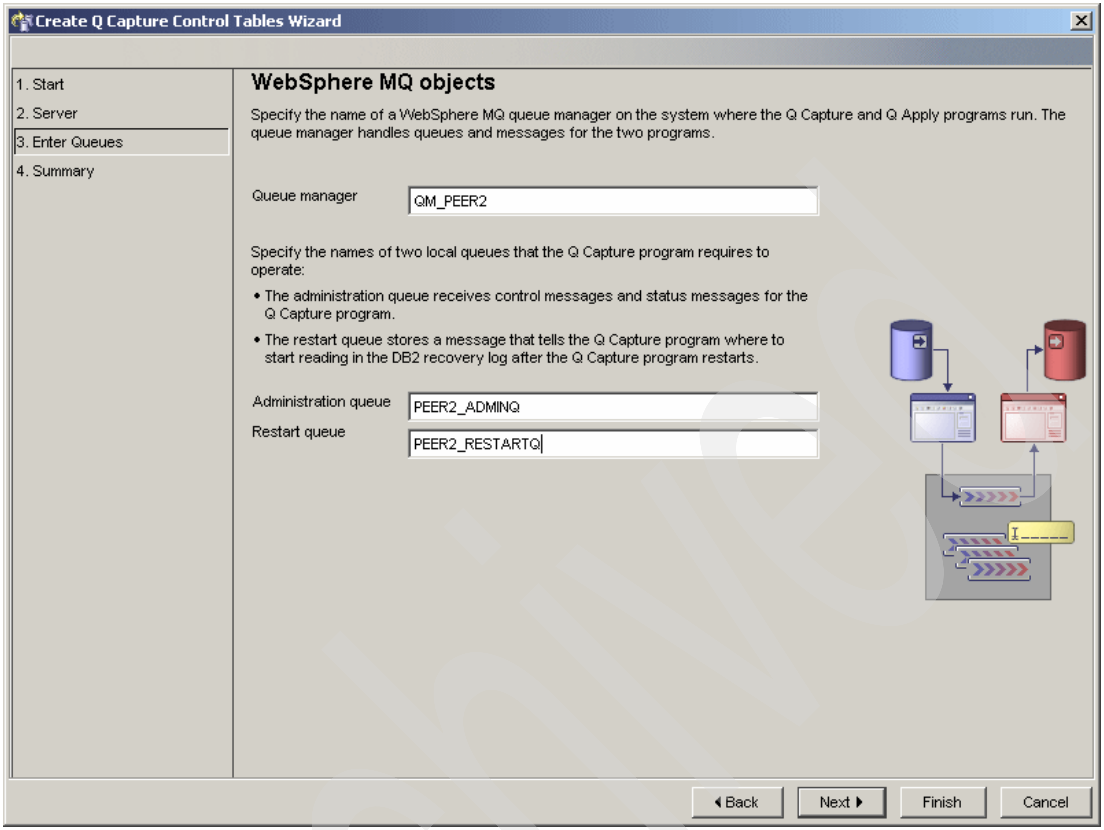


Figure 6-22 Q information for Q Capture server

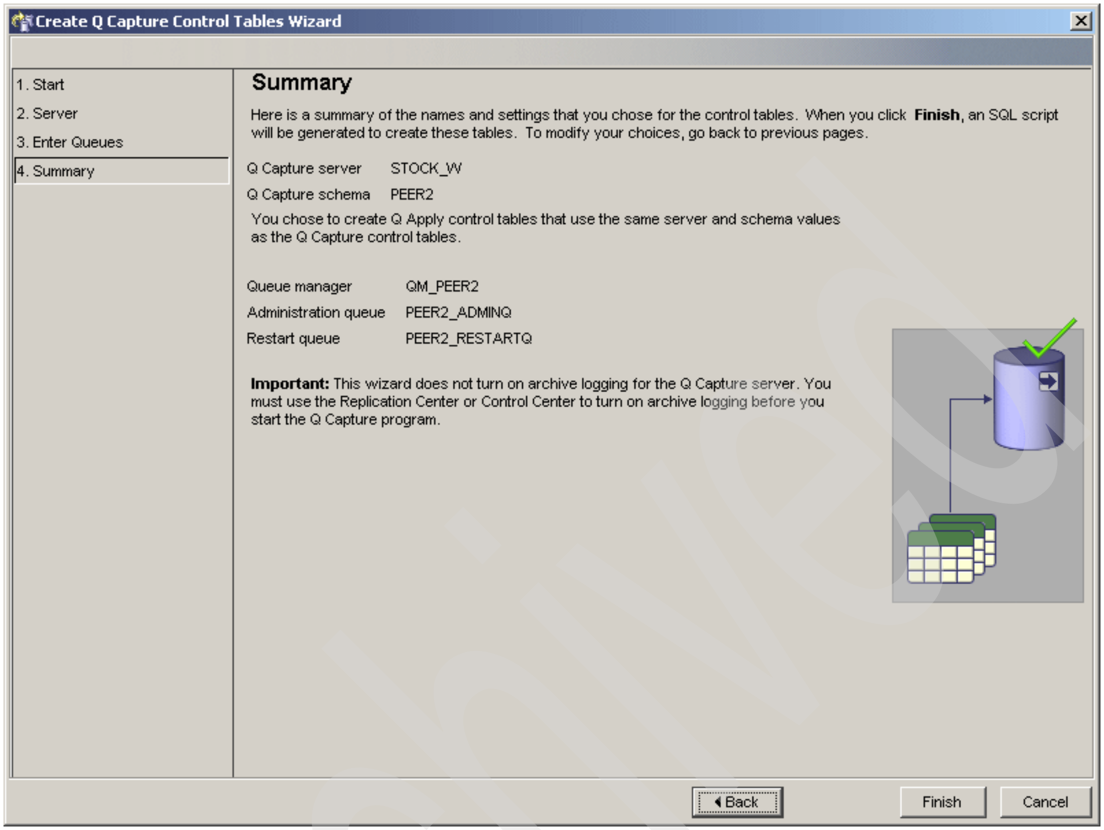


Figure 6-23 Q Capture server summary information

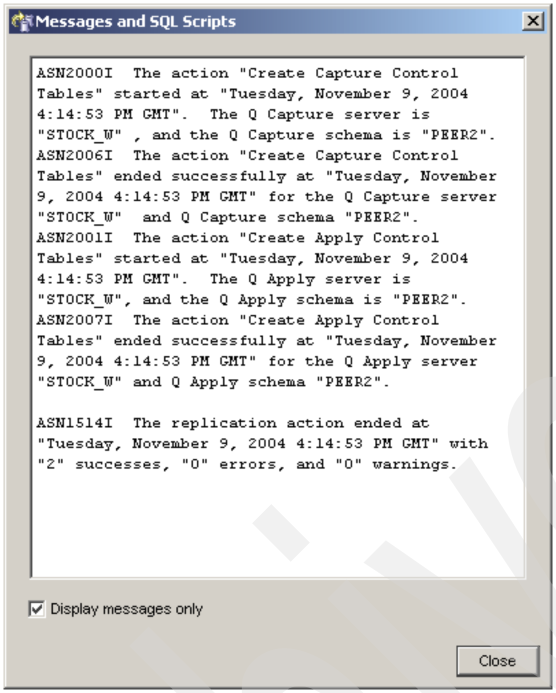


Figure 6-24 Messages and SQL Scripts screen for Q Capture server

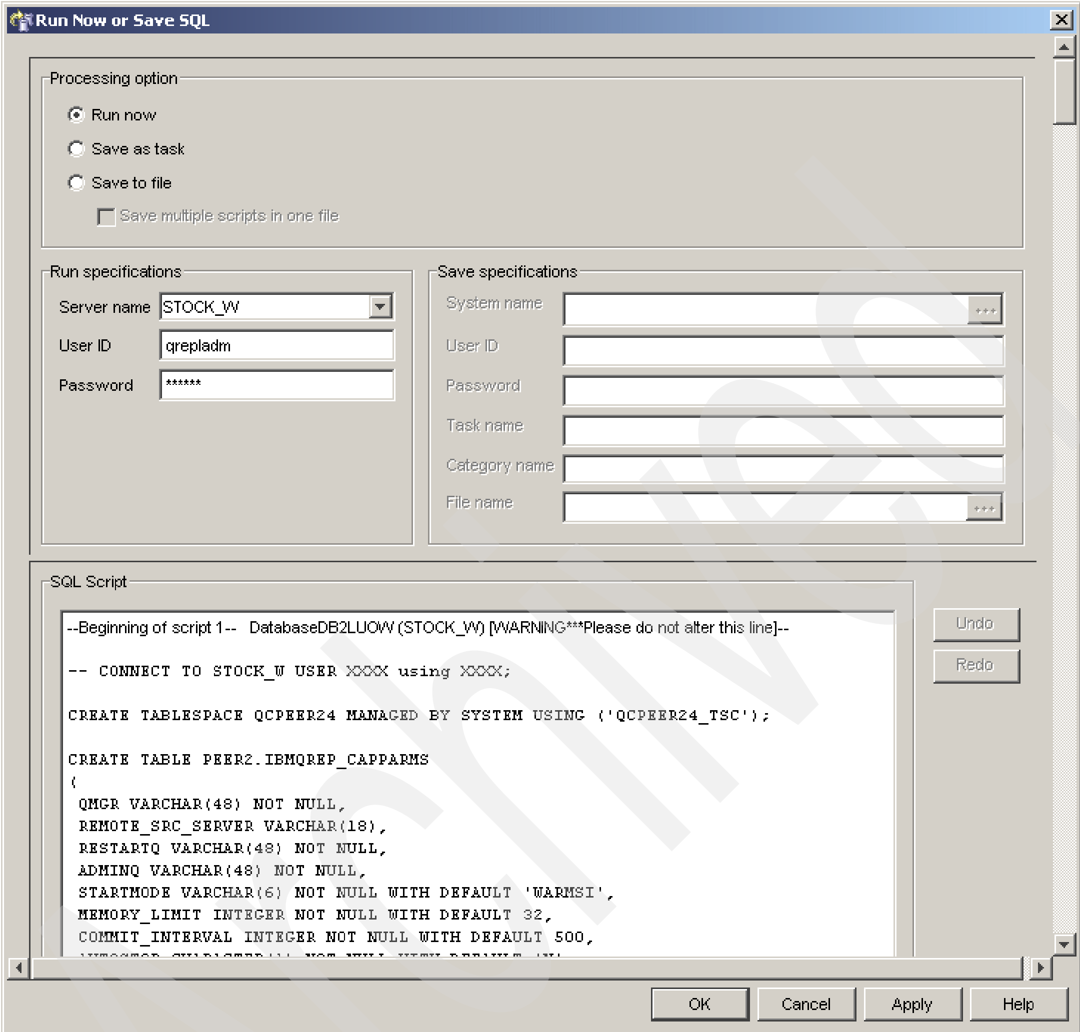


Figure 6-25 Run Now or Save SQL screen

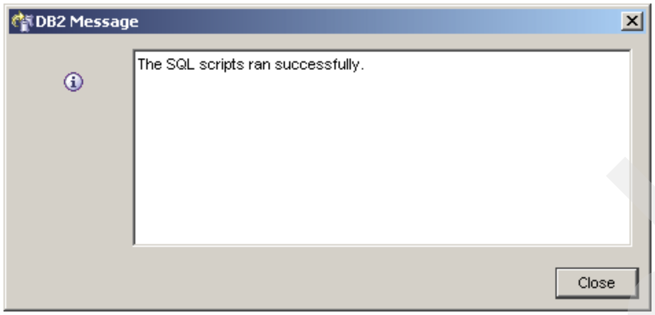


Figure 6-26 SQL scrips ran successfully message

Step 6e: Create Q Subscriptions and replication queue maps

After the Q Capture and Q Apply Control Tables have been created, the Q subscriptions and replication queue maps need to be created. Click the **Create a Q Subscription** option in Figure 6-27 to display Figure 6-28 to begin the process.

> **Note:** Since we created the Q Apply control tables earlier by selecting the **Create both Q Capture and Q Apply control tables on this server** in Figure 6-21 on page 590, we skip the Create Q Apply Control Tables option in Figure 6-28.

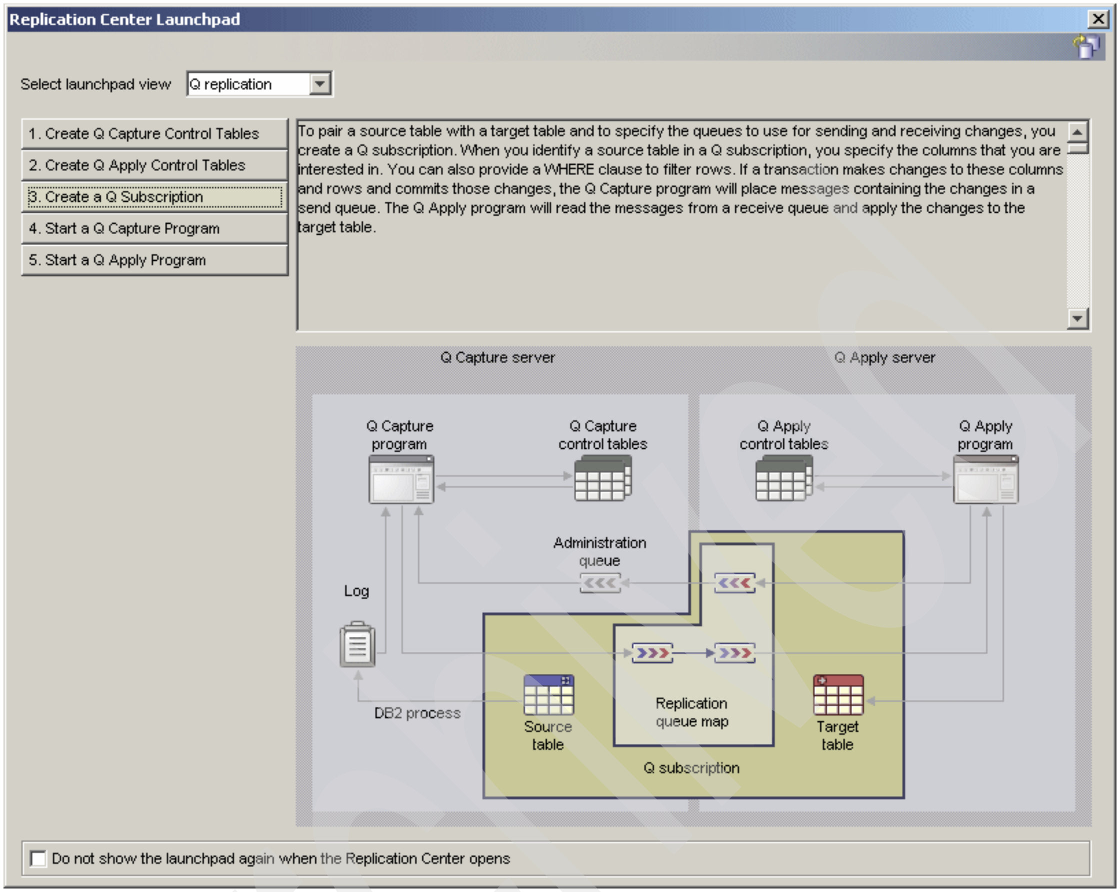


Figure 6-27 Five steps in setting up Q replication infrastructure

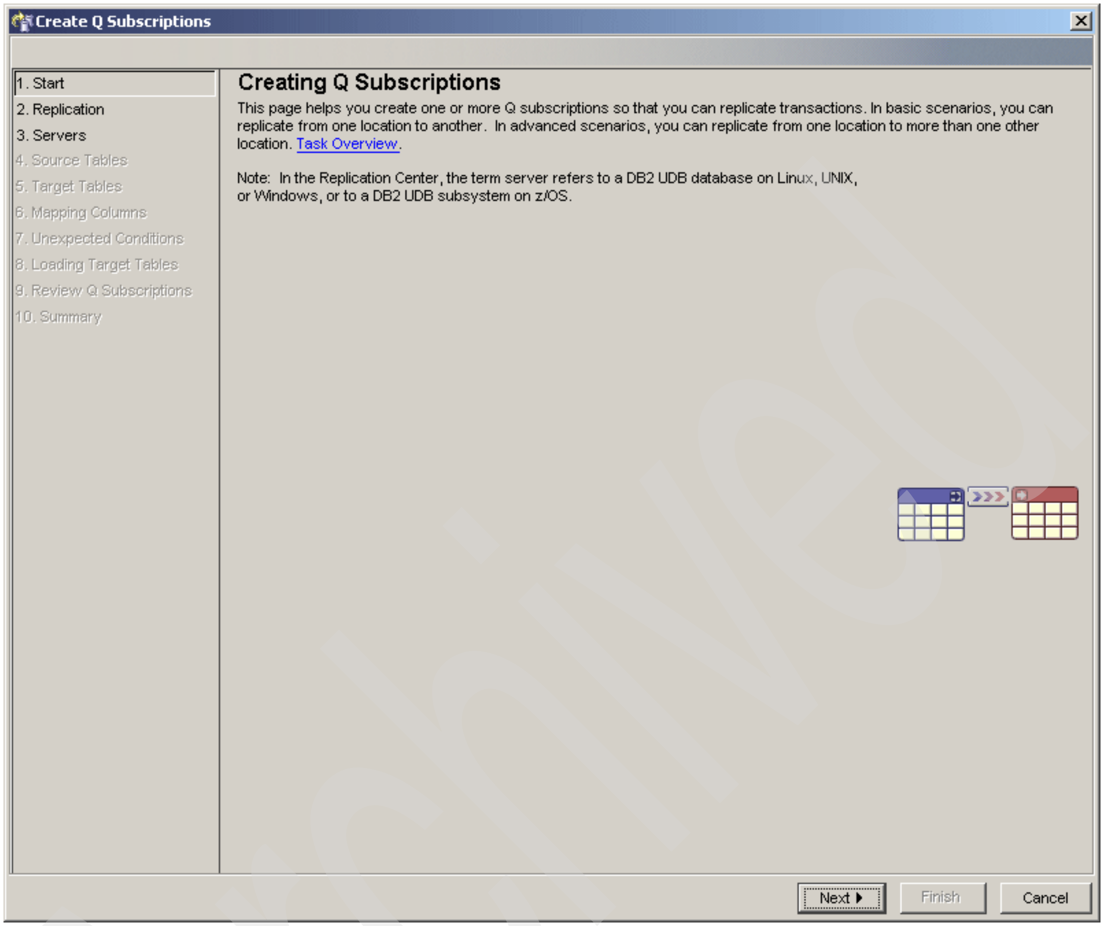


Figure 6-28 Create Q Subscriptions

Click **Next** to continue to Figure 6-29 to choose the type of replication topology desired.

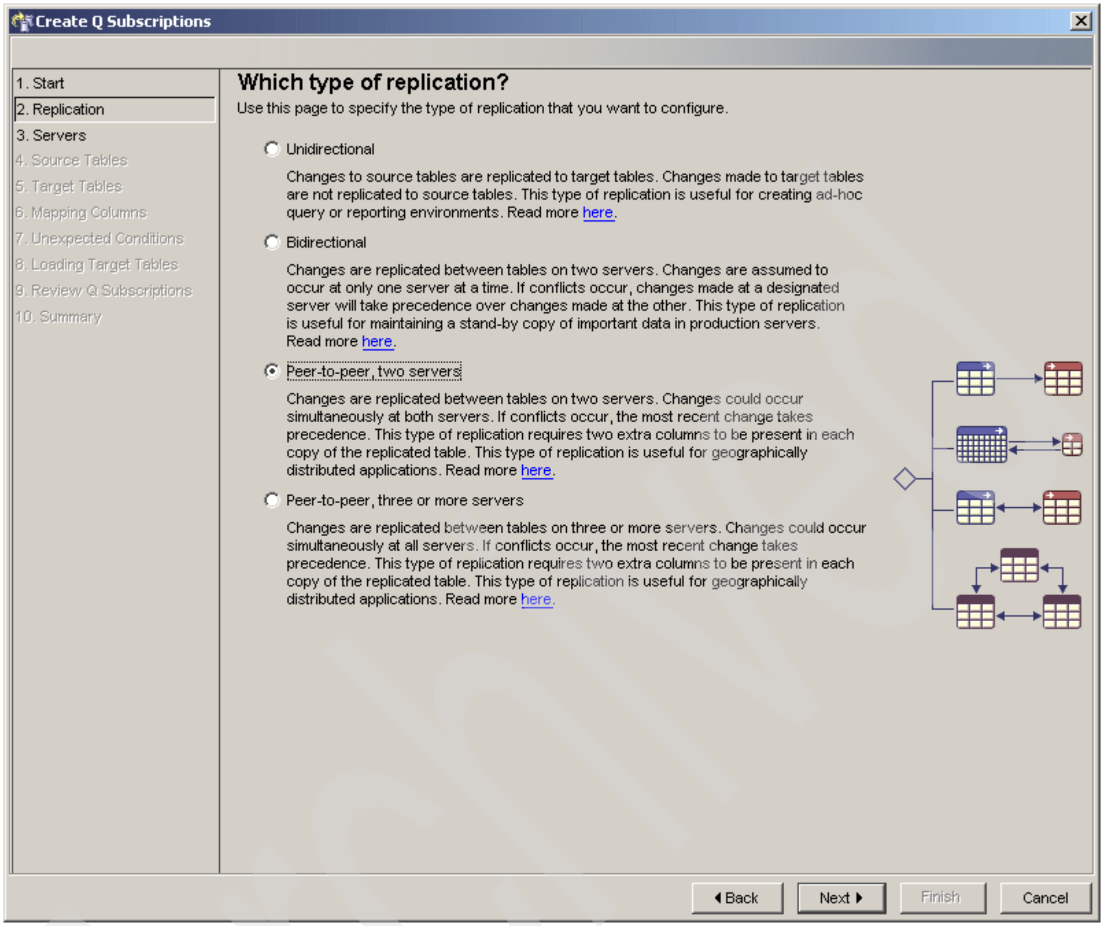


Figure 6-29 Which type of Replication

Select the **Peer-to-peer, two servers** setting and click **Next** to provide details about the servers involved in the replication topology, as shown in Figure 6-30.

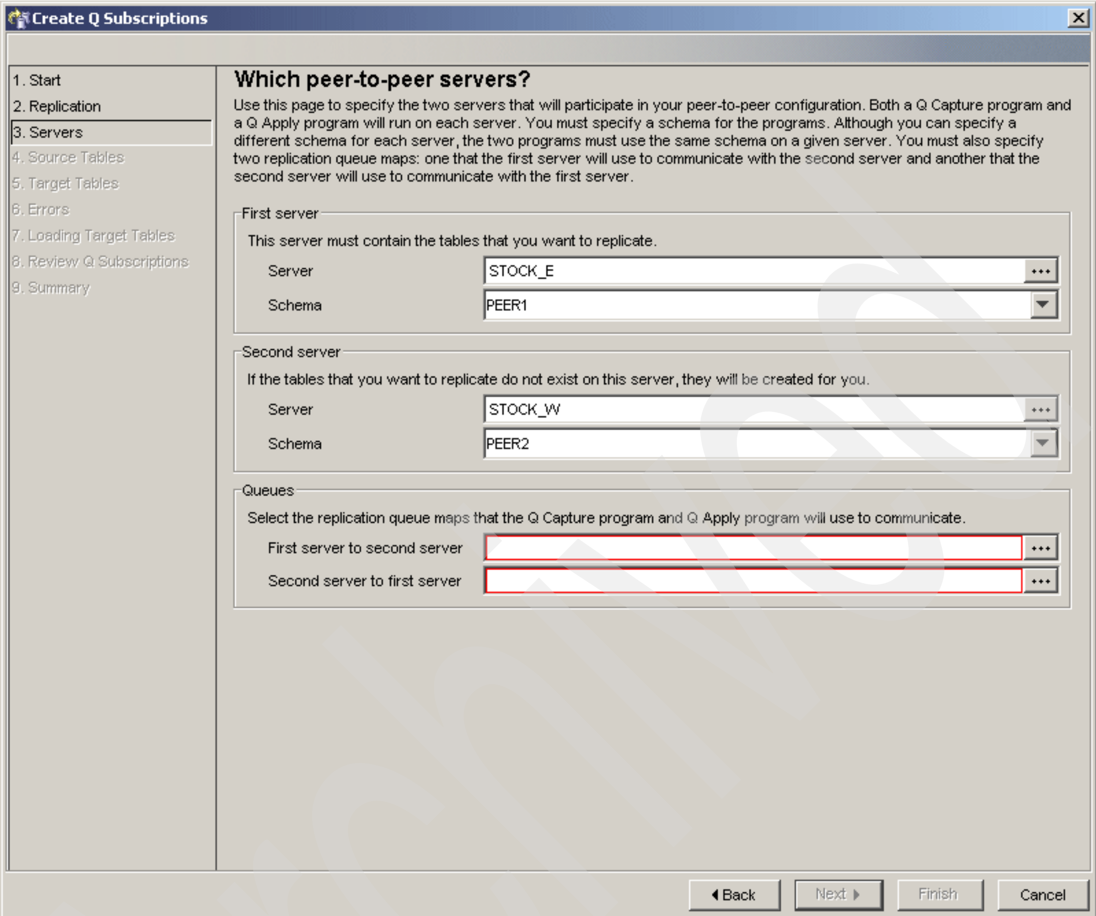


Figure 6-30 Which peer-to-peer servers

Click the **...** button for the “First server to second server” to get a list of the available replication queue maps, as shown in Figure 6-31.

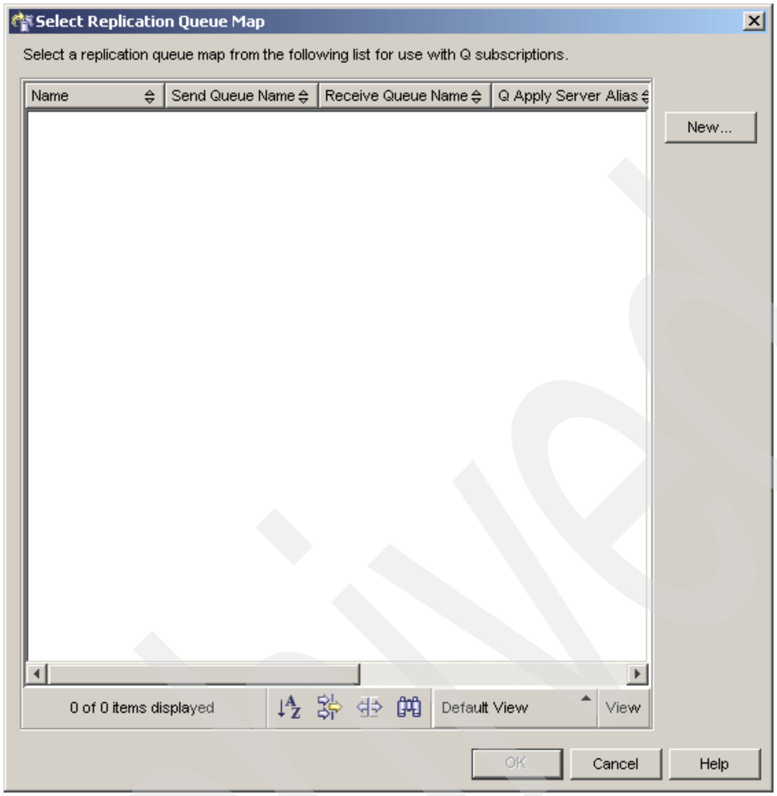


Figure 6-31 Select Replication Queue Map

Since no replication queue maps have been defined as yet, Figure 6-31 has no list of existing replication queue maps. A new replication queue map needs to be created by clicking the **New...** button. This displays Figure 6-32.

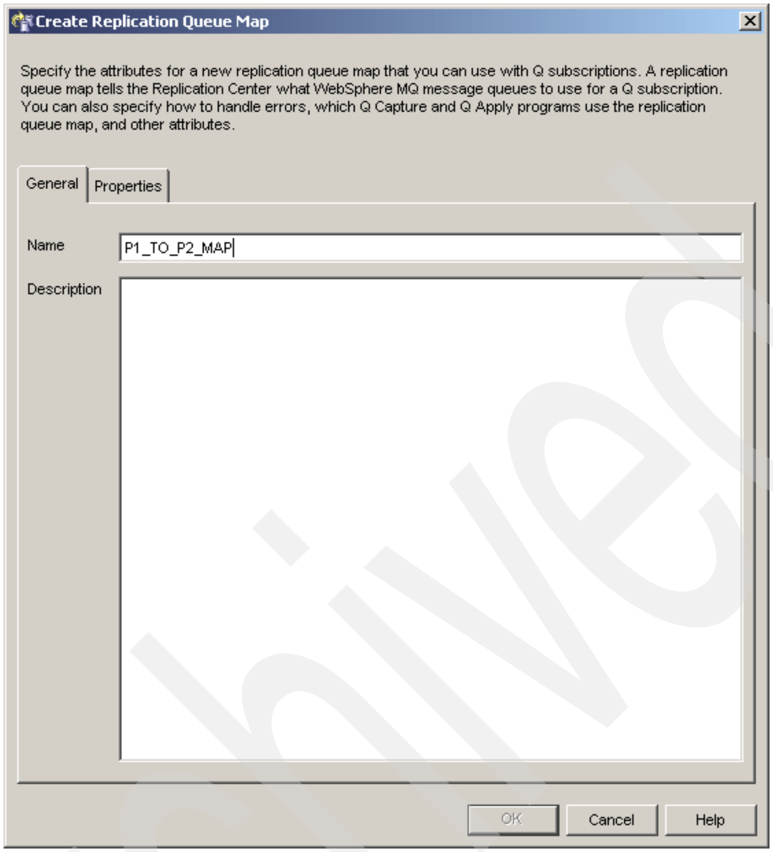


Figure 6-32 Create Replication Queue Map

Supply the name P1_TO_P2_MAP (reference A.27 in Table 6-3 on page 552), and then choose the **Properties** tab to display Figure 6-33.

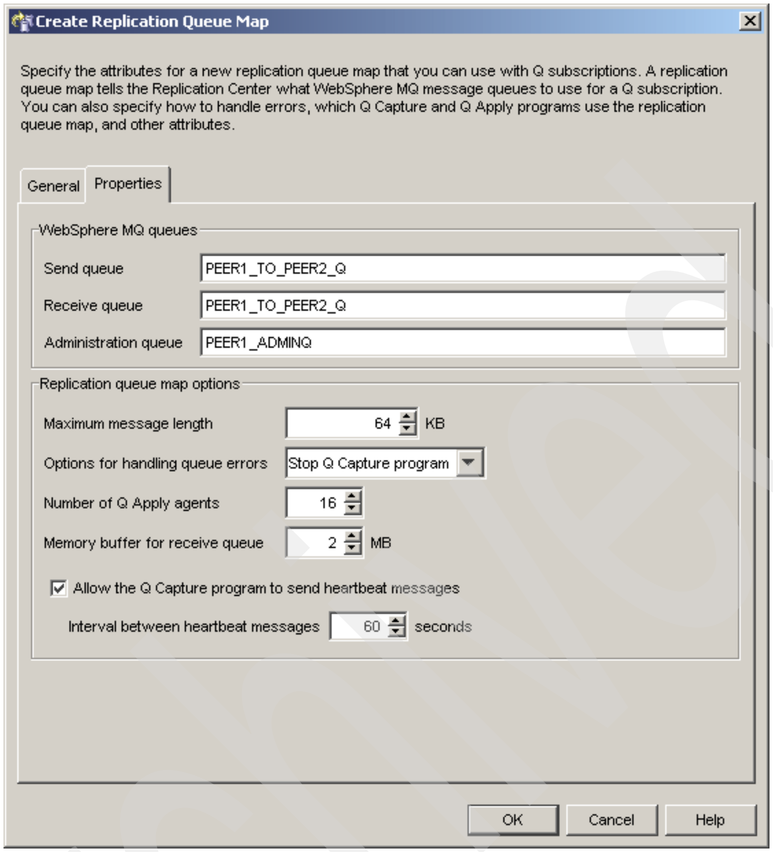


Figure 6-33 Create Replication Queue Map - Properties

Supply values for the send queue (PEER1_TO_PEER2_Q"reference A.18 in Table 6-3 on page 552), receive queue (PEER1_TO_PEER2_Q reference B.19 in Table 6-2 on page 550), administration queue (PEER1_ADMINQ reference A.17 in Table 6-3 on page 552), and let other values default.

Click **OK** to complete the definition, and generate the scripts for creating the replication queue map objects on the servers, as shown in Figure 6-36.

> **Attention:** In fixpak 9, which is due out in April 2005, the screens shown in Figure 6-32 and Figure 6-33 have been enhanced to appear as shown in Figure 6-34 and Figure 6-35, respectively. Please ignore the values shown therein, since the screens have been shown to only highlight the difference in the new screen layout.

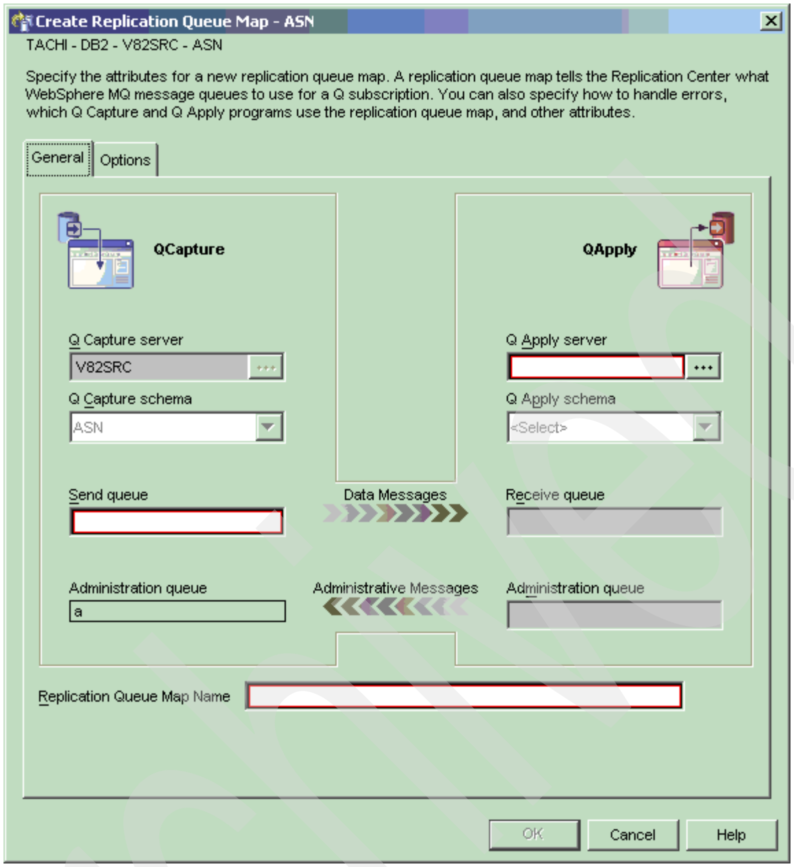


Figure 6-34 Fixpak 9 Create Replication Queue Map screen

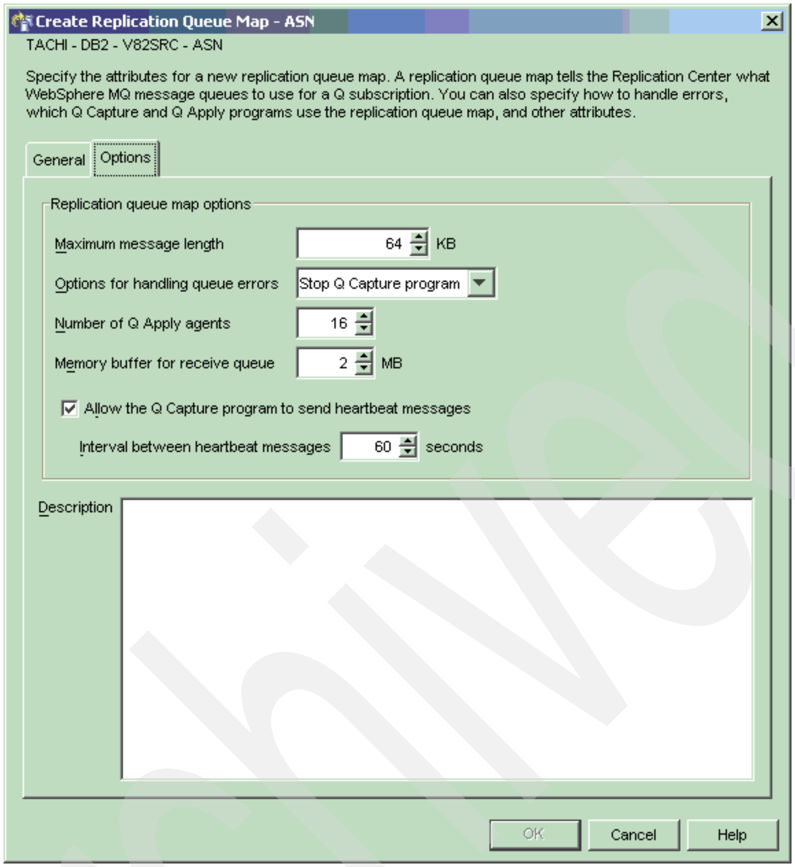


Figure 6-35 Fixpak 9 Create Replication Queue Map - Options screen

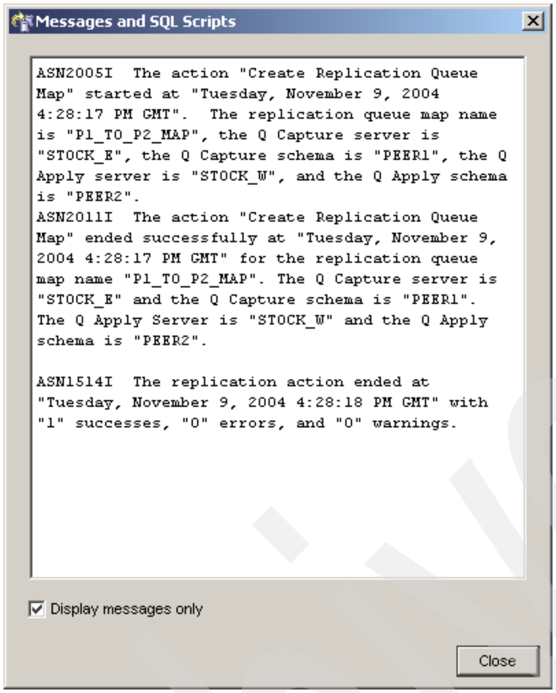


Figure 6-36 Messages and SQL Scripts

Click **Close** to bring up Figure 6-37 for choosing whether to execute the scripts right away or save the scripts for later execution.

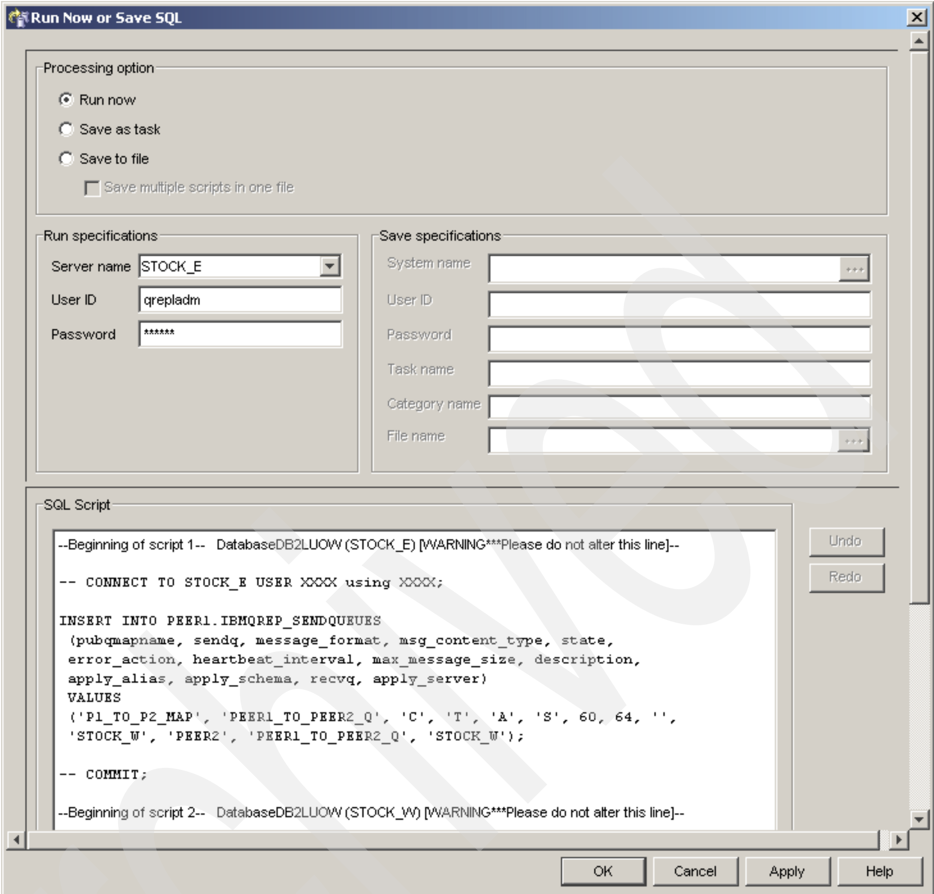


Figure 6-37 Run Now or Save SQL

Select the **Run now** option and click **OK** to execute the script immediately. Figure 6-38 is displayed when the scripts run successfully.

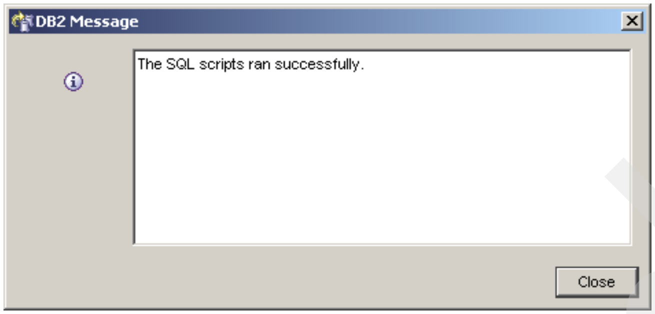


Figure 6-38 SQL scripts ran successfully message

Click **Close** to proceed to Figure 6-39, which shows the just created replication queue map. Highlight the row with the P1_TO_P2_MAP in the Name column and click **OK** to proceed to Figure 6-40.

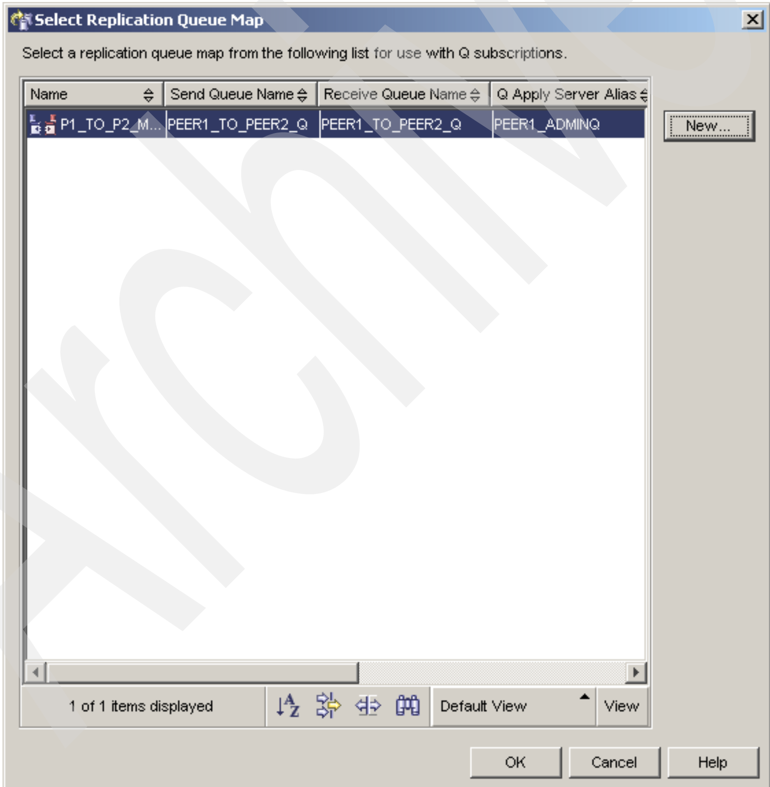


Figure 6-39 Select Replication Queue Map

The next step is to select the replication queue map for communication between the “Second server to first server.”

Repeat the steps performed earlier for the replication queue map definition from the “First server to second server” (as described in Figure 6-30 on page 599 through Figure 6-39) using the appropriate template values corresponding to node PEER2 on Azov. This is described in Figure 6-40 through Figure 6-47 on page 614.

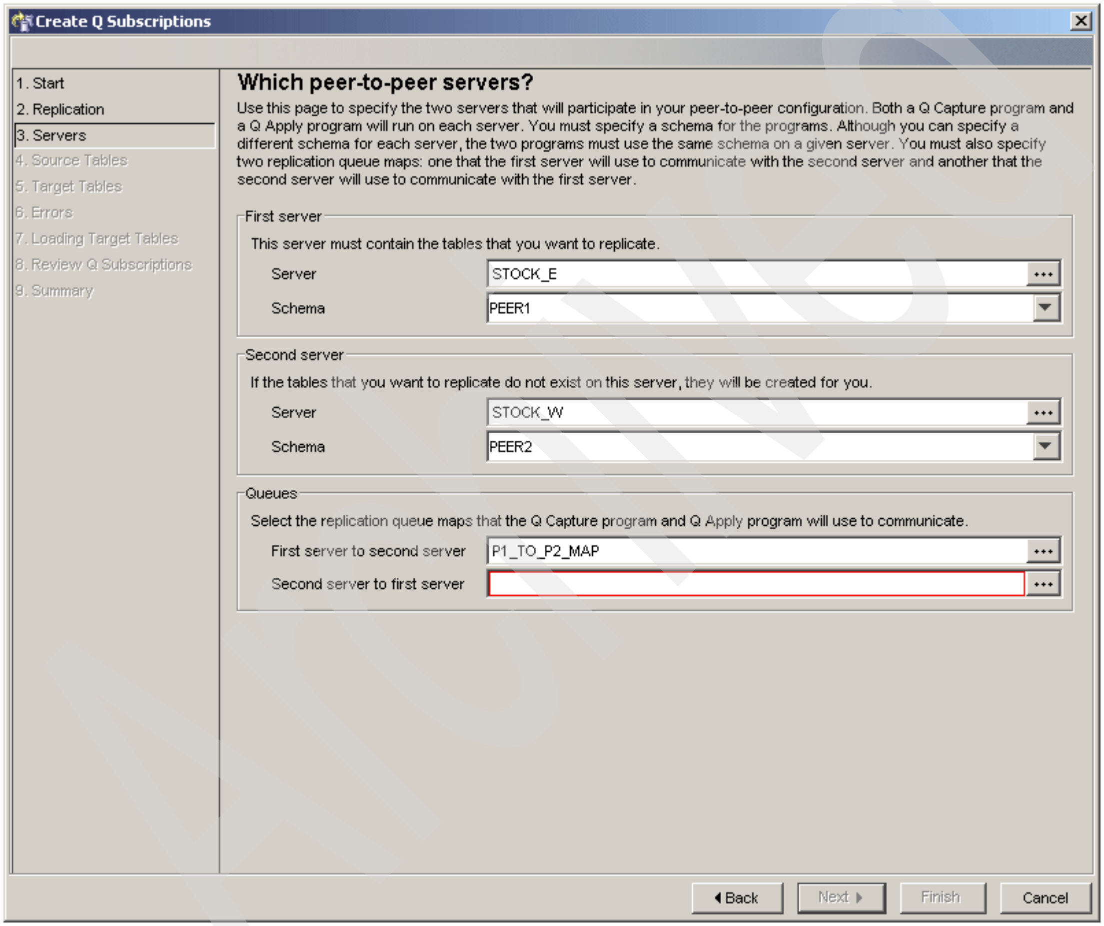


Figure 6-40 Which peer-to-peer servers

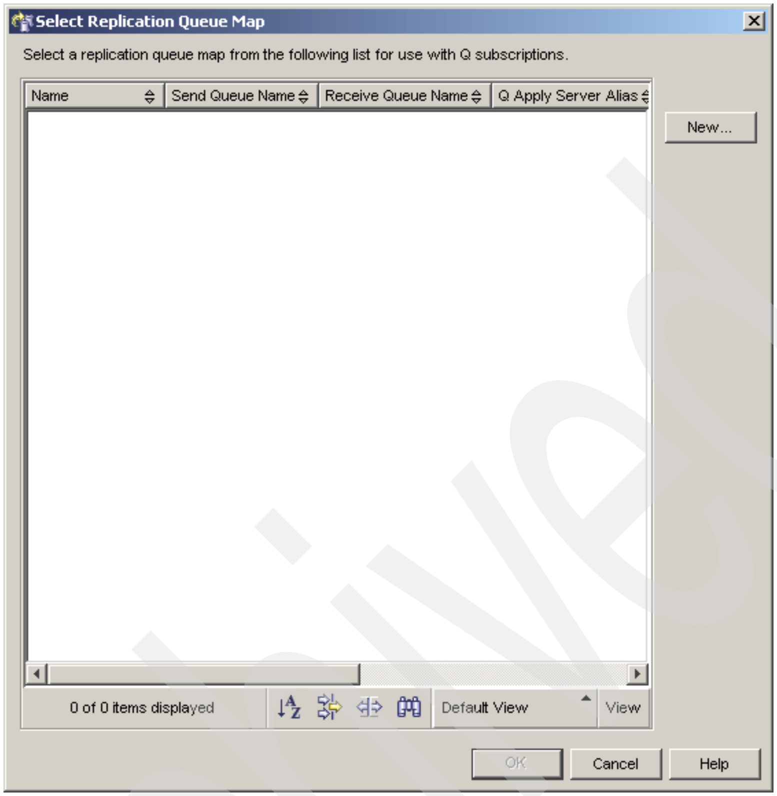


Figure 6-41 Select Replication Queue Map

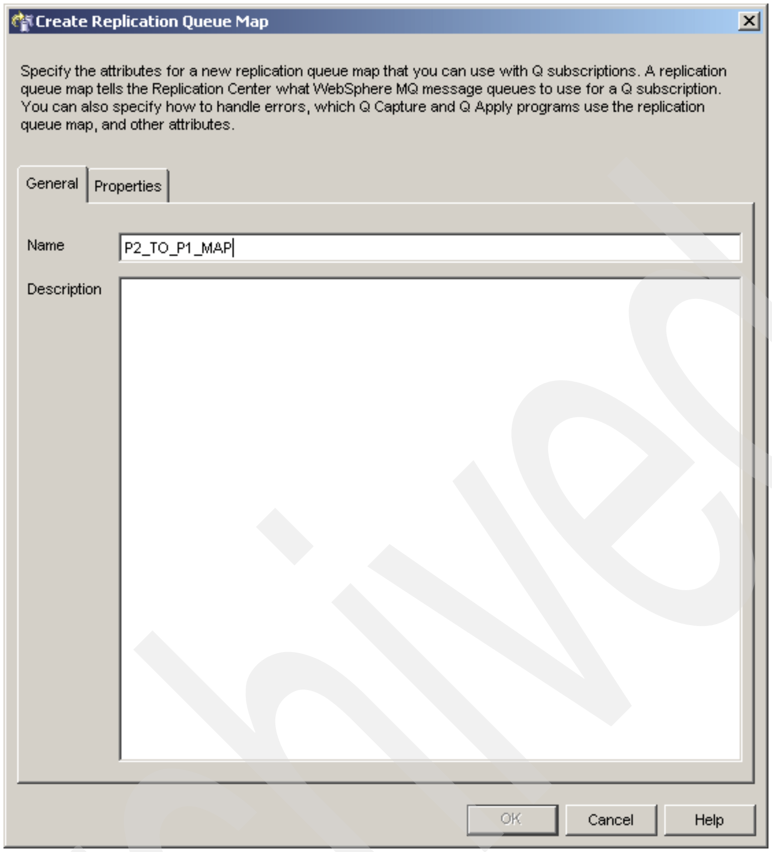


Figure 6-42 Create Replication Queue Map

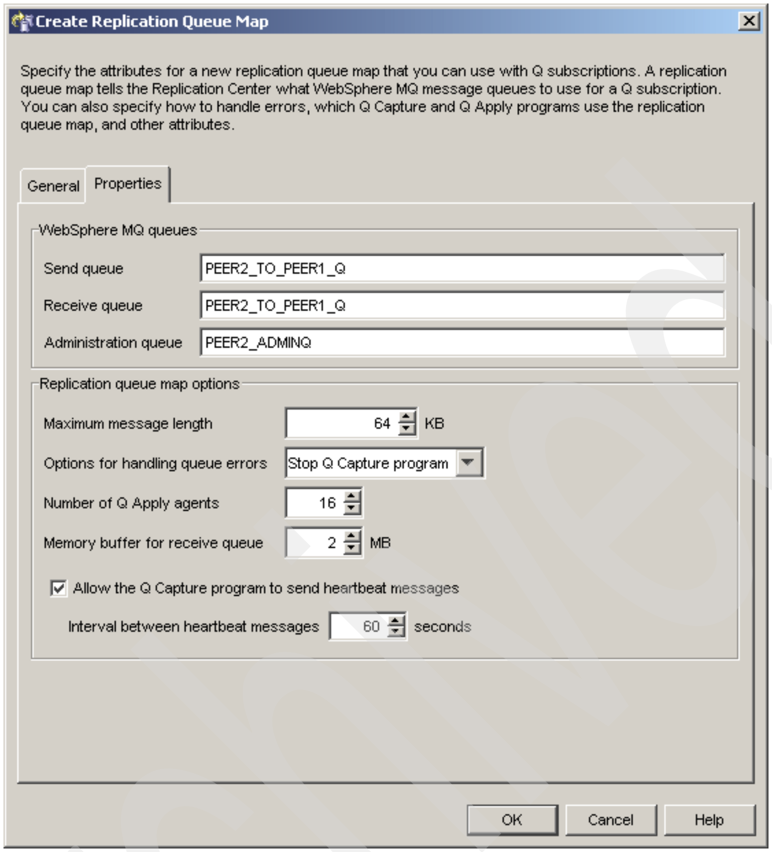


Figure 6-43 Create Replication Queue Map - Properties

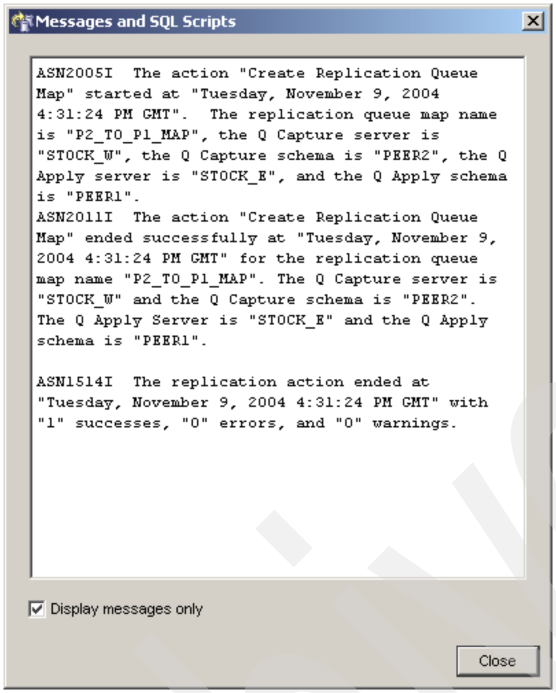


Figure 6-44 Messages and SQL Scrips

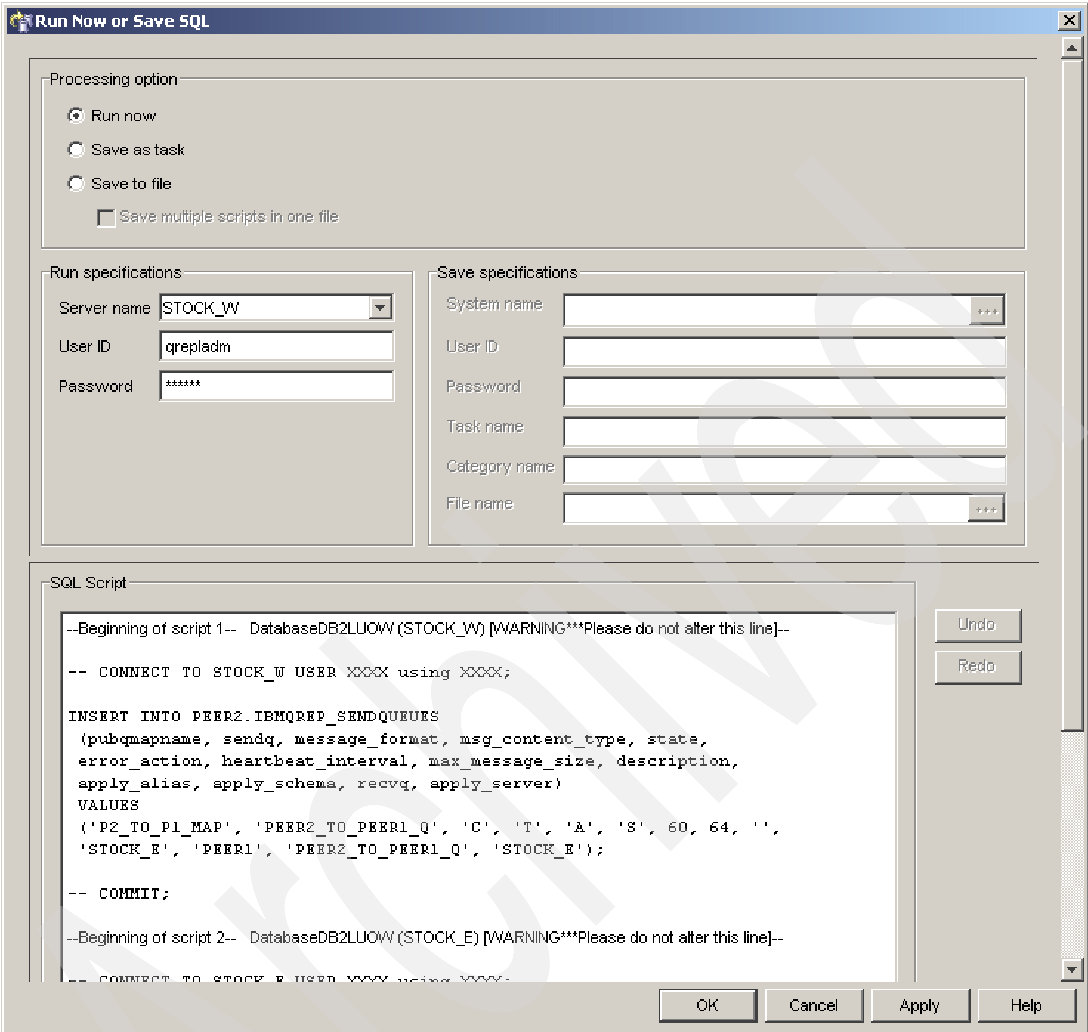


Figure 6-45 Run Now or Save SQL

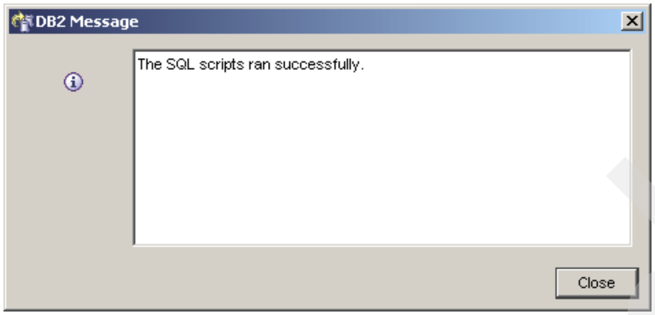


Figure 6-46 SQL scripts ran successfully message

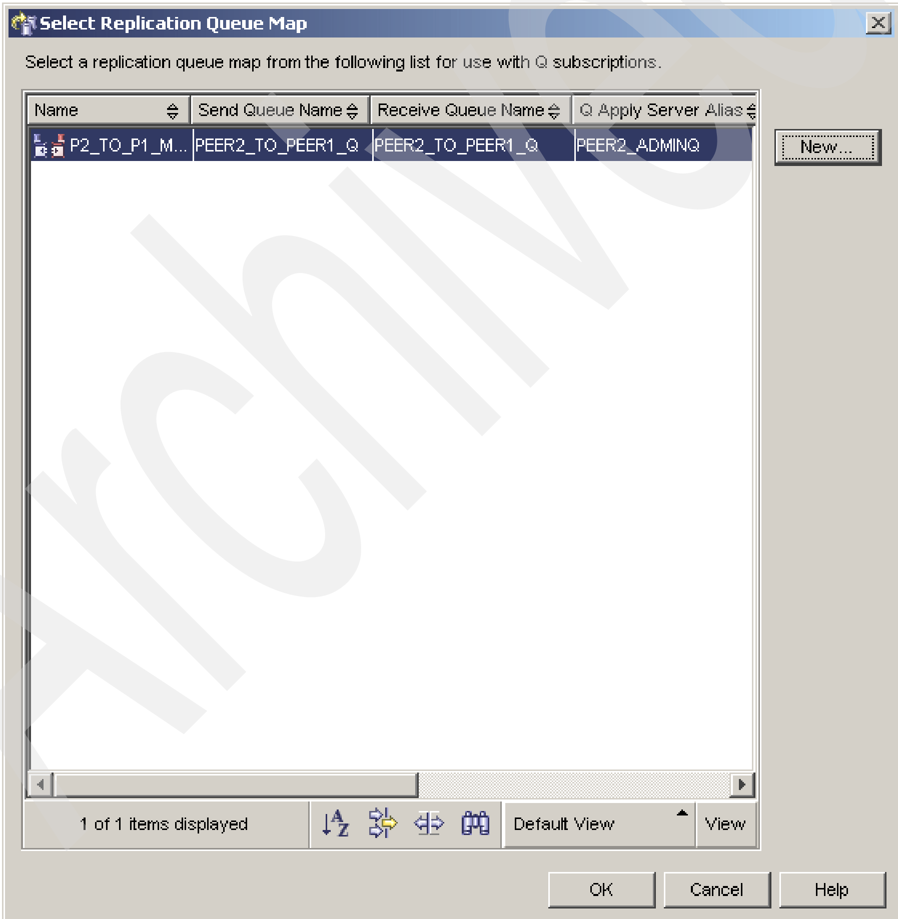


Figure 6-47 Select Replication Queue Map

At the completion of creating the replication queue maps for communication between the servers (see Figure 6-48), the individual subscriptions need to be defined.

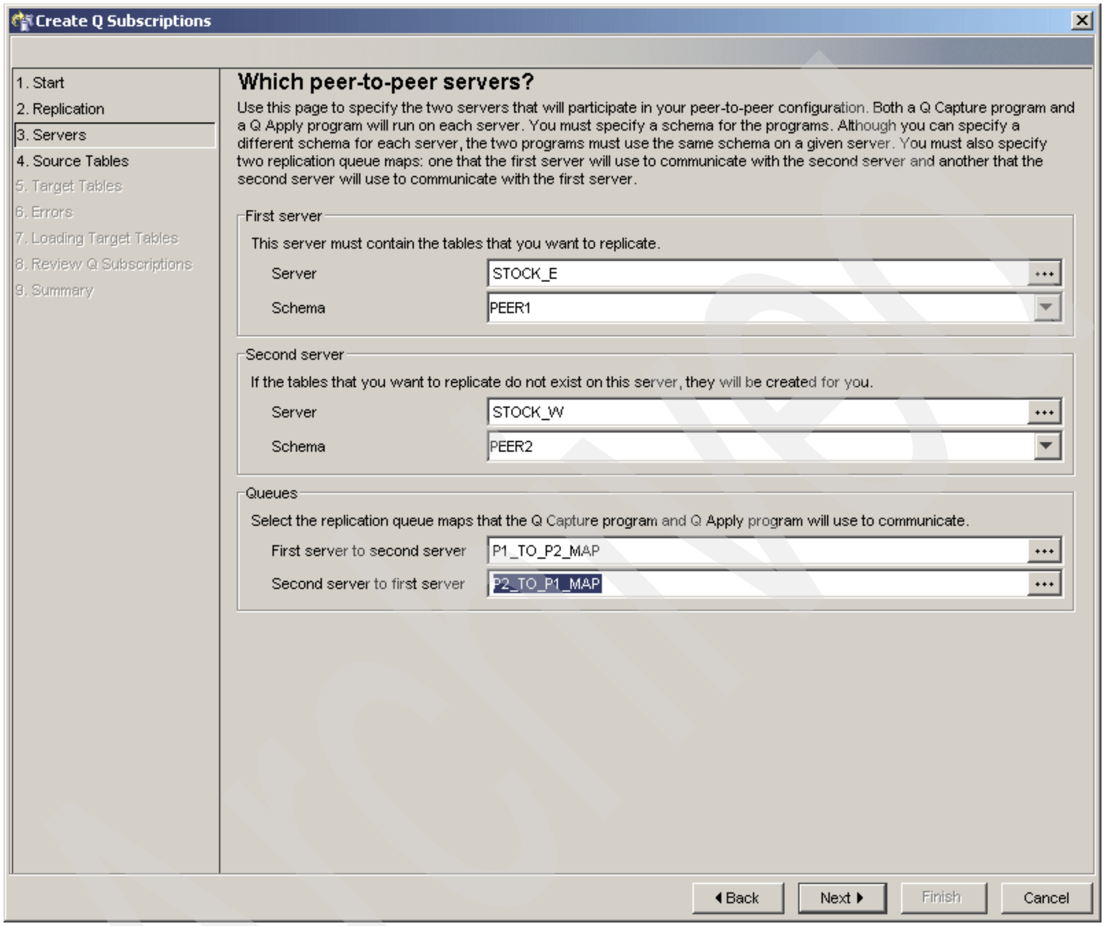


Figure 6-48 Completion of the replication queue maps selection

Click **Next** in Figure 6-48 to begin the process of defining the subscriptions as shown in Figure 6-49.

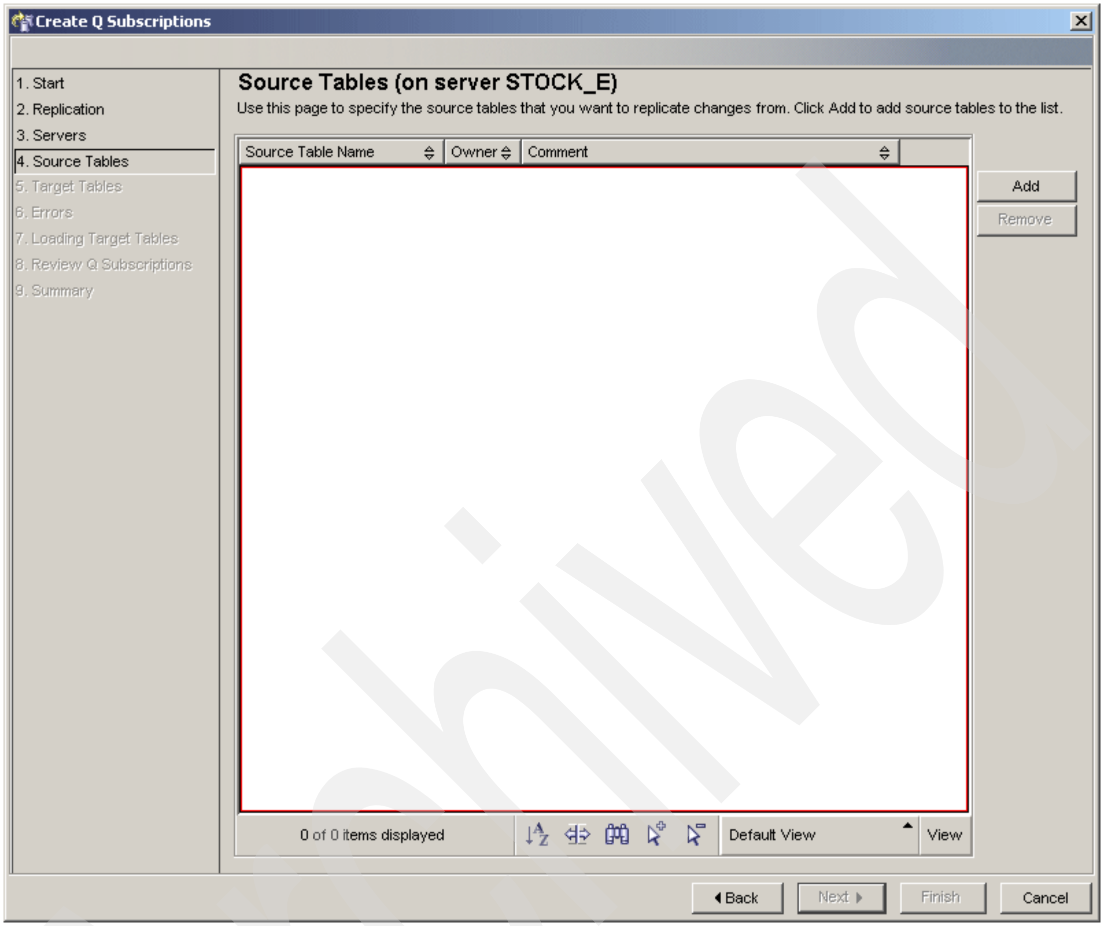


Figure 6-49 Create Q Subscriptions - Source Tables (on server STOCK_E)

Since there are no tables to select from, click the **Add** button in Figure 6-49 to add to the list of tables available for subscriptions, as shown in Figure 6-50.

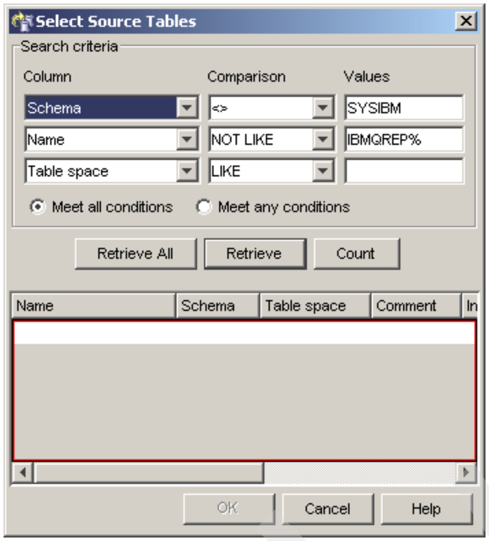


Figure 6-50 Select Source Tables

Exclude all tables that have a schema name of SYSIBM, and tables with names beginning with IBMQREP%. Click **Retrieve** to list all the tables that qualify, as shown in Figure 6-50. Clicking **Retrieve All** ignores the criteria.

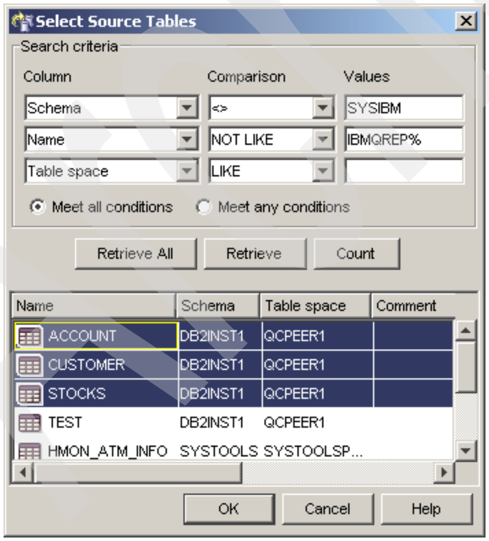


Figure 6-51 Select Source Tables

Select the **ACCOUNT**, **CUSTOMER**, and **STOCKS** tables and click **OK** to display Figure 6-51.

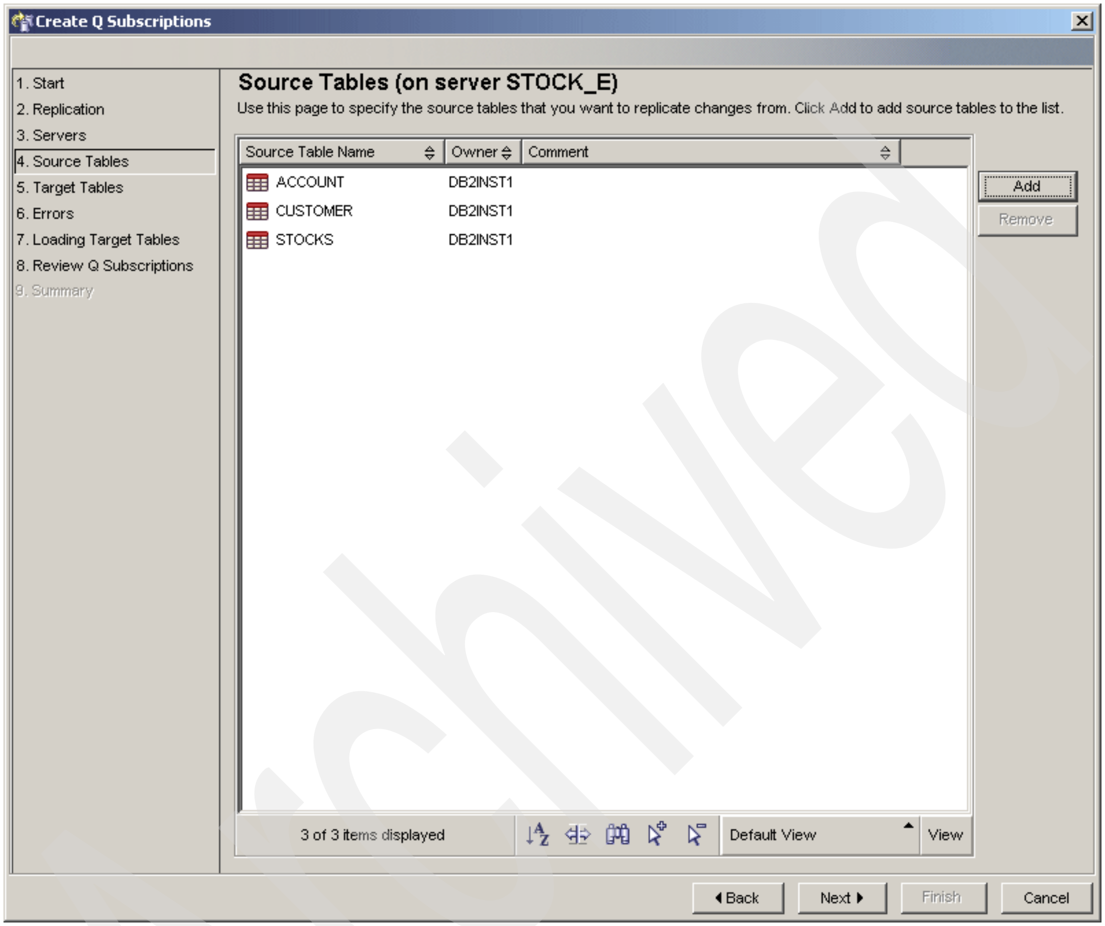


Figure 6-52 Source Tables

Click **Next** to set the profiles for the target tables, as shown in Figure 6-53.

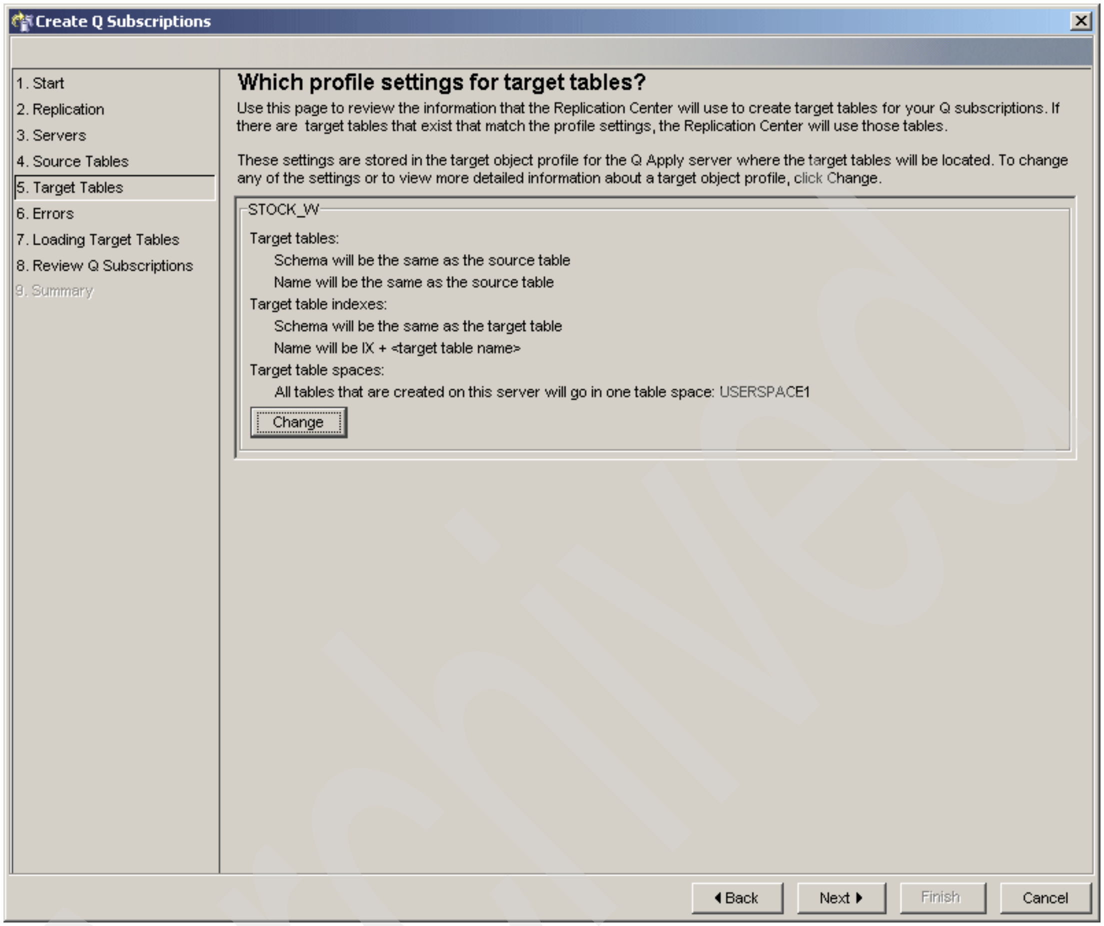


Figure 6-53 Create Q Subscriptions - Which profile settings for target tables

Figure 6-53 provides the option for changing the default settings for the target tables. We chose to go with the defaults. Click **Next** to specify the action to be taken in the event of errors, as shown in Figure 6-54.

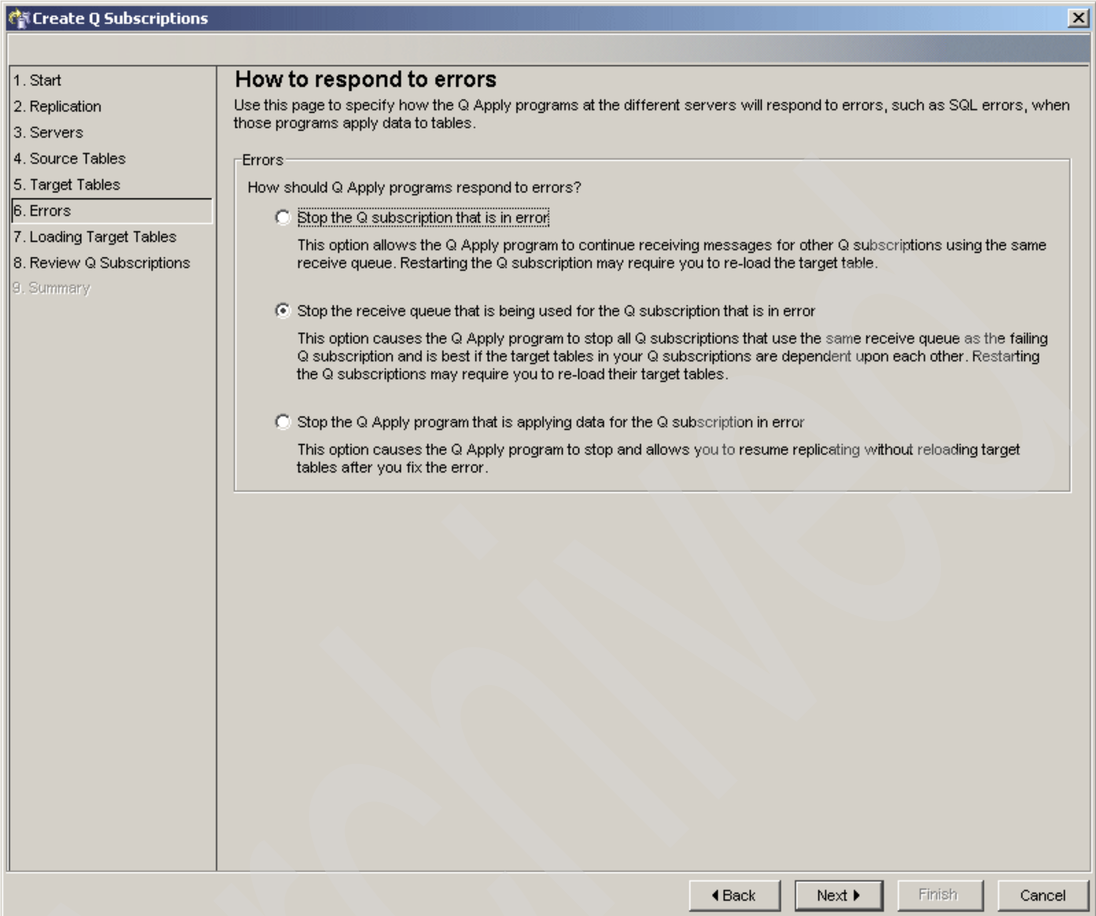


Figure 6-54 How to respond to errors

We selected the **Stop the receive queue that is being used by the Q Subscription in error** setting. Click **Next** to specify how the target tables should be loaded in Figure 6-55.

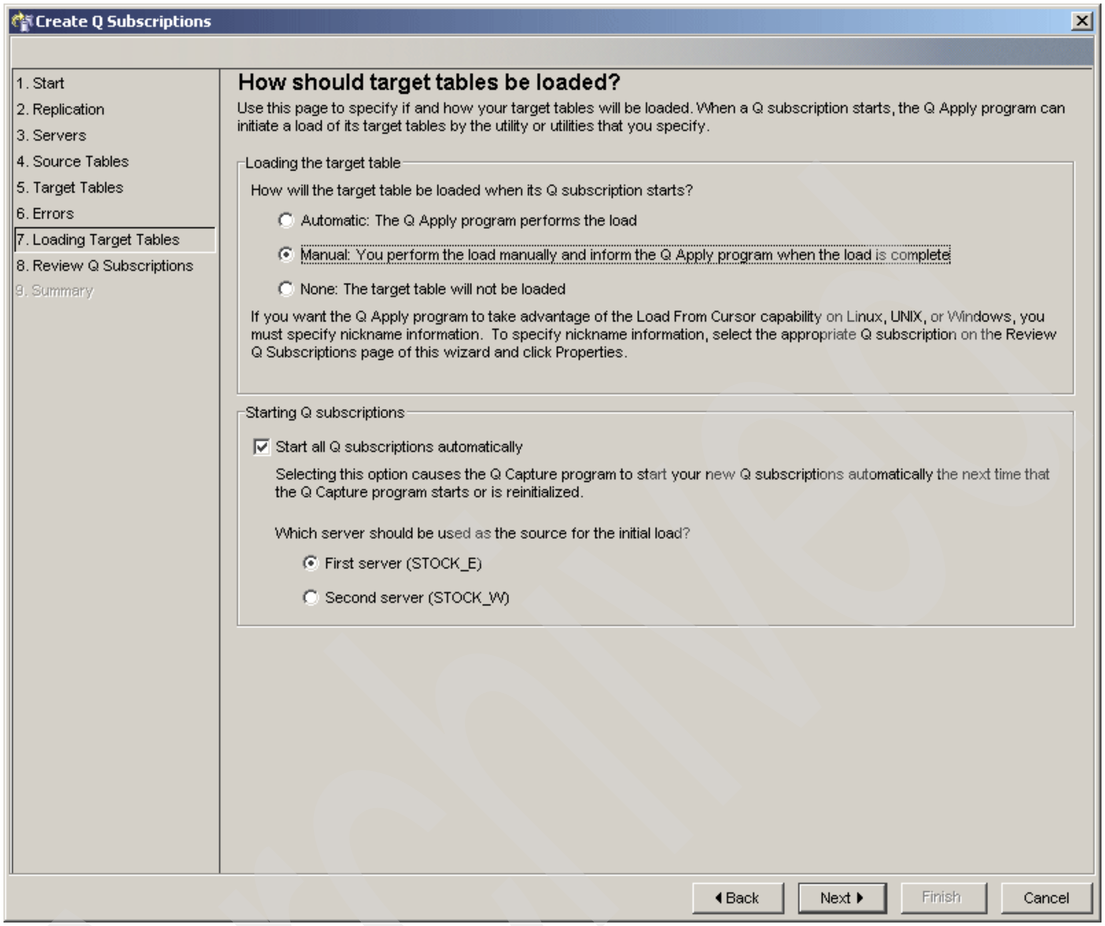


Figure 6-55 How should target tables be loaded

We chose the **Manual** option for this set of tables, indicating that we would be responsible for synchronizing the data in the target table with that of the source tables and sending a LOADDONE signal to the source Q Capture program when the load was completed. We selected the **Start all Q subscriptions automatically** option target tables. We also selected STOCK_E to be the source for the initial load, as shown in Figure 6-55.

Click **Next** to review the subscriptions defined, as shown in Figure 6-56.

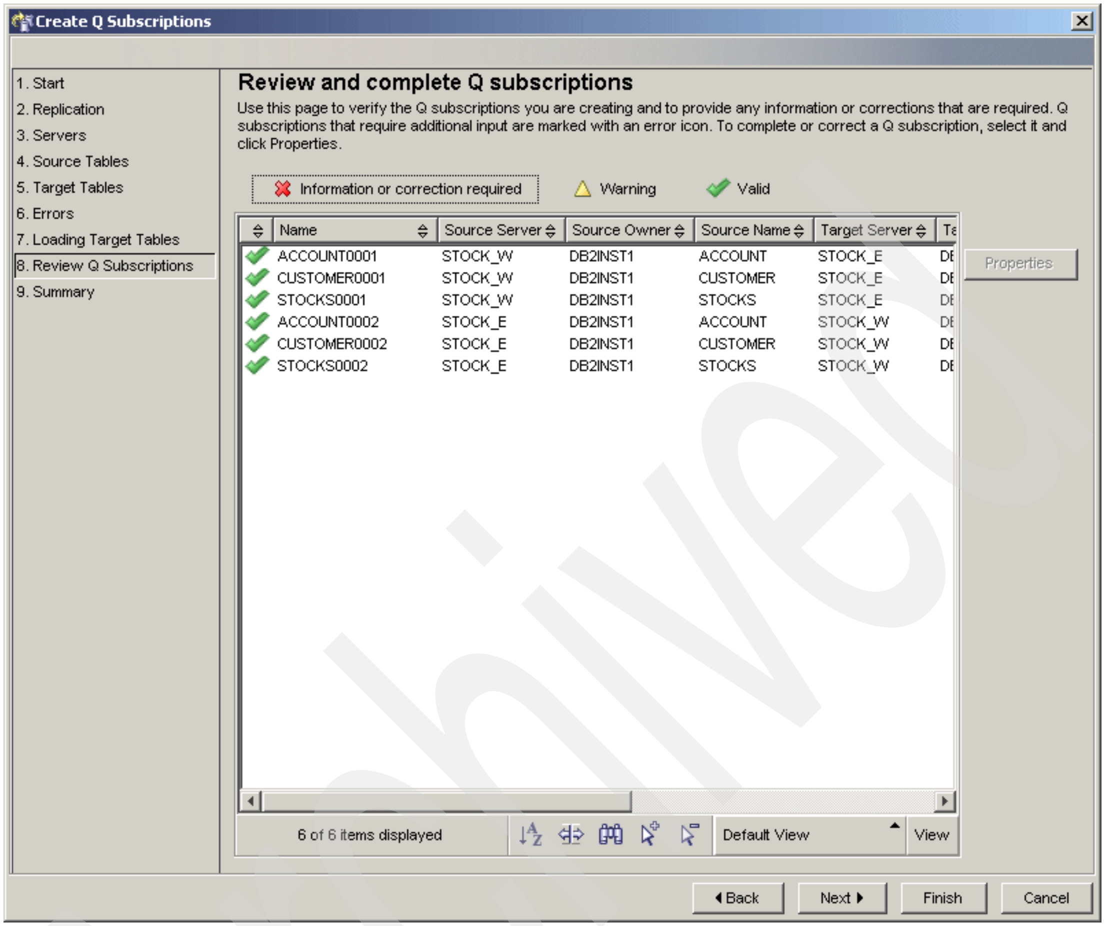


Figure 6-56 Review and complete Q subscriptions

Figure 6-56 shows six subscriptions in all—one for each of the three tables on each server. Click **Next** to complete the definition of the subscriptions and view the summary, as shown in Figure 6-57.

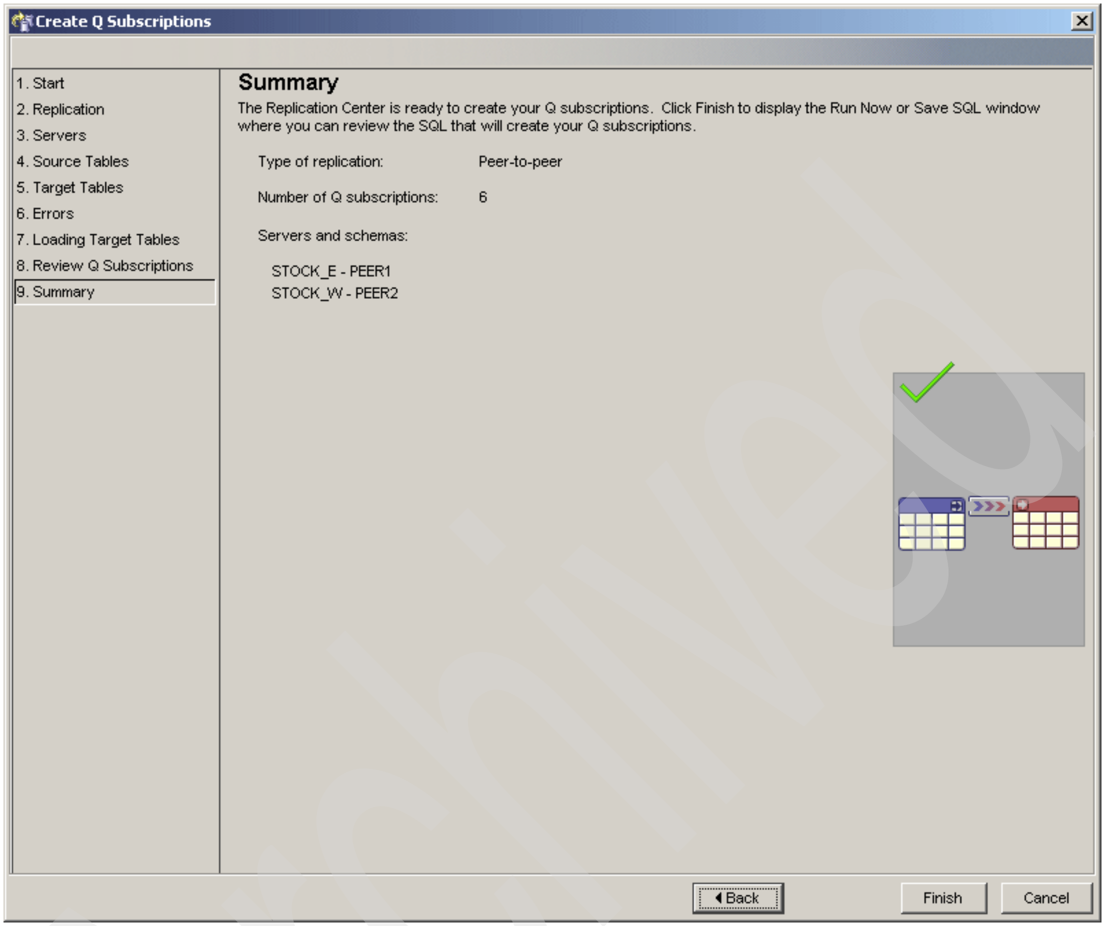


Figure 6-57 Create Q Subscriptions - Summary

Review the summary in Figure 6-57 and click **Finish** to generate the scripts for creating the various objects on the two servers, as shown in Figure 6-58.

Click the **Back** button to go back and correct any errors.

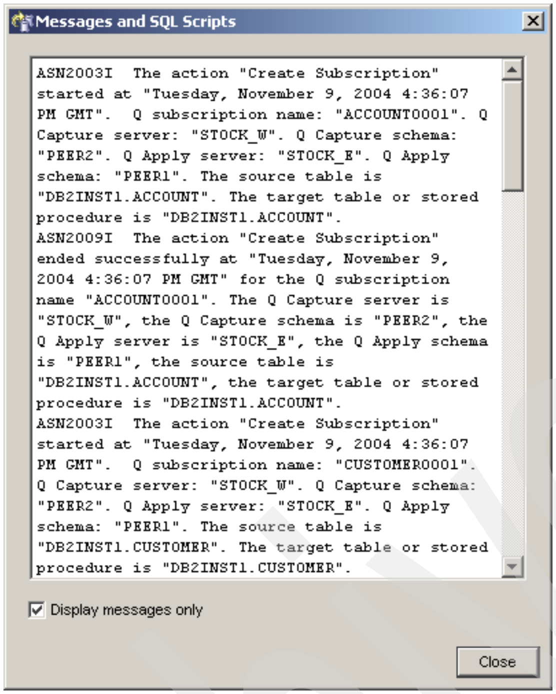


Figure 6-58 Messages and SQL Scripts

Click **Close** to decide whether to run the scripts right away or save them for later execution, as shown in Figure 6-59.

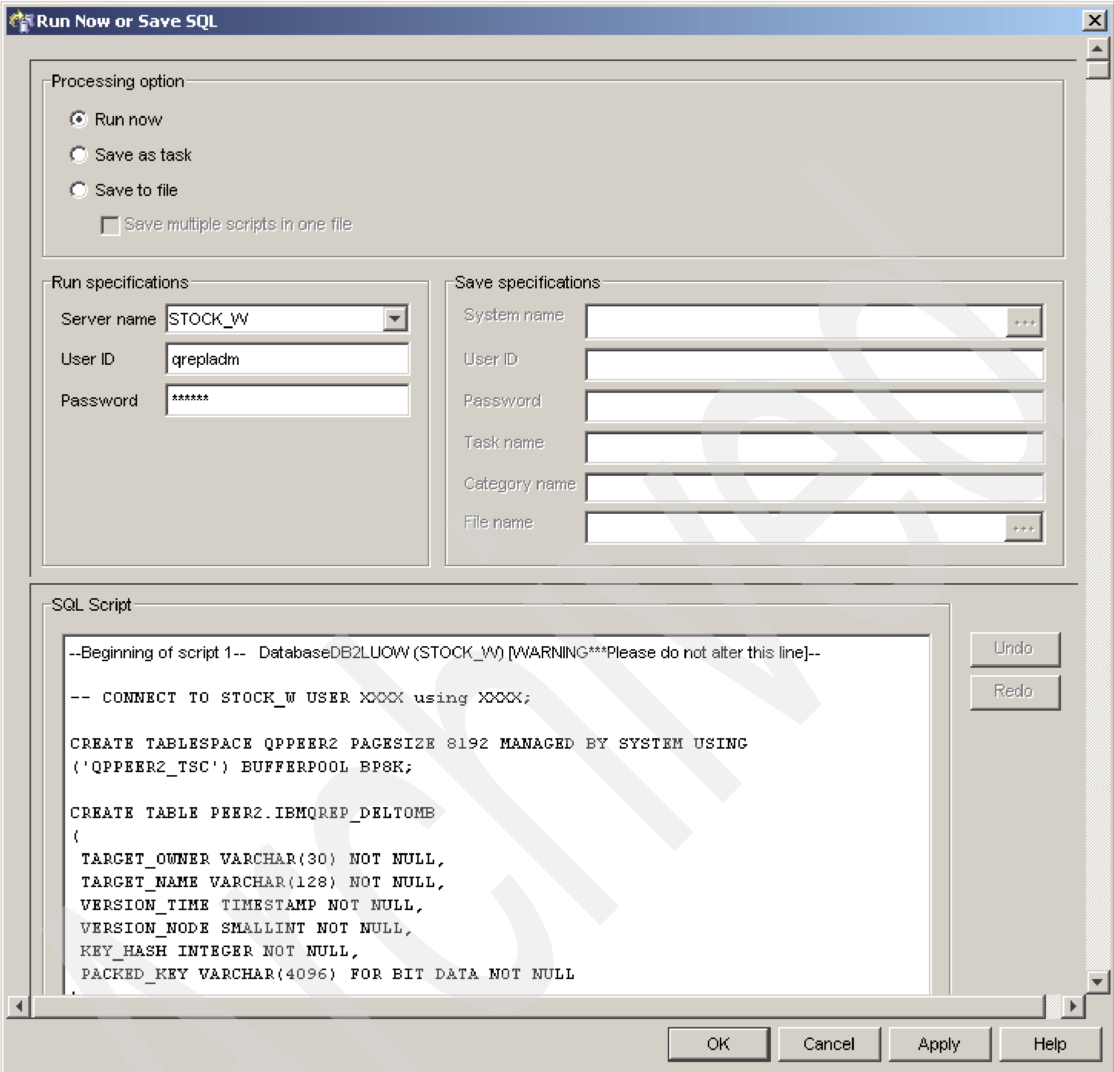


Figure 6-59 Run Now or Save SQL

We selected **Run now** and clicked **OK** to execute the scripts immediately. Figure 6-60 shows that the scripts ran successfully.

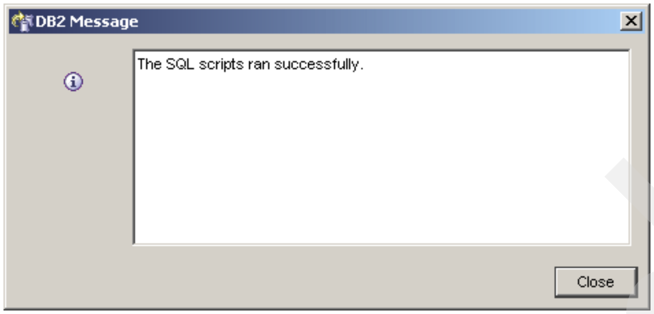


Figure 6-60 SQL scripts ran successfully message

With the completion of the subscriptions, the Q Capture program needs to be started on both servers.

Attention: Q Capture should be started before Q Apply to ensure that all messages needed for activating subscriptions are processed correctly.

Step 6f: Start the Q Capture on both servers

Click **Start Q Capture Program** in Figure 6-61 to start Q Capture on the appropriate servers.

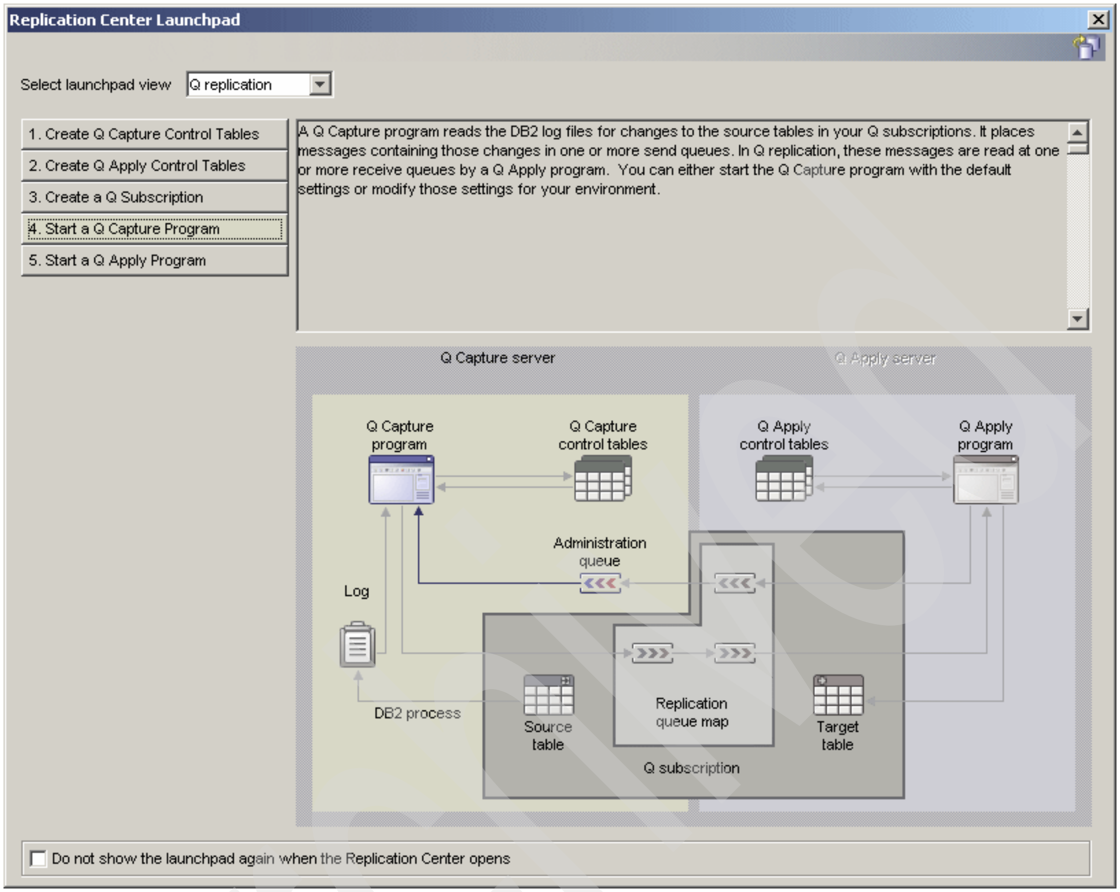


Figure 6-61   Five steps in setting up Q replication infrastructure

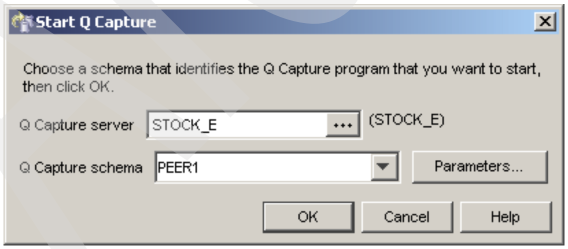


Figure 6-62   Start Q Capture

Specify the Q Capture server STOCK_E and Q Capture schema PEER1 and click **Parameters...** in Figure 6-62 to specify the Q Capture startup parameters in Figure 6-63.

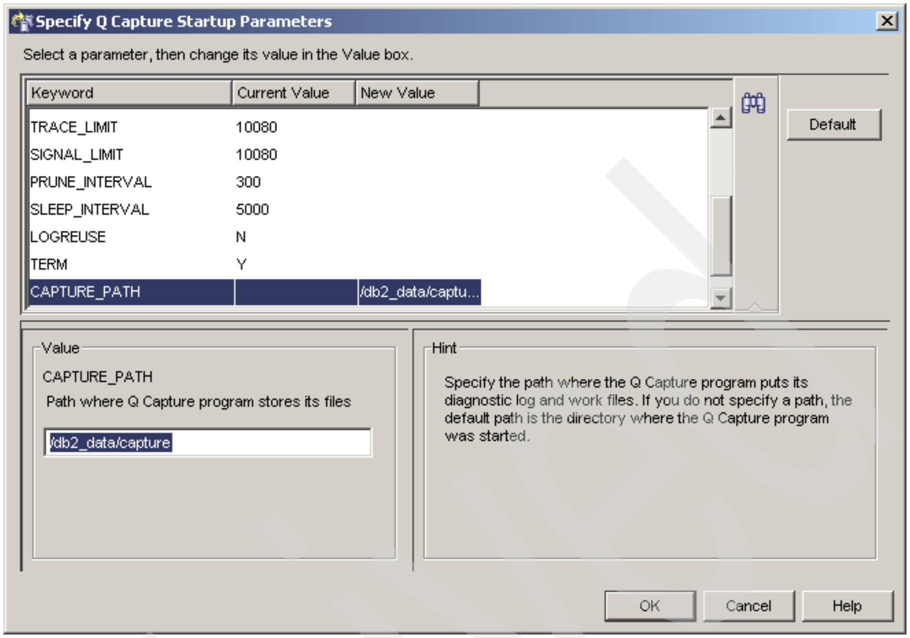


Figure 6-63 Specify Q Capture Startup Parameters

We chose the defaults for all parameters, and specified the CAPTURE_PATH parameter to be /db2_data/capture (reference A.24 in Table 6-3 on page 552). Click **OK** to generate the command for immediate execution or to be saved for later execution, as shown in Figure 6-64.

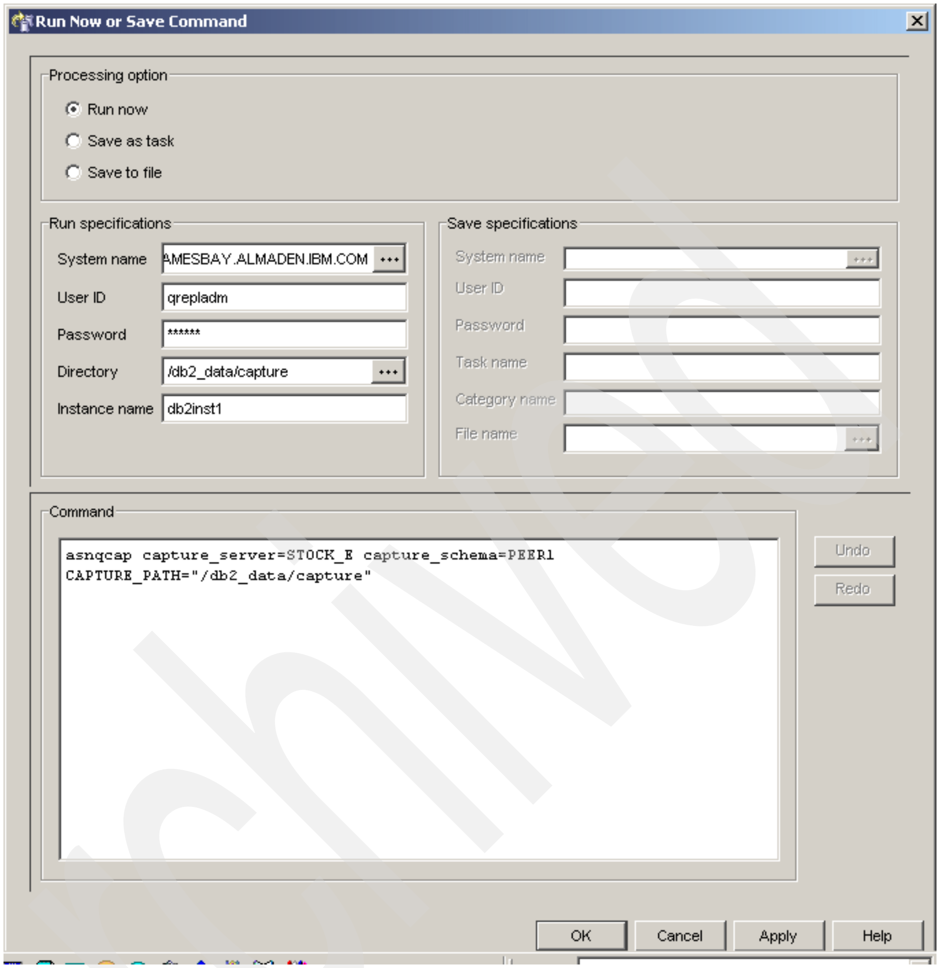


Figure 6-64 Run Now or Save Command

We selected **Run now** and clicked **OK** for the command to be executed immediately. Figure 6-65 displays the message indicating that the command was submitted.

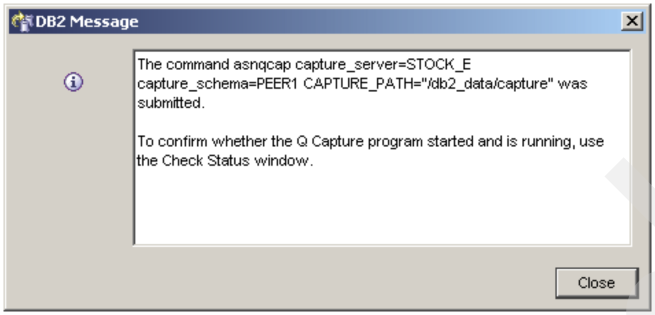


Figure 6-65 Command submitted message

> **Note:** We checked the status of the Q Capture program as described in "Step 6h: Verify status of Q Capture and Q Apply processes" on page 642.

Click **Close** to continue with the starting of Q Capture on the second server.

Repeat the steps described in Figure 6-61 on page 627 through Figure 6-65 for the second server, as shown in Figure 6-66 through Figure 6-70 on page 634, using the appropriate reference IDs for the second server (Azov) corresponding to those of the first server (Jamesbay).

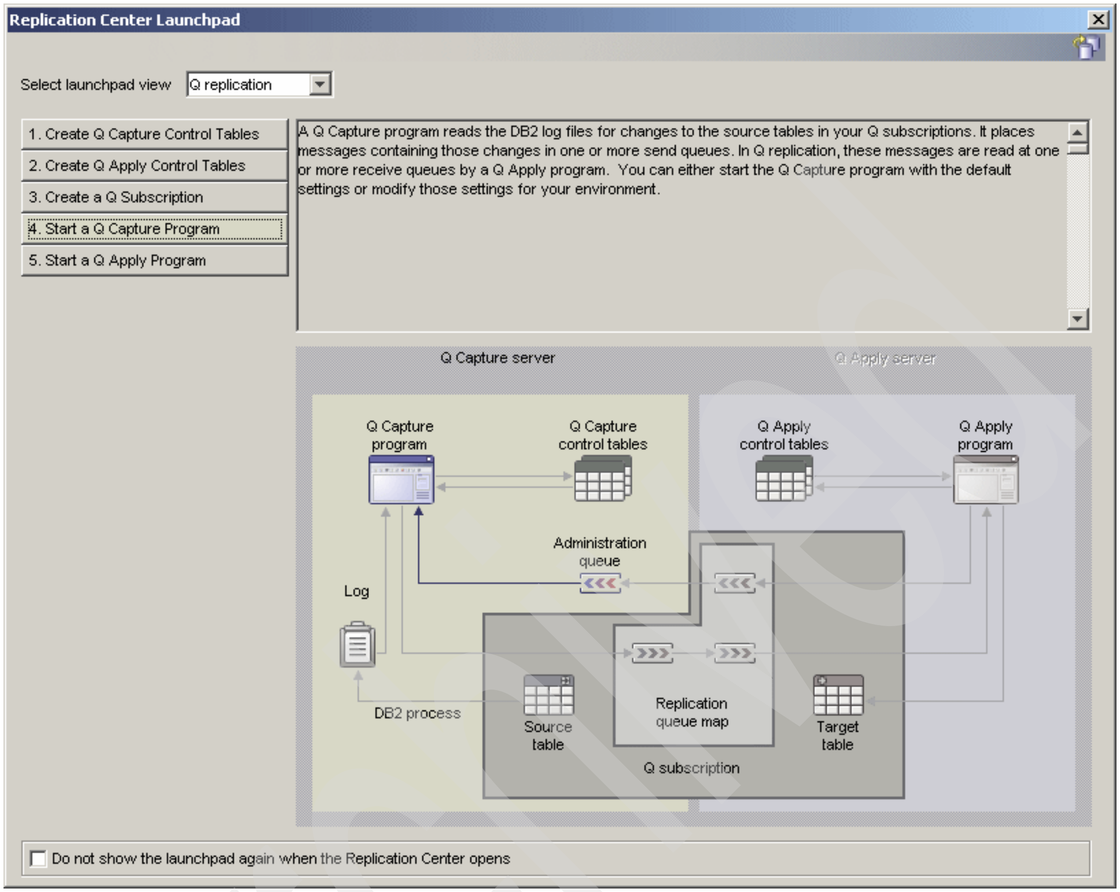


Figure 6-66 Five steps to setting up Q replication infrastructure

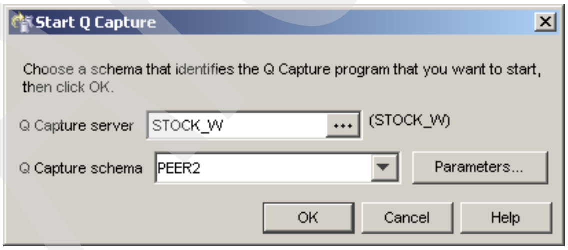


Figure 6-67 Start Q Capture

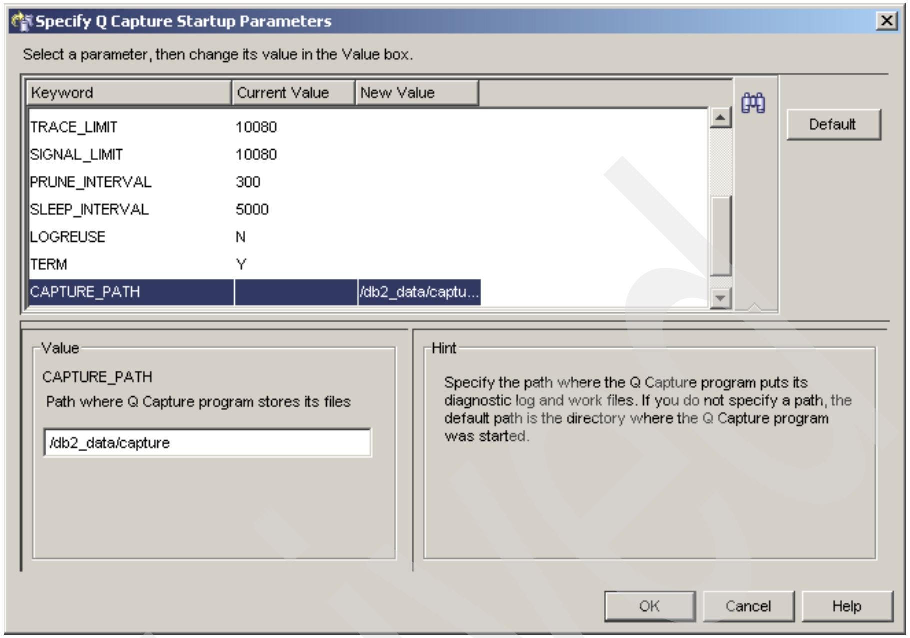


Figure 6-68 Specify Q Capture Startup Parameters

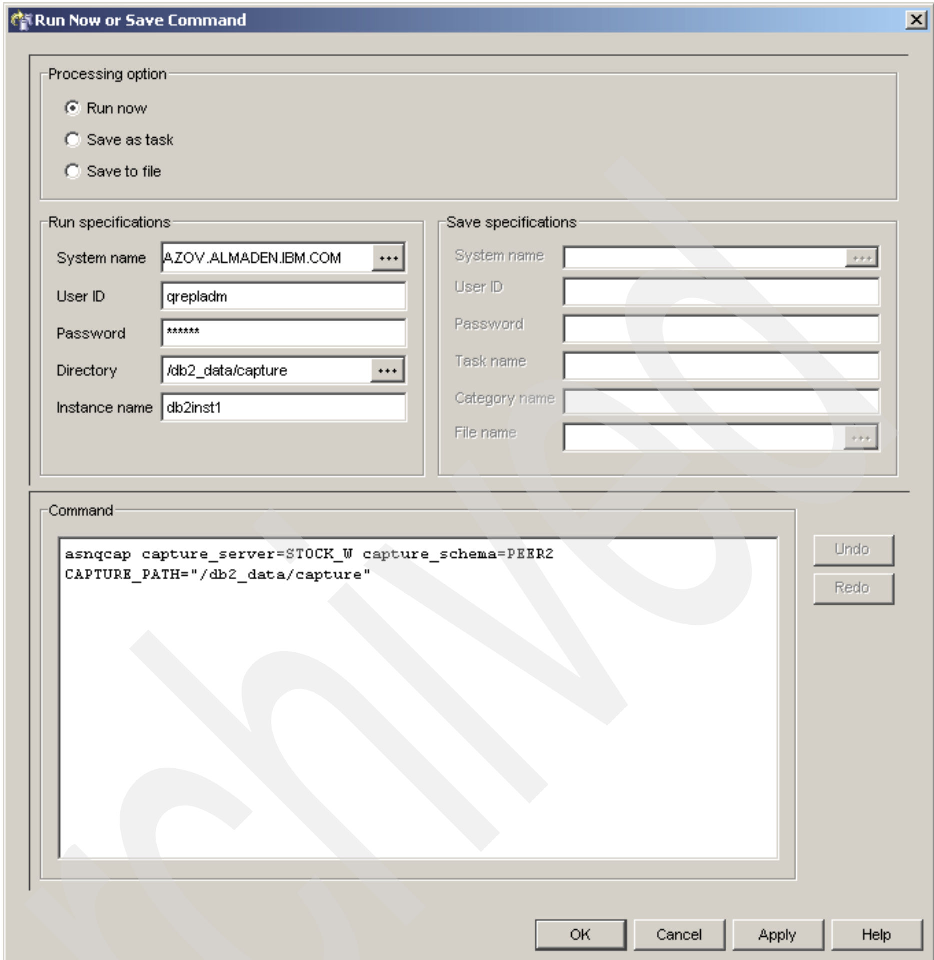


Figure 6-69 Run Now or Save Command

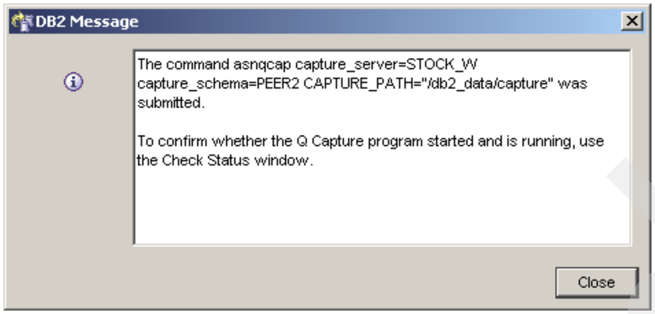


Figure 6-70 Command submitted message

The Q Apply program next needs to be started on both servers.

Step 6g: Start the Q Apply on both servers

Click **Start Q Apply Program** in Figure 6-71 to start Q Apply on the appropriate servers.

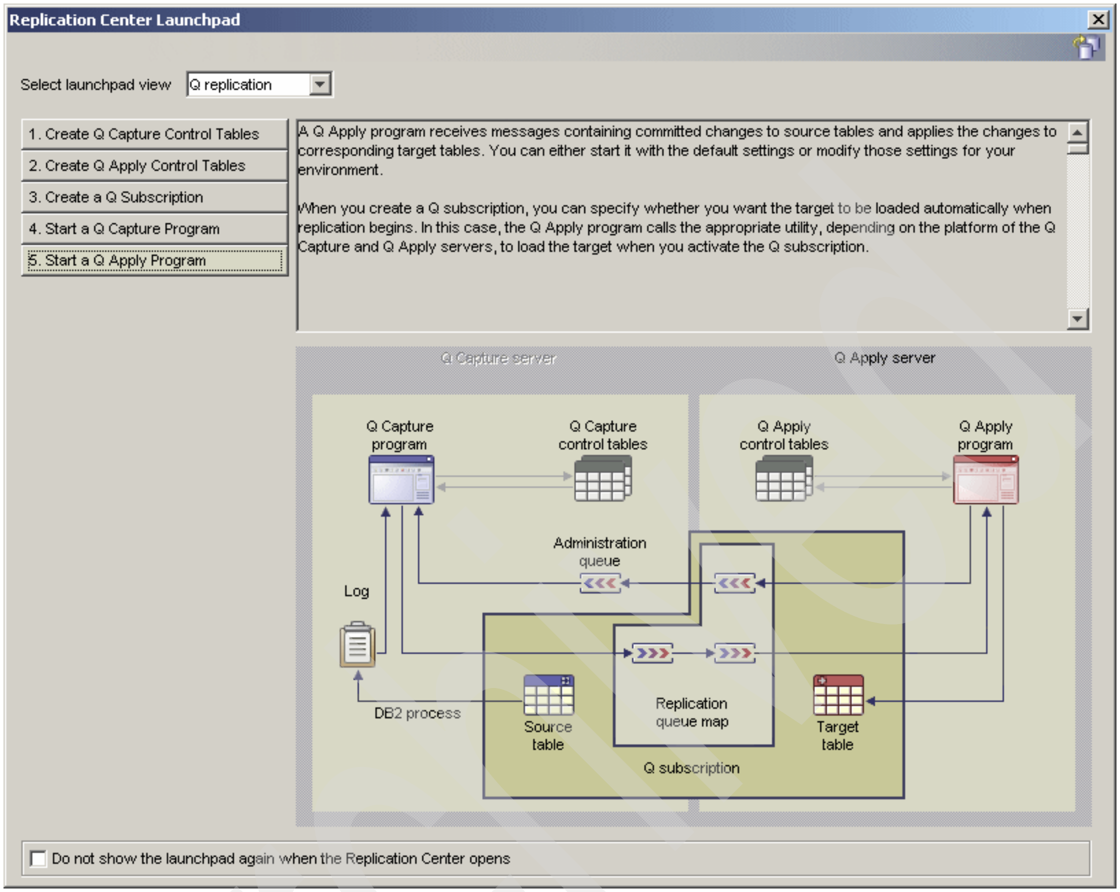


Figure 6-71 Five steps in setting up Q replication infrastructure

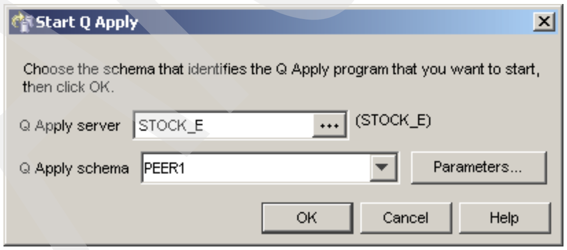


Figure 6-72 Start Q Apply

Specify the Q Apply server STOCK_W and Q Apply schema PEER2 and click **Parameters...** in Figure 6-72 to specify the Q Apply startup parameters in Figure 6-73.

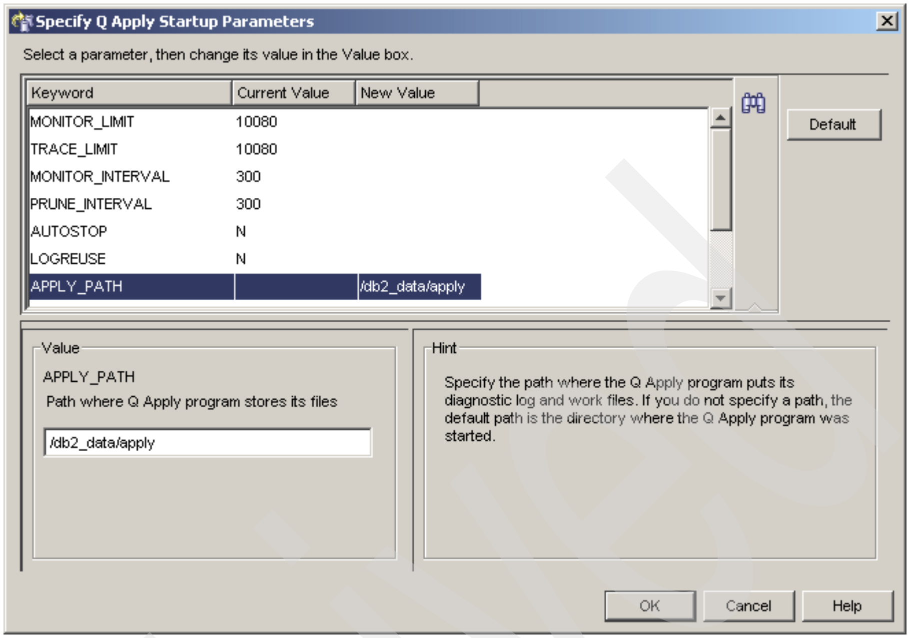


Figure 6-73   Specify Q Apply Startup Parameters

We chose the defaults for all parameters, and specified the APPLY_PATH parameter to be /db2_data/apply (reference A.25 in Table 6-3 on page 552). Click **OK** to generate the command for immediate execution or to be saved for later execution, as shown in Figure 6-74.

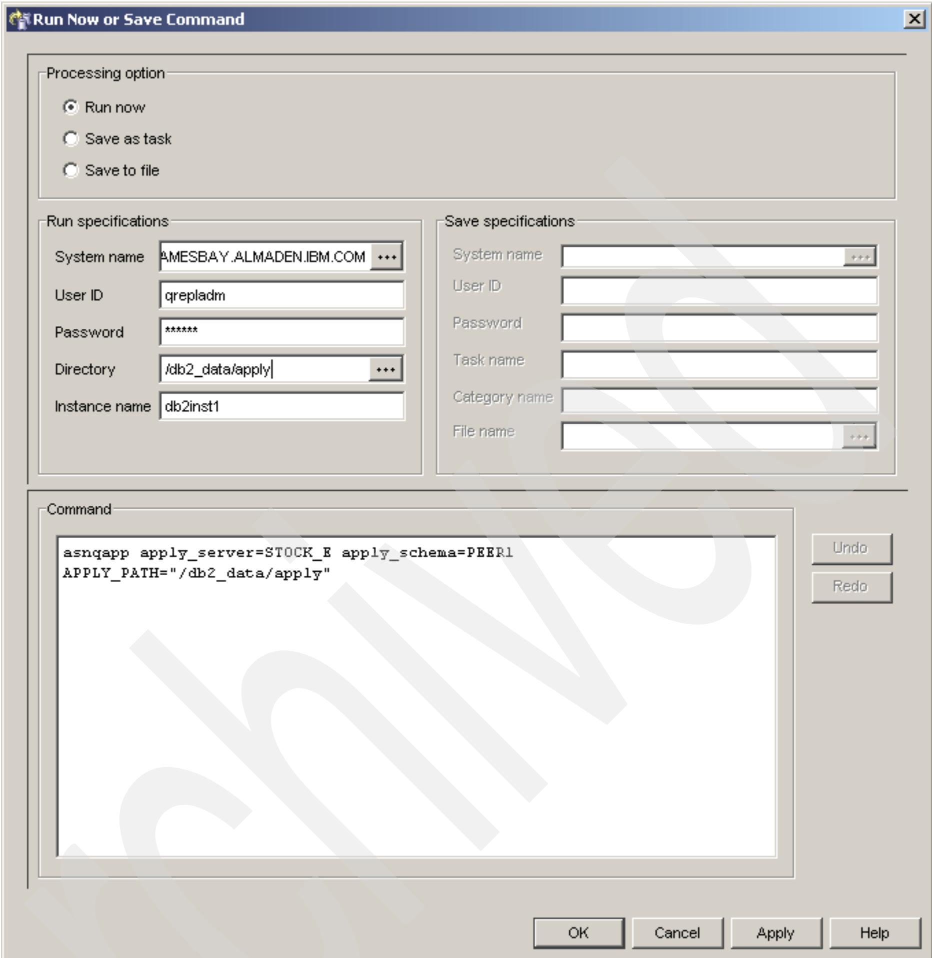


Figure 6-74 Run Now or Save Command

We selected **Run now** and clicked **OK** for the command to be executed immediately. Figure 6-75 displays the message indicating that the command was submitted.

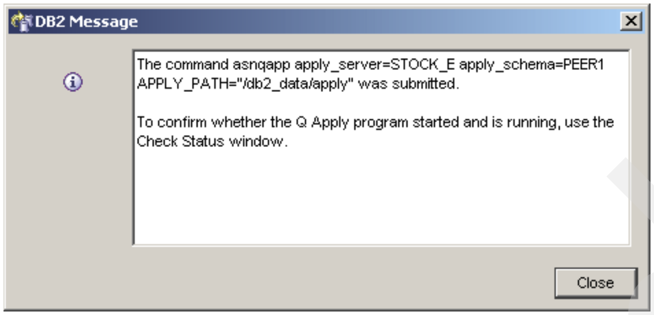


Figure 6-75 Command submitted message

> **Note:** We checked the status of the Q Apply program as described in "Step 6h: Verify status of Q Capture and Q Apply processes" on page 642.

Click **Close** to continue with the starting of Q Apply on the second server.

Repeat the steps described in Figure 6-71 on page 635 through Figure 6-75 for the second server, as shown in Figure 6-76 on page 639 through Figure 6-80 on page 642, using the appropriate reference IDs for the second server (Azov) corresponding to those of the first server (Jamesbay).

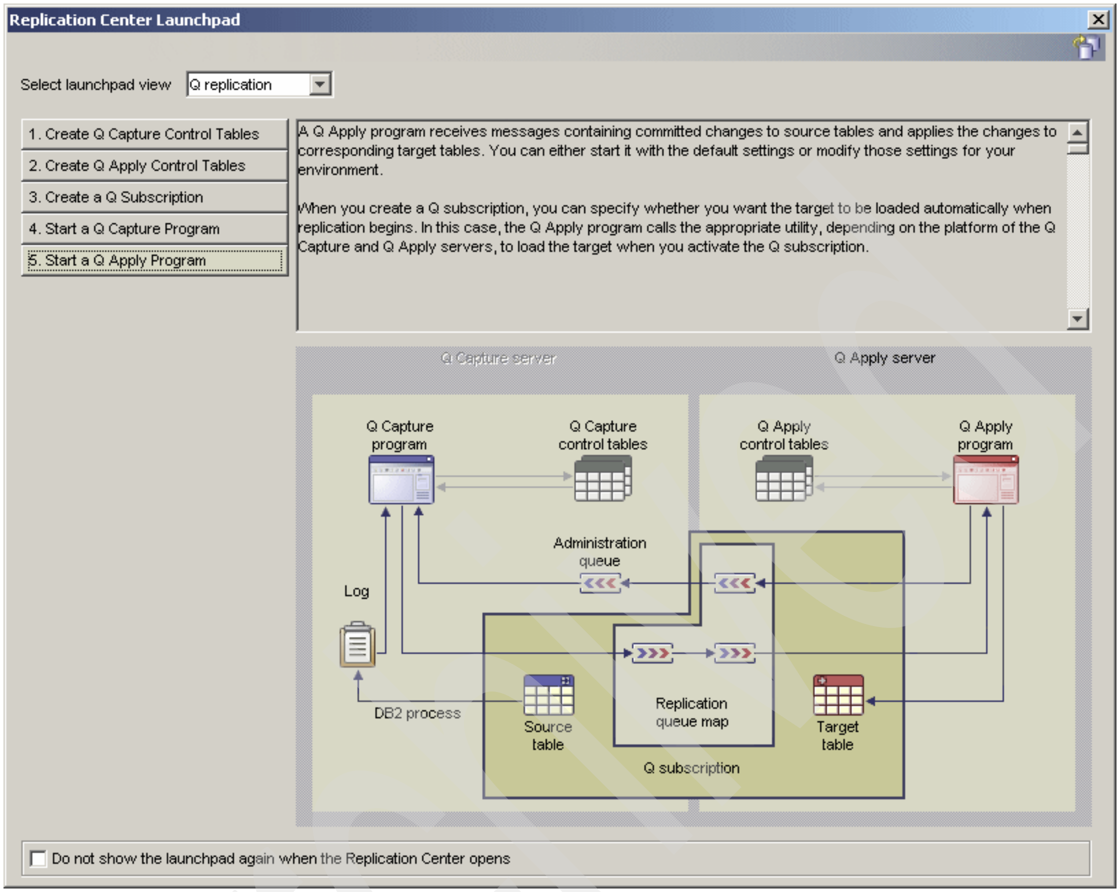


Figure 6-76 Five steps to setting up Q replication infrastructure

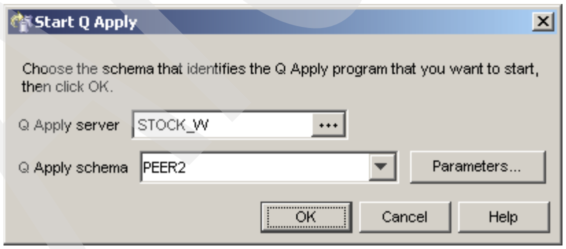


Figure 6-77 Start Q Apply

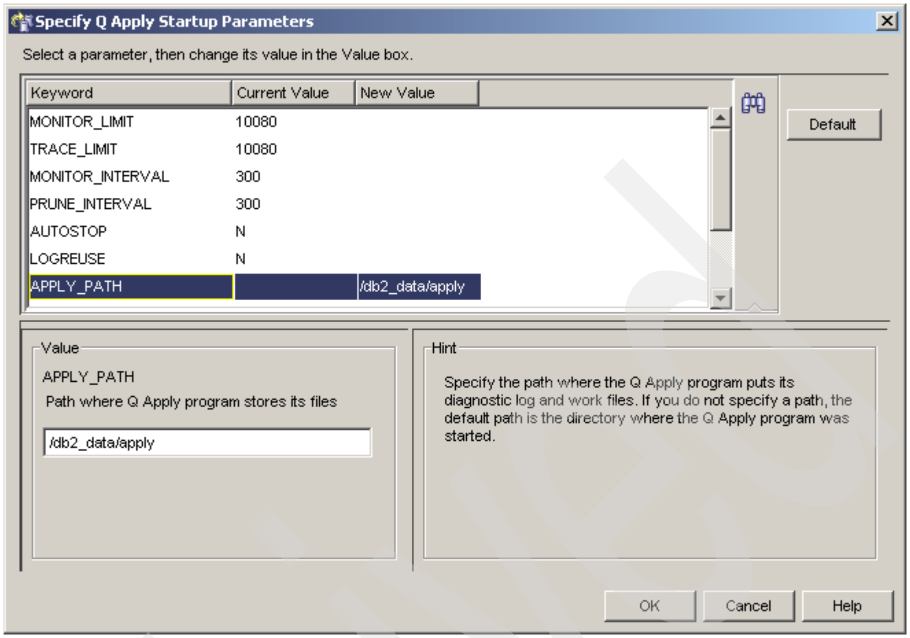


Figure 6-78 Specify Q Apply Startup Parameters

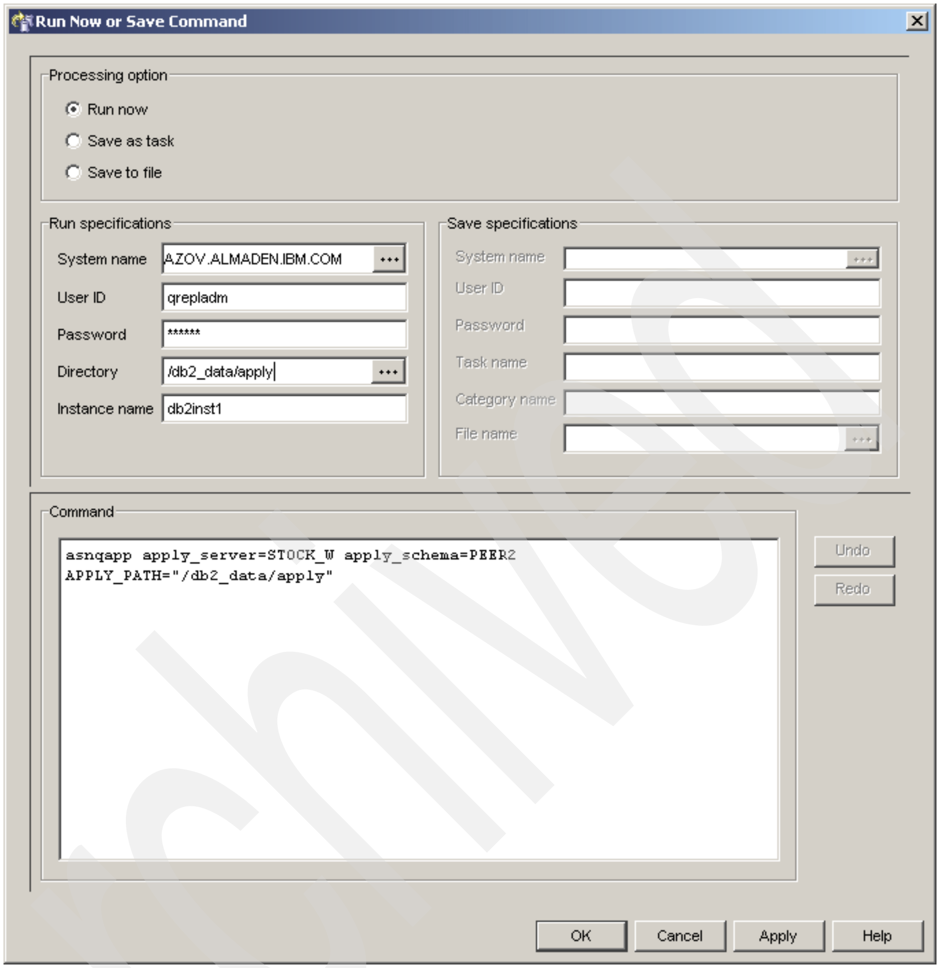


Figure 6-79 Run Now or Save Command

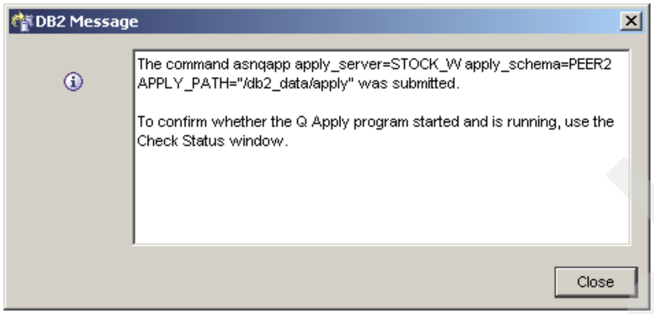


Figure 6-80 Command submitted message

Step 6h: Verify status of Q Capture and Q Apply processes

Here we review the status of the Q Capture and Q Apply processes started earlier.

A quick way of determining whether the Q Capture and Q Apply processes are running on a server is to issue the AIX **ps** command, as shown in Example 6-23.

Example 6-23 Checking for Q Capture and Q Apply processes

```
$ ps -ef | grep asn
qrepladm 49936    1   0   Nov 17      -  3:22 asnqcap capture_server=STOCK_W capture_schema=PEER2
qrepladm 51964    1   0   Nov 17      -  1:58 asnqapp apply_server=STOCK_W apply_schema=PEER2
```

The recommended approach is to select the **Q Capture Servers** folder in the navigation pane of the Replication Center, as shown in Figure 6-81. This lists four servers (EAST_DB and STOCK_E on Jamesbay, and WEST_DB and STOCK_W on Azov, as the four Q Capture Servers in the right pane).

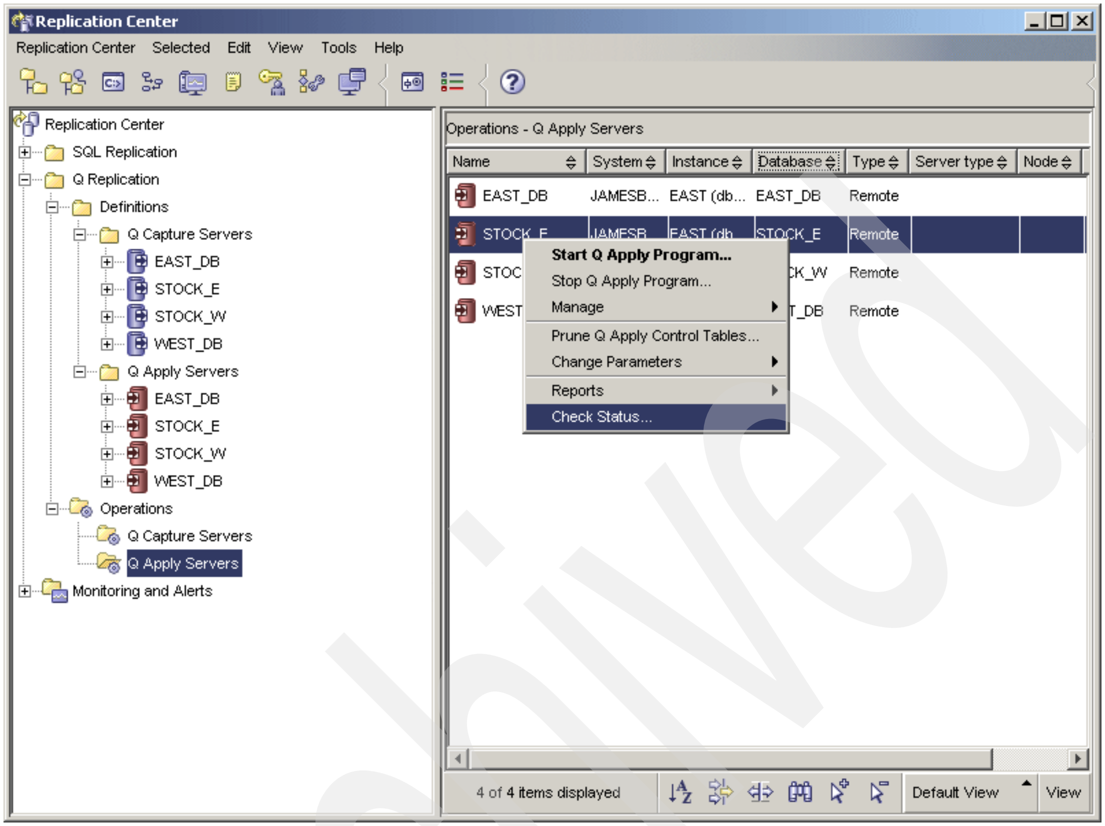


Figure 6-81   Check Status of the STOCK_W Q Capture server

Select the row with STOCK_E in the Name column and right-click to obtain a drop-down list. Select **Check Status...** (as shown in Figure 6-81), which displays Figure 6-82.

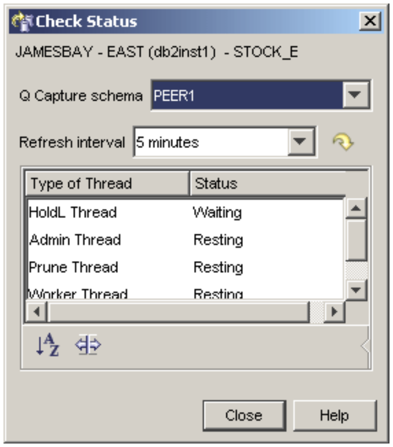


Figure 6-82 Check Status of STOCK_E Q Capture threads

Figure 6-82 shows four threads—Holdl (Waiting), Admin (Resting), Prune (Resting), and Worker (Resting)—indicating that the Q Capture program is operating properly. Figure 6-82 shows the contents of the Q Capture log indicating a successful start as well.

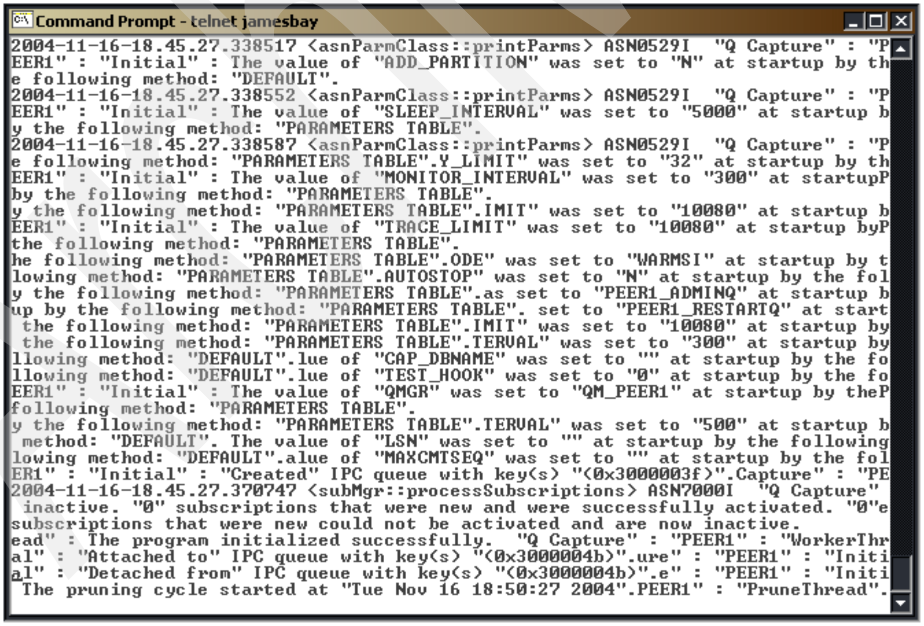


Figure 6-83 Q Capture log contents

Note: The same approach used to view the status of the Q Capture process can be applied to Q Apply as well. However, in the case of Q Apply, there will be three threads—Holdl (Waiting), Admin (Resting), and browser BR00000 (Working)—indicating proper operation of the Q Apply program, as shown in Figure 6-84.

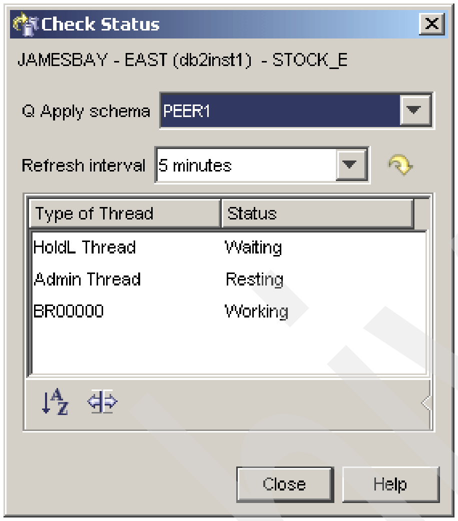


Figure 6-84 Check Status of STOCK_E Q Apply threads

After the Q Capture and Q Apply programs have been started successfully, the Q Capture log will indicate that the subscriptions are ready to be loaded, as shown in Figure 6-85. This indicates that the manual load of the tables involved in the subscriptions can begin.

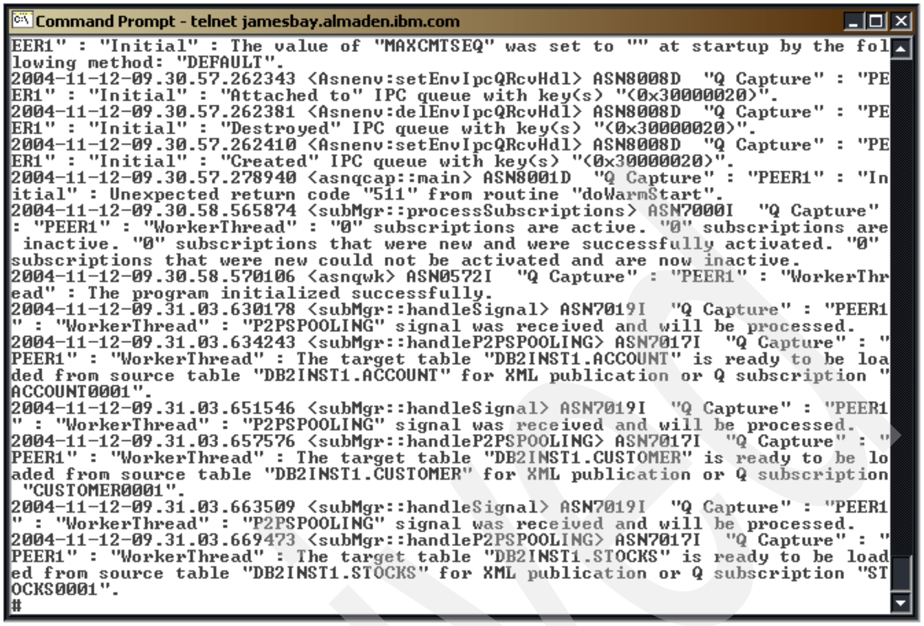


Figure 6-85 Q Capture log contents

Navigate to the Q Subscriptions folder as shown in Figure 6-86. The icons in the right pane show the state of the subscriptions, indicating that the subscriptions are in the load phase.

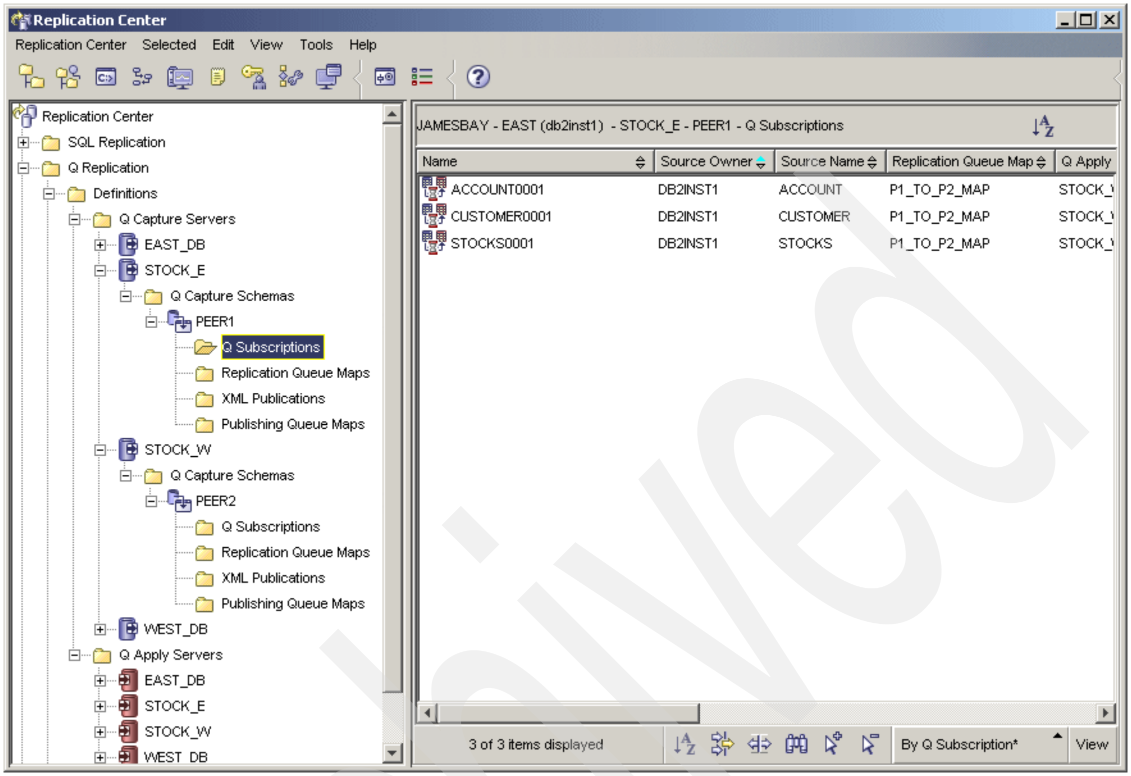


Figure 6-86 Q Subscription state

The status of the subscriptions may also be determined by navigating to the Manage option, as shown in Figure 6-87, which displays Figure 6-88.

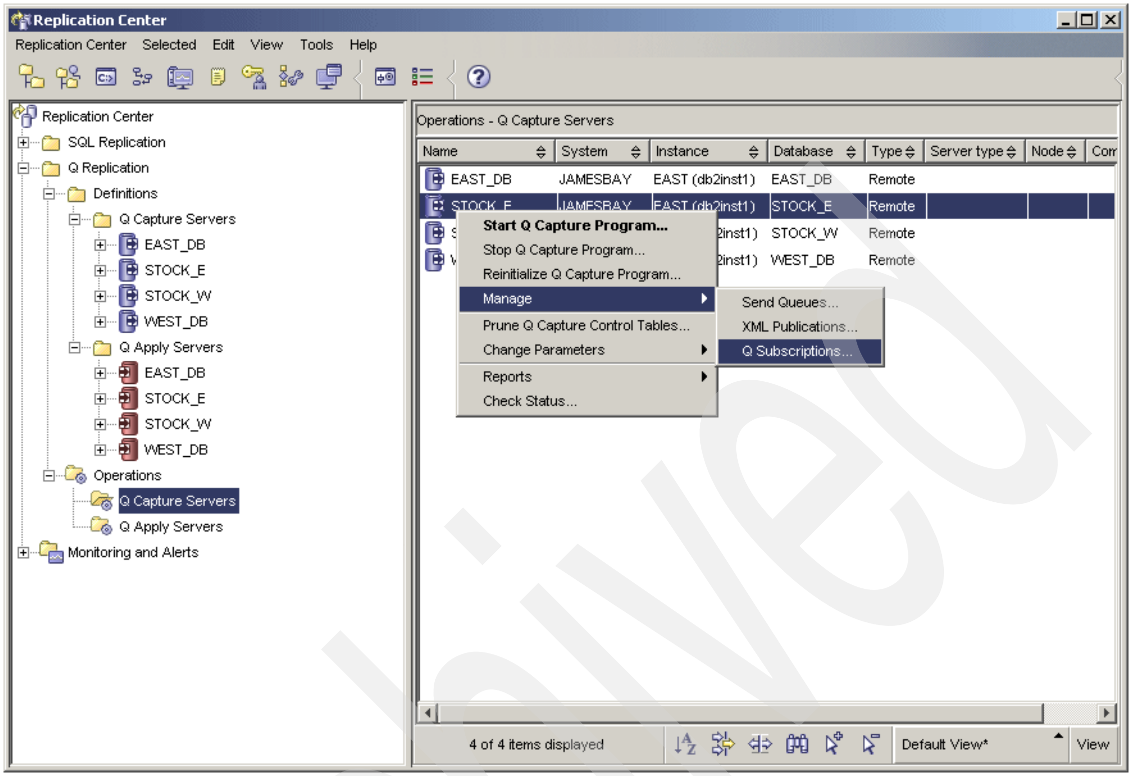


Figure 6-87 Manage Q Subscriptions

Figure 6-88 shows the status of all subscriptions on that server. The icons will again show that the subscriptions are in loading phase. The State column indicates `Requires manual load`. The yellow arrow next to the Refresh interval field can be clicked to refresh the screen to view the current subscription state.

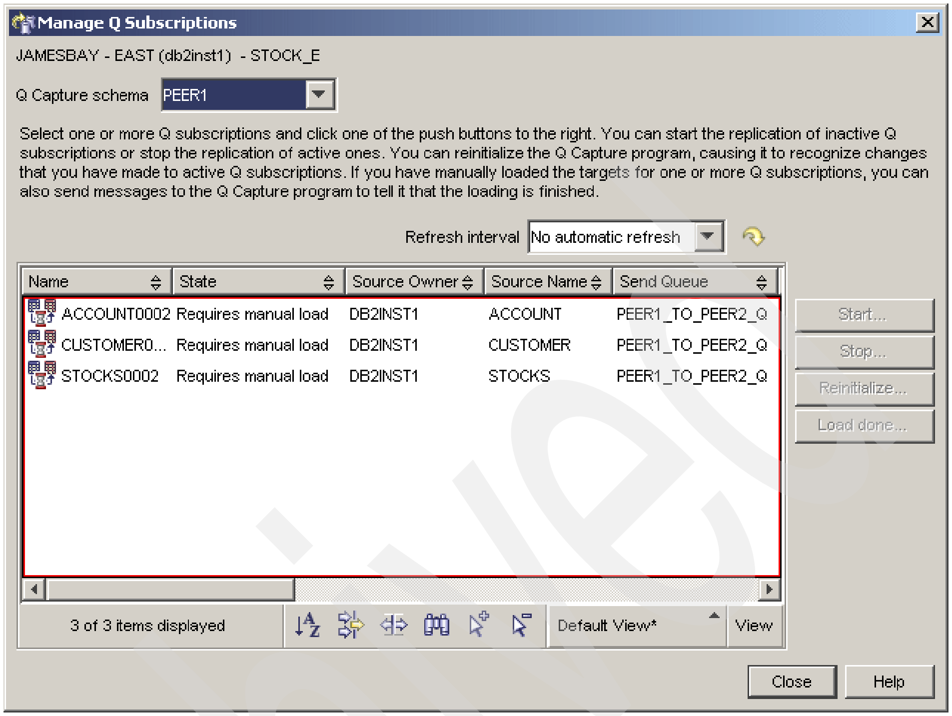


Figure 6-88   Q Subscriptions state

Step 6i: Perform manual load if appropriate

Since we chose manual loading of the tables involved in the subscriptions, this step needs to be performed for our peer-to-peer scenario. The steps involved are:

1. Check the state of the subscriptions and the send and receive queues.

   Prior to beginning the manual load, one should check the state of the subscriptions in both the IBMQREP_SUBS and IBMQREP_TARGETS tables, and the state of the receive and send queues, as shown in Example 6-24 through Example 6-27 on page 651. The states should be as those shown in the examples.

Example 6-24   Check STATE and HAS_LOADPHASE in IBMQREP_SUBS table

```
$ CONNECT TO STOCK_E

$ SELECT SUBNAME, STATE, HAS_LOADPHASE FROM PEER1.IBMQREP_SUBS

   SUBNAME                  STATE HAS_LOADPHASE
   -----------------------------------------
```

```
   ACCOUNT0001            L     E
   CUSTOMER0001           L     E
   STOCKS0001             L     E

     3 record(s) selected.

$ CONNECT RESET
```

In Example 6-24, the L STATE indicates that the Q subscription is loading, and the E HAS_LOADPHASE indicates that it is a manual load.

Example 6-25 Check STATE and HAS_LOADPHASE in IBMQREP_TARGETS table

```
$ CONNECT TO STOCK_W

$ SELECT SUBNAME, SUBGROUP, STATE, HAS_LOADPHASE FROM PEER1.IBMQREP_TARGETS

   SUBNAME         SUBGROUP                               STATE HAS_LOADPHASE
   ----------------------------------------------------------------------------
   ACCOUNT0001     P2P0001                                       A      E
   CUSTOMER0001    P2P0001                                       A      E
   STOCKS0001      P2P0001                                       A      E

     3 record(s) selected.

$ CONNECT RESET
```

In Example 6-25, the A STATE indicates that the Q Apply program is applying changes to the target, and the E HAS_LOADPHASE indicates that it is a manual load.

Example 6-26 Check STATE in IBMQREP_SENDQUEUES table

```
$ CONNECT TO STOCK_E

$ SELECT substr(sendq,1,20) as SENDQ, substr(recvq,1,20) as RECVQ, STATE FROM
PEER1.IBMQREP_SENDQUEUES

   SENDQ                RECVQ                STATE
   -------------------- -------------------- -----
   PEER1_TO_PEER2_Q     PEER1_TO_PEER2_Q     A

     1 record(s) selected.

$ CONNECT RESET
```

In Example 6-26, the A STATE indicates that the named send queue is active.

Example 6-27 Check STATE in IBMQREP_RECVQUEUES table

```
$ CONNECT TO STOCK_E

$ SELECT substr(sendq,1,20) as SENDQ, substr(recvq,1,20) as RECVQ, STATE FROM
PEER1.IBMQREP_RECVQUEUES

   SENDQ                RECVQ                STATE
   -------------------- -------------------- -----
   PEER2_TO_PEER1_Q     PEER2_TO_PEER1_Q     A

   1 record(s) selected.

$ CONNECT RESET
```

In Example 6-27, the A STATE indicates that the named receive queue is active.

2. Insert LOADDONE signal into IBMQREP_SIGNAL table.

   After the manual load of the tables is completed, Q replication needs to be told about it by inserting a LOADDONE signal in the IBMQREP_SIGNAL table so that the subscription can be activated. This can be done via the Replication Center or an SQL INSERT statement as follows:

   a. Replication Center GUI.

      Navigate to Figure 6-89 for generating a LOADDONE signal using the path shown in Figure 6-87 on page 648.

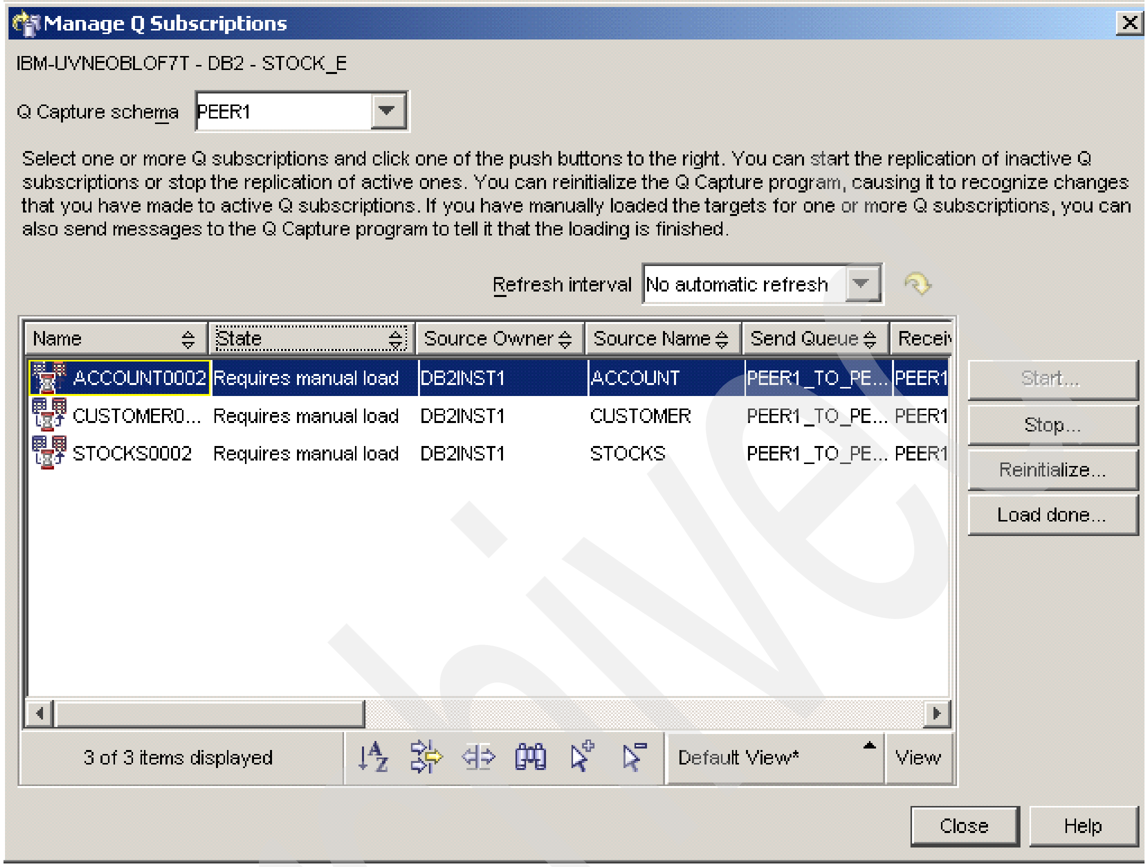


Figure 6-89 LOADDONE signal

Click the **Load done...** button to generate the LOADDONE signal.

b. SQL INSERT statement.

Example 6-28 shows the SQL INSERT statement for submitting a LOADDONE signal.

Example 6-28 SSQL INSERT statement submitting a LOADDONE signal

```
CONNECT TO STOCK_E ;

INSERT INTO PEER1.IBMQREP_SIGNAL
   (SIGNAL_TIME, SIGNAL_TYPE, SIGNAL_SUBTYPE, SIGNAL_INPUT_IN, SIGNAL_STATE)
VALUES
   (CURRENT TIMESTAMP, 'CMD', 'LOADDONE', 'ACCOUNT0002', 'P') ;

INSERT INTO PEER1.IBMQREP_SIGNAL
```

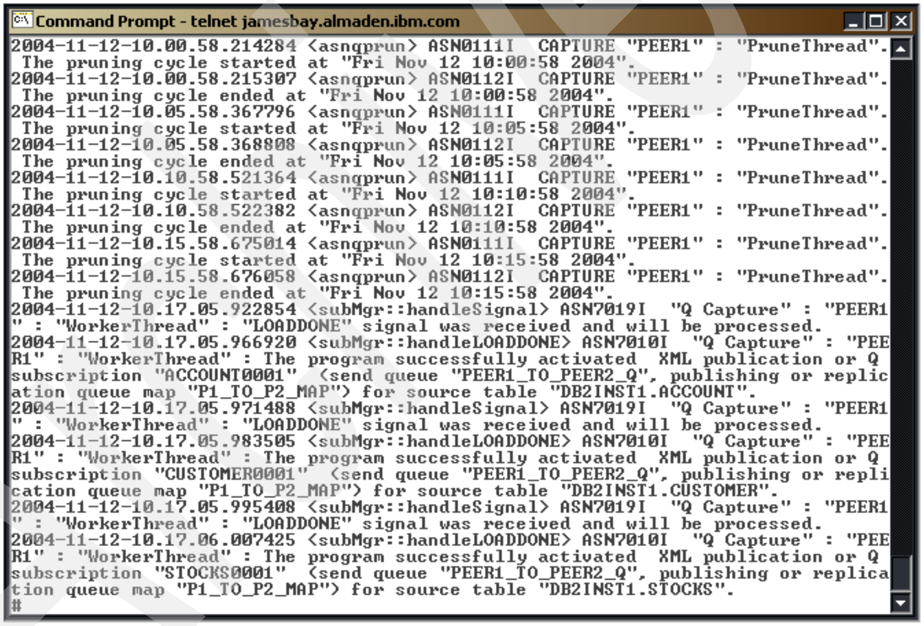
```
   (SIGNAL_TIME, SIGNAL_TYPE, SIGNAL_SUBTYPE, SIGNAL_INPUT_IN, SIGNAL_STATE)
VALUES
   (CURRENT TIMESTAMP, 'CMD', 'LOADDONE', 'CUSTOMER0002', 'P') ;

INSERT INTO PEER1.IBMQREP_SIGNAL
   (SIGNAL_TIME, SIGNAL_TYPE, SIGNAL_SUBTYPE, SIGNAL_INPUT_IN, SIGNAL_STATE)
VALUES
   (CURRENT TIMESTAMP, 'CMD', 'LOADDONE', 'STOCKS0002', 'P') ;

CONNECT RESET ;
```

3. Check for successful activation.

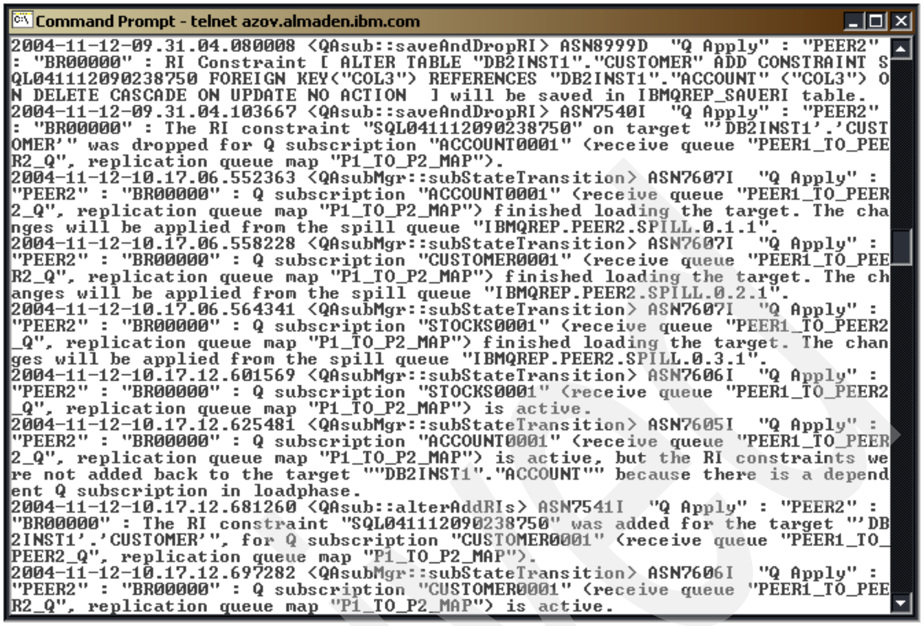
   After the LOADDONE signal is submitted, the Q Capture log should be viewed to determine whether the signal was received for the subscription. Shortly thereafter, the Q Capture log should contain a message indicating that the subscription is active, as shown in Figure 6-90.



```
Command Prompt - telnet jamesbay.almaden.ibm.com
2004-11-12-10.00.58.214284 <asnqprun> ASN0111I  CAPTURE "PEER1" : "PruneThread".
 The pruning cycle started at "Fri Nov 12 10:00:58 2004".
2004-11-12-10.00.58.215307 <asnqprun> ASN0112I  CAPTURE "PEER1" : "PruneThread".
 The pruning cycle ended at "Fri Nov 12 10:00:58 2004".
2004-11-12-10.05.58.367796 <asnqprun> ASN0111I  CAPTURE "PEER1" : "PruneThread".
 The pruning cycle started at "Fri Nov 12 10:05:58 2004".
2004-11-12-10.05.58.368808 <asnqprun> ASN0112I  CAPTURE "PEER1" : "PruneThread".
 The pruning cycle ended at "Fri Nov 12 10:05:58 2004".
2004-11-12-10.10.58.521364 <asnqprun> ASN0111I  CAPTURE "PEER1" : "PruneThread".
 The pruning cycle started at "Fri Nov 12 10:10:58 2004".
2004-11-12-10.10.58.522382 <asnqprun> ASN0112I  CAPTURE "PEER1" : "PruneThread".
 The pruning cycle ended at "Fri Nov 12 10:10:58 2004".
2004-11-12-10.15.58.675014 <asnqprun> ASN0111I  CAPTURE "PEER1" : "PruneThread".
 The pruning cycle started at "Fri Nov 12 10:15:58 2004".
2004-11-12-10.15.58.676058 <asnqprun> ASN0112I  CAPTURE "PEER1" : "PruneThread".
 The pruning cycle ended at "Fri Nov 12 10:15:58 2004".
2004-11-12-10.17.05.922854 <subMgr::handleSignal> ASN7019I  "Q Capture" : "PEER1
" : "WorkerThread" : "LOADDONE" signal was received and will be processed.
2004-11-12-10.17.05.966920 <subMgr::handleLOADDONE> ASN7010I  "Q Capture" : "PEE
R1" : "WorkerThread" : The program successfully activated  XML publication or Q
subscription "ACCOUNT0001"  (send queue "PEER1_TO_PEER2_Q", publishing or replic
ation queue map "P1_TO_P2_MAP") for source table "DB2INST1.ACCOUNT".
2004-11-12-10.17.05.971488 <subMgr::handleSignal> ASN7019I  "Q Capture" : "PEER1
" : "WorkerThread" : "LOADDONE" signal was received and will be processed.
2004-11-12-10.17.05.983505 <subMgr::handleLOADDONE> ASN7010I  "Q Capture" : "PEE
R1" : "WorkerThread" : The program successfully activated  XML publication or Q
subscription "CUSTOMER0001"  (send queue "PEER1_TO_PEER2_Q", publishing or repli
cation queue map "P1_TO_P2_MAP") for source table "DB2INST1.CUSTOMER".
2004-11-12-10.17.05.995408 <subMgr::handleSignal> ASN7019I  "Q Capture" : "PEER1
" : "WorkerThread" : "LOADDONE" signal was received and will be processed.
2004-11-12-10.17.06.007425 <subMgr::handleLOADDONE> ASN7010I  "Q Capture" : "PEE
R1" : "WorkerThread" : The program successfully activated  XML publication or Q
subscription "STOCKS0001"  (send queue "PEER1_TO_PEER2_Q", publishing or replica
tion queue map "P1_TO_P2_MAP") for source table "DB2INST1.STOCKS".
#
```

Figure 6-90   Q Capture log - LOADDONE and Active.message

   After the LOADDONE is received, the Q Apply program applies transactions from the Spill Q and then reinstates any referential integrity constraints for the table(s). The Q Apply log needs to be monitored for any messages. Monitor the Q Apply message log for errors. The Q Apply log in Figure 6-91 contains messages indicating that the Q subscriptions have been activated.




```
2004-11-12-09.31.04.080008 <QAsub::saveAndDropRI> ASN8999D  "Q Apply" : "PEER2"
: "BR00000" : RI Constraint [ ALTER TABLE "DB2INST1"."CUSTOMER" ADD CONSTRAINT S
QL041112090238750 FOREIGN KEY("COL3") REFERENCES "DB2INST1"."ACCOUNT" ("COL3") O
N DELETE CASCADE ON UPDATE NO ACTION  ] will be saved in IBMQREP_SAVERI table.
2004-11-12-09.31.04.103667 <QAsub::saveAndDropRI> ASN7540I  "Q Apply" : "PEER2"
: "BR00000" : The RI constraint "SQL041112090238750" on target "'DB2INST1'.'CUST
OMER'" was dropped for Q subscription "ACCOUNT0001" (receive queue "PEER1_TO_PEE
R2_Q", replication queue map "P1_TO_P2_MAP").
2004-11-12-10.17.06.552363 <QAsubMgr::subStateTransition> ASN7607I  "Q Apply" :
"PEER2" : "BR00000" : Q subscription "ACCOUNT0001" (receive queue "PEER1_TO_PEER
2_Q", replication queue map "P1_TO_P2_MAP") finished loading the target. The cha
nges will be applied from the spill queue "IBMQREP.PEER2.SPILL.0.1.1".
2004-11-12-10.17.06.558228 <QAsubMgr::subStateTransition> ASN7607I  "Q Apply" :
"PEER2" : "BR00000" : Q subscription "CUSTOMER0001" (receive queue "PEER1_TO_PEE
R2_Q", replication queue map "P1_TO_P2_MAP") finished loading the target. The ch
anges will be applied from the spill queue "IBMQREP.PEER2.SPILL.0.2.1".
2004-11-12-10.17.06.564341 <QAsubMgr::subStateTransition> ASN7607I  "Q Apply" :
"PEER2" : "BR00000" : Q subscription "STOCKS0001" (receive queue "PEER1_TO_PEER2
_Q", replication queue map "P1_TO_P2_MAP") finished loading the target. The chan
ges will be applied from the spill queue "IBMQREP.PEER2.SPILL.0.3.1".
2004-11-12-10.17.12.601569 <QAsubMgr::subStateTransition> ASN7606I  "Q Apply" :
"PEER2" : "BR00000" : Q subscription "STOCKS0001" (receive queue "PEER1_TO_PEER2
_Q", replication queue map "P1_TO_P2_MAP") is active.
2004-11-12-10.17.12.625481 <QAsubMgr::subStateTransition> ASN7605I  "Q Apply" :
"PEER2" : "BR00000" : Q subscription "ACCOUNT0001" (receive queue "PEER1_TO_PEER
2_Q", replication queue map "P1_TO_P2_MAP") is active, but the RI constraints we
re not added back to the target ""DB2INST1"."ACCOUNT"" because there is a depend
ent Q subscription in loadphase.
2004-11-12-10.17.12.681260 <QAsub::alterAddRIs> ASN7541I  "Q Apply" : "PEER2" :
"BR00000" : The RI constraint "SQL041112090238750" was added for the target "'DB
2INST1'.'CUSTOMER'", for Q subscription "CUSTOMER0001" (receive queue "PEER1_TO_
PEER2_Q", replication queue map "P1_TO_P2_MAP").
2004-11-12-10.17.12.697282 <QAsubMgr::subStateTransition> ASN7606I  "Q Apply" :
"PEER2" : "BR00000" : Q subscription "CUSTOMER0001" (receive queue "PEER1_TO_PEE
R2_Q", replication queue map "P1_TO_P2_MAP") is active.
```

Figure 6-91 Q Apply log

Figure 6-92 shows the navigation to the Q Subscriptions folder for STOCK_E displaying a green ball icon for the subscriptions in the right-hand pane—indicating that the subscriptions are active.

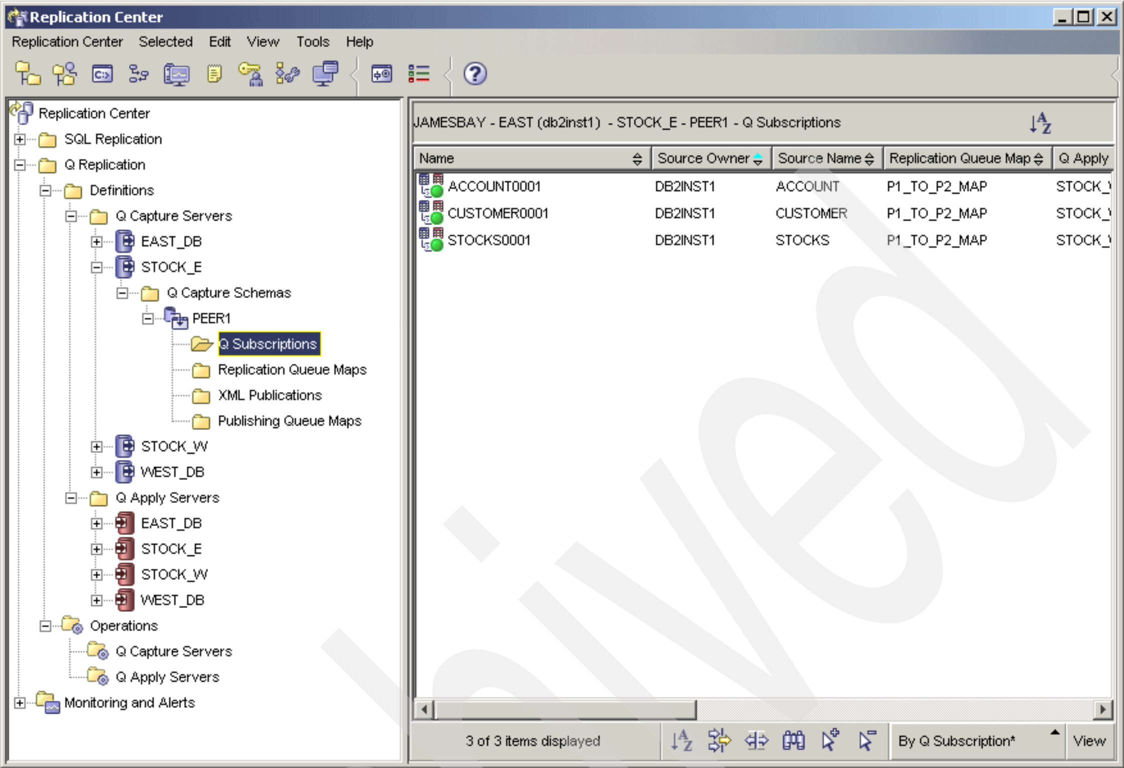


Figure 6-92 Q subscriptions status

The subscriptions state may also be reviewed in the Operations folder of the navigation pane, as shown in Figure 6-93 and Figure 6-94.

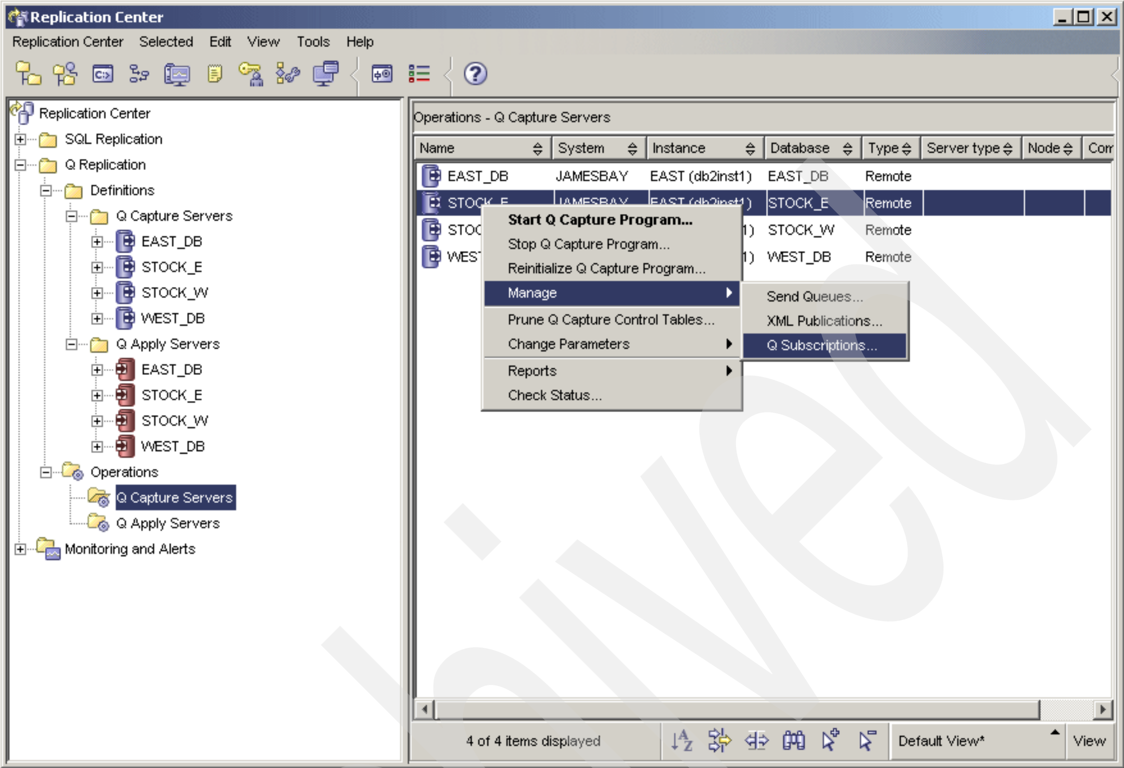


Figure 6-93 Operations folder - Manage Q Subscriptions

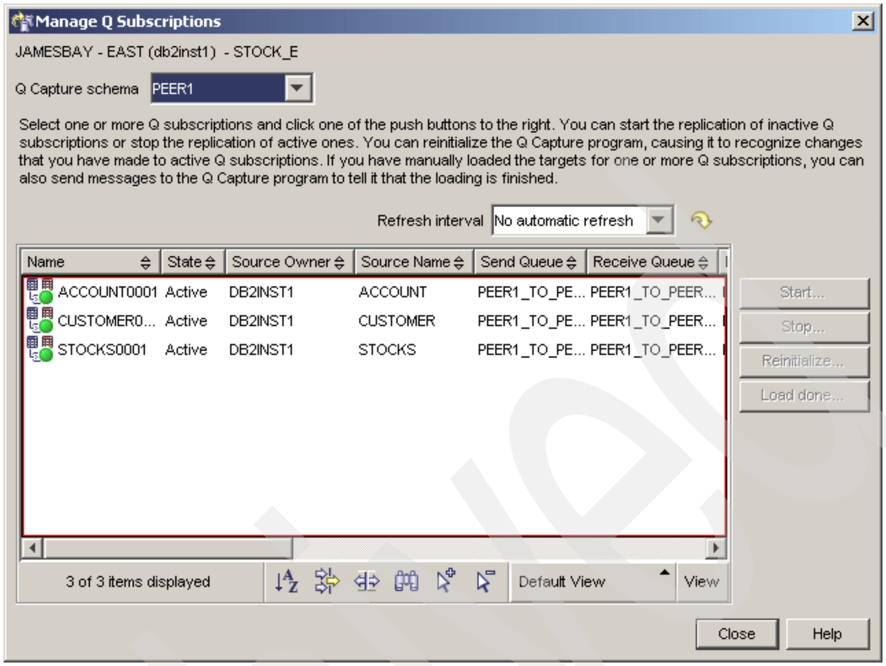


Figure 6-94 Q Subscriptions state

6.5.7 Step 6: Configure and activate Q replication using commands

WebSphere II Q replication configuration may be performed using GUI, as described in 6.5.6, “Step 6: Configure and activate Q replication using GUI” on page 572; or by using ASNCLP commands from the Unix command line. This section describes the configuration of the same peer-to-peer Q replication topology solution 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 6-29.

Example 6-29 Set up Java environment for user ID qrepladm

```
PATH=$PATH:/usr/java14/bin
```

```
CP=/db2_data/db2inst1/sqllib/java/Common.jar
CP=$CP:/db2_data/db2inst1/sqllib/tools/db2cmn.jar
CP=$CP:/db2_data/db2inst1/sqllib/tools/db2replapis.jar
CP=$CP:/db2_data/db2inst1/sqllib/tools/db2qreplapis.jar
CP=$CP:/db2_data/db2inst1/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 setup was successful. Enter `quit` to exit the Repl> command line, as shown in Example 6-30.

Example 6-30 asnclp setup 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 6-31.

Example 6-31 Execute commands from a file

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

The ASNCLP commands in the following sections are categorized according to the following tasks:

1. Create the Q Capture and Q Apply control tables.
2. Create the replication queue maps.
3. Create the subscriptions.
4. Create a password file.
5. Update the CAPTURE_PATH and APPLY_PATH.
6. Start the Q Capture and Q Apply programs.
7. Verify successful implementation.

Create the Q Capture and Q Apply control tables

Example 6-32 lists the ASNCLP commands for creating the Q Capture and Q Apply control tables on node PEER1 (Jamesbay), while Example 6-33 does the same on node PEER2 (Azov).

Example 6-32 Create the Q Capture and Q Apply control tables on PEER1

```
#################################################
```

```
# The following commands create the control
# tables in an existing tablespace USERSPACE1.  If
# a tablespaces needs to be created automatically
# the user ID (qrepladm) must have SYSCTRL
# authority.
####################################################
ASNCLP SESSION SET TO Q REPLICATION;
SET OUTPUT MULTIDIR;
#############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 6-1 on page 549, Table 6-2 on page 550 and Table 6-3 on page 552
############################################################
##STOCK_E from reference A.7a in Table 6-1 on page 549
##QREPLADM from reference A.7b in Table 6-1 on page 549
##QM_PEER1 from reference A.11 in Table 6-2 on page 550
##Capture schema PEER1 from reference A.22 in Table 6-3 on page 552
##PEER1_RESTARTQ from reference A.16 in Table 6-2 on page 550
##PEER1_ADMINQ from reference A.17 in Table 6-2 on page 550
#############################################################
SET SERVER CAPTURE TO DB STOCK_E ID QREPLADM PASSWORD "xxxxxx";
SET QMANAGER "QM_PEER1" FOR CAPTURE SCHEMA;
SET CAPTURE SCHEMA SOURCE PEER1;
SET RUN SCRIPT NOW STOP ON SQL ERROR ON;
CREATE CONTROL TABLES FOR CAPTURE SERVER USING RESTARTQ "PEER1_RESTARTQ" ADMINQ
"PEER1_ADMINQ" STARTMODE WARMSI IN UW TBSPACE USERSPACE1;
############################################################
##STOCK_E from reference A.7a in Table 6-1 on page 549
##QREPLADM from reference A.7c in Table 6-1 on page 549
##QM_PEER1 from reference A.11 in Table 6-2 on page 550
##Apply schema PEER1 from reference A.23 in Table 6-3 on page 552
#############################################################
SET SERVER TARGET TO DB STOCK_E ID QREPLADM PASSWORD "xxxxxx";
SET QMANAGER "QM_PEER1" FOR APPLY SCHEMA;
SET APPLY SCHEMA PEER1;
SET RUN SCRIPT NOW STOP ON SQL ERROR ON;
CREATE CONTROL TABLES FOR APPLY SERVER IN UW TBSPACE USERSPACE1;
```

Example 6-33 Create the Q Capture and Q Apply control tables on PEER2

```
######################################################
# The following commands create the control
# tables in an existing tablespace USERSPACE1.  If
# a tablespaces needs to be created automatically
# the user ID (qrepladm) must have SYSCTRL
# authority.
#####################################################
ASNCLP SESSION SET TO Q REPLICATION;
SET OUTPUT MULTIDIR;
```

```
###########################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 6-1 on page 549, Table 6-2 on page 550 and Table 6-3 on page 552
##########################################################
##STOCK_W from reference B.7a in Table 6-1 on page 549
##QREPLADM from reference B.7b in Table 6-1 on page 549
##QM_PEER2 from reference B.11 in Table 6-2 on page 550
##Capture schema PEER2 from reference B.22 in Table 6-3 on page 552
##PEER2_RESTARTQ from reference B.16 in Table 6-2 on page 550
##PEER2_ADMINQ from reference B.17 in Table 6-2 on page 550
###########################################################
SET SERVER CAPTURE TO DB STOCK_W ID QREPLADM PASSWORD "xxxxxx";
SET QMANAGER "QM_PEER2" FOR CAPTURE SCHEMA;
SET CAPTURE SCHEMA SOURCE PEER2;
SET RUN SCRIPT NOW STOP ON SQL ERROR ON;
CREATE CONTROL TABLES FOR CAPTURE SERVER USING RESTARTQ "PEER2_RESTARTQ" ADMINQ
"PEER2_ADMINQ" STARTMODE WARMSI IN UW TBSPACE USERSPACE1;
##########################################################
##STOCK_W from reference B.7a in Table 6-1 on page 549
##QREPLADM from reference B.7c in Table 6-1 on page 549
##QM_PEER2 from reference B.11 in Table 6-2 on page 550
##Apply schema PEER2 from reference B.23 in Table 6-3 on page 552
###########################################################
SET SERVER TARGET TO DB STOCK_W ID QREPLADM PASSWORD "xxxxxx";
SET QMANAGER "QM_PEER2" FOR APPLY SCHEMA;
SET APPLY SCHEMA PEER2;
SET RUN SCRIPT NOW STOP ON SQL ERROR ON;
CREATE CONTROL TABLES FOR APPLY SERVER IN UW TBSPACE USERSPACE1;
```

Create the replication queue maps

Example 6-34 lists the ASNCLP commands for creating the replication queue maps on node EAST (Jamesbay), while Example 6-35 does the same on node WEST (Azov).

Example 6-34 Create the replication queue maps on PEER1

```
###########################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 6-1 on page 549, Table 6-2 on page 550 and Table 6-3 on page 552
##########################################################
##STOCK_E from reference A.7a in Table 6-1 on page 549
##QREPLADM from reference A.7b in Table 6-1 on page 549
##STOCK_W from reference B.7a in Table 6-1 on page 549
##QREPLADM from reference B.7c in Table 6-1 on page 549
##Capture schema PEER1 from reference A.22 in Table 6-3 on page 552
##Apply schema PEER2 from reference B.23 in Table 6-3 on page 552
```

```
##P1_TO_P2_MAP from reference A.27 in Table 6-3 on page 552
##PEER1_ADMINQ from reference A.17 in Table 6-2 on page 550
##RECVQ PEER1_TO_PEER2_Q from reference B.19 in Table 6-2 on page 550
##SENDQ PEER1_TO_PEER2_Q from reference A.18 in Table 6-2 on page 550
#############################################################

ASNCLP SESSION SET TO Q REPLICATION;
SET OUTPUT MULTIDIR;

SET SERVER CAPTURE TO DB STOCK_E ID "QREPLADM" PASSWORD "xxxxxx";
SET SERVER TARGET  TO DB STOCK_W ID "QREPLADM" PASSWORD "xxxxxx";

SET CAPTURE SCHEMA SOURCE PEER1;
SET APPLY   SCHEMA PEER2;
SET RUN SCRIPT NOW STOP ON SQL ERROR ON;
CREATE REPLQMAP P1_TO_P2_MAP USING ADMINQ "PEER1_ADMINQ" RECVQ
"PEER1_TO_PEER2_Q" SENDQ "PEER1_TO_PEER2_Q" NUM APPLY AGENTS 4 MEMORY LIMIT 30
ERROR ACTION I MAX MESSAGE SIZE 2048;
```

Example 6-35 Create the replication queue maps on PEER2

```
############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 6-1 on page 549, Table 6-2 on page 550 and Table 6-3 on page 552
############################################################
##STOCK_E from reference A.7a in Table 6-1 on page 549
##QREPLADM from reference A.7b in Table 6-1 on page 549
##STOCK_W from reference B.7a in Table 6-1 on page 549
##QREPLADM from reference B.7c in Table 6-1 on page 549
##Capture schema PEER2 from reference B.22 in Table 6-3 on page 552
##Apply schema PEER1 from reference A.23 in Table 6-3 on page 552
##P2_TO_P1_MAP from reference B.27 in Table 6-3 on page 552
##PEER2_ADMINQ from reference B.17 in Table 6-2 on page 550
##RECVQ PEER2_TO_PEER1_Q from reference A.19 in Table 6-2 on page 550
##SENDQ PEER2_TO_PEER1_Q from reference B.18 in Table 6-2 on page 550
#############################################################
ASNCLP SESSION SET TO Q REPLICATION;
SET OUTPUT MULTIDIR;

SET SERVER CAPTURE TO DB STOCK_W ID "QREPLADM" PASSWORD "xxxxxx";
SET SERVER TARGET  TO DB STOCK_E ID "QREPLADM" PASSWORD "xxxxxx";

SET CAPTURE SCHEMA SOURCE PEER2;
SET APPLY   SCHEMA PEER1;
SET RUN SCRIPT NOW STOP ON SQL ERROR ON;
```

```
CREATE REPLQMAP W_TO_E_MAP USING ADMINQ "PEER2_ADMINQ" RECVQ "PEER2_TO_PEER1_Q"
SENDQ "PEER2_TO_PEER1_Q" NUM APPLY AGENTS 4 MEMORY LIMIT 30 ERROR ACTION I MAX
MESSAGE SIZE 2048;
```

Create the subscriptions

Example 6-36 and Example 6-37 list the ASNCLP commands for creating the subscriptions on node PEER1 (Jamesbay) and node PEER2 (Azov).

Example 6-36 Create the subscriptions

```
#####################################################
# Create subscriptions on PEER1
#####################################################
ASNCLP SESSION SET TO Q REPLICATION;
SET OUTPUT MULTIDIR;
SET RUN SCRIPT NOW STOP ON SQL ERROR ON;
LOAD MULTIDIR REPL SCRIPT "subs";
```

Example 6-37 Content of the subs file

```
#############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 6-1 on page 549, Table 6-2 on page 550 and Table 6-3 on page 552
############################################################
##STOCK_E from reference A.7a in Table 6-1 on page 549
##DB2INST1 from reference A.3 in Table 6-1 on page 549
##STOCK_W from reference B.7a in Table 6-1 on page 549
##DB2INST1 from reference B.3 in Table 6-1 on page 549
##Capture schema PEER1 from reference A.22 in Table 6-3 on page 552
##Apply schema PEER2 from reference B.23 in Table 6-3 on page 552
##P1_TO_P2_MAP from reference A.27 in Table 6-3 on page 552
##P2_TO_P1_MAP from reference B.27 in Table 6-3 on page 552
##STOCK_E.PEER1.DB2INST1.ACCOUNT & STOCK_W.PEER2.DB2INST1.ACCOUNT table pair
##STOCK_E.PEER1.DB2INST1.CUSTOMER & STOCK_W.PEER2.DB2INST1.CUSTOMER table pair
##STOCK_E.PEER1.DB2INST1.STOCKS & STOCK_W.PEER2.DB2INST1.STOCKS table pair
#############################################################

SET SUBGROUP "P2P0001";

SET SERVER MULTIDIR TO DB STOCK_E ID "QREPLADM" PASSWORD "xxxxxx";
SET SERVER MULTIDIR TO DB stock_w ID "QREPLADM" PASSWORD "xxxxxx";

SET MULTIDIR SCHEMA STOCK_E.PEER1;
SET MULTIDIR SCHEMA STOCK_W.PEER2;

SET CONNECTION SOURCE STOCK_E.PEER1 TARGET STOCK_W.PEER2 REPLQMAP P1_TO_P2_MAP;
SET CONNECTION SOURCE STOCK_W.PEER2 TARGET STOCK_E.PEER1 REPLQMAP P2_TO_P1_MAP;
```

```
set tables (STOCK_E.PEER1.DB2INST1.ACCOUNT, STOCK_W.PEER2.DB2INST1.ACCOUNT);
create qsub subtype P
source has load phase E
target error action Q;

set tables (STOCK_E.PEER1.DB2INST1.CUSTOMER, STOCK_W.PEER2.DB2INST1.CUSTOMER);
create qsub subtype P
source has load phase E
target error action Q;

set tables (STOCK_E.PEER1.DB2INST1.STOCKS, STOCK_W.PEER2.DB2INST1.STOCKS);
create qsub subtype P
source has load phase E
target error action Q;
```

Create a password file

If automatic loading of the tables was chosen, 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 table.

Example 6-38 creates the password file for user ID qrepladm on node PEER1 to access the source tables on STOCK_W, while Example 6-39 does the same on node PEER2.

Example 6-38 Create the password file on PEER1

```
cd ~qrepladm/apply
asnpwd init
asnpwd add alias STOCK_W id QREPLADM password "xxxxxx"
```

Example 6-39 Create the password file on PEER2

```
cd ~qrepladm/apply
asnpwd init
asnpwd add alias STOCK_E id QREPLADM password "xxxxxx"
```

Note: Automatic loading occurs every time a subscription changes from a deactivated state to an activated state.

Update the CAPTURE_PATH and APPLY_PATH

The Replication control tables IBMQREP_CAPPARMS and IBMQREP_APPLYPARMS contain a column for CAPTURE_PATH and APPLY_PATH, respectively. Prior to starting the Q Capture and Q Apply programs, these columns need to be set to the values /db2_data/capture and

/db2_data/apply on PEER1 (Jamesbay) and PEER2 (Azov), as shown in Example 6-40 and Example 6-41, respectively.

Example 6-40 Update the Capture and Apply paths on PEER1

```
#############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 6-1 on page 549, Table 6-2 on page 550 and Table 6-3 on page 552
############################################################
##STOCK_E from reference A.7a in Table 6-1 on page 549
##/db2_data/capture from reference A.24 in Table 6-3 on page 552
##/db2_data/apply from reference A.25 in Table 6-3 on page 552
#############################################################

CONNECT TO STOCK_E ;

UPDATE EAST.IBMQREP_APPLYPARMS SET APPLY_PATH='/db2_data/apply',
PWDFILE='asnpwd.aut' ;

UPDATE EAST.IBMQREP_CAPPARMS SET CAPTURE_PATH='/db2_data/capture' ;

CONNECT RESET ;
```

Example 6-41 Update the Capture and Apply paths on PEER2

```
#############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 6-1 on page 549, Table 6-2 on page 550 and Table 6-3 on page 552
############################################################
##STOCK_W from reference B.7a in Table 6-1 on page 549
##/db2_data/capture from reference B.24 in Table 6-3 on page 552
##/db2_data/apply from reference B.25 in Table 6-3 on page 552
#############################################################
CONNECT TO STOCK_W;

UPDATE WEST.IBMQREP_APPLYPARMS SET APPLY_PATH='/db2_data/apply',
PWDFILE='asnpwd.aut' ;

UPDATE WEST.IBMQREP_CAPPARMS SET CAPTURE_PATH='/db2_data/capture' ;

CONNECT RESET ;
```

Start the Q Capture and Q Apply programs

The only way to verify that the Q replication objects have been created properly is to start replication. The Q Capture and Q Apply programs can be started through the Replication Center or via the Unix command line, as shown in Example 6-42 through Example 6-45.

Example 6-42 Start Q Capture on PEER1

```
############################################################
##STOCK_E from reference A.7a in Table 6-1 on page 549
##Capture schema PEER1 from reference A.22 in Table 6-3 on page 552
#############################################################
asnqcap capture_server=STOCK_E capture_schema=PEER1
```

Example 6-43 Start Q Capture on PEER2

```
############################################################
##STOCK_W from reference B.7a in Table 6-1 on page 549
##Capture schema PEER2 from reference B.22 in Table 6-3 on page 552
#############################################################
asnqcap capture_server=STOCK_W capture_schema=PEER2
```

Example 6-44 Start Q Apply on PEER1

```
############################################################
##STOCK_E from reference A.7a in Table 6-1 on page 549
##Apply schema PEER1 from reference A.23 in Table 6-3 on page 552
#############################################################
asnqapp apply_server=STOCK_E apply_schema=PEER1
```

Example 6-45 Start Q Apply on PEER2

```
############################################################
##STOCK_W from reference B.7a in Table 6-1 on page 549
##Apply schema PEER2 from reference B.23 in Table 6-3 on page 552
#############################################################
asnqapp apply_server=STOCK_W apply_schema=PEER2
```

Verify successful implementation

Since we chose manual load for all Q Subscriptions, we first need to load the tables so that all tables are synchronized, and verify successful implementation. This is discussed in detail in "Step 6h: Verify status of Q Capture and Q Apply processes" on page 642, and "Step 6i: Perform manual load if appropriate" on page 649.

6.5.8 Step 7: Set up monitoring environment using the GUI

To ensure a smooth operational Q replication environment, it is necessary to set up the Replication Alert Monitor to monitor the Q Capture and Q Apply processes, as well as the efficacy of the replication. An overview of the Replication Alert Monitor is described in 2.1.4, “Replication Alert Monitor” on page 21.

The Replication Alert Monitor provides basic monitoring of the following:

- Q Capture program
  - Status
  - Errors
  - Warnings
  - Latency
  - Memory
  - Transaction size
  - Subscriptions inactivated
- Q Apply program
  - Status
  - Errors
  - Warnings
  - Latency
  - End-to-end Latency
  - Memory
  - Exceptions
  - Spill queue depth
  - Receive queue depth

In this section we document the step-by-step setup of the Replication Alert Monitor using the GUI interface to manage the Q Capture and Q Apply processes in the STOCK_E and STOCK_W database servers. This approach can then be applied to manage the same processes on the EAST_DB and WEST_DB servers as well.

Figure 6-95 lists the main steps involved in setting up this environment. Each of these steps is described in detail in the following sections.

Important: It is assumed that the user ID qrepladm (reference M.5c in Table 6-4 on page 553) has been created with the appropriate database privileges on the server (reference M.1 in Table 6-4 on page 553) to be configured. If not, then this task must first be performed.

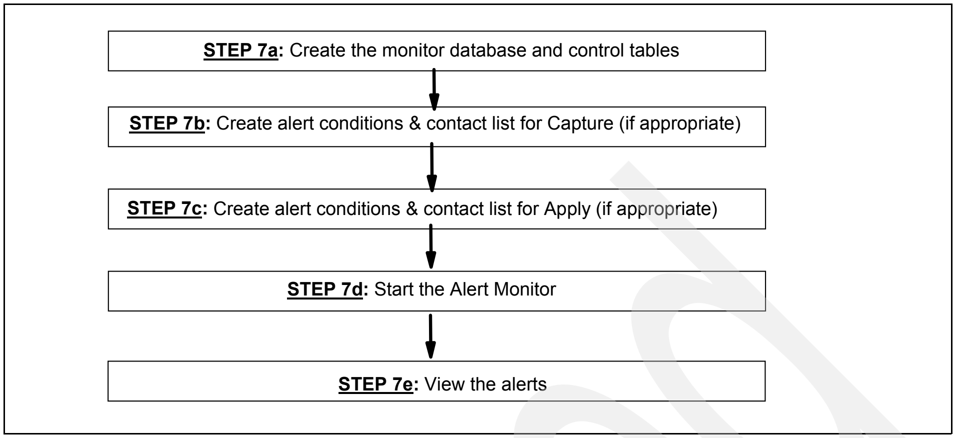


Figure 6-95 Overview of Replication Alert Monitor setup steps

Step 7a: Create the monitor database and control tables

Create a monitor database MONDB (reference M.5a in Table 6-4 on page 553) in the DB2 instance DB2 (reference M.5b in Table 6-4 on page 553) on node SOUTH (Kanaga.almaden.ibm.com reference M.1 in Table 6-4 on page 553) using commands, as shown in Example 6-46. The database may also be created via the DB2 Control Center (not shown here).

Example 6-46 Create the monitor database MONDB on Kanaga.almaden.ibm.com

```
create database MONDB;
```

> **Note:** The general recommendation is to isolate the Replication Alert Monitor database on a server and instance separate from the managed database servers.

The MONDB database then needs to be cataloged on Jamesbay, as shown in Example 6-47, since we intend to use the Replication Center on Jamesbay for our monitoring.

Example 6-47 Catalog MONDB on Jamesbay

```
db2 catalog tcpip node SOUTH remote 9.1.39.90 server 50000
db2 catalog database MONDB at node SOUTH
db2 terminate;
```

Invoke the Replication Center on Jamesbay, and right-click the **Monitor Control Servers** folder in the navigation pane of the Replication Center. Then select **Create Monitor Control Tables...** (as shown in Figure 6-96) to bring up a list of potential servers wherein the Monitor[3] Control Tables may be created, as shown in Figure 6-97.

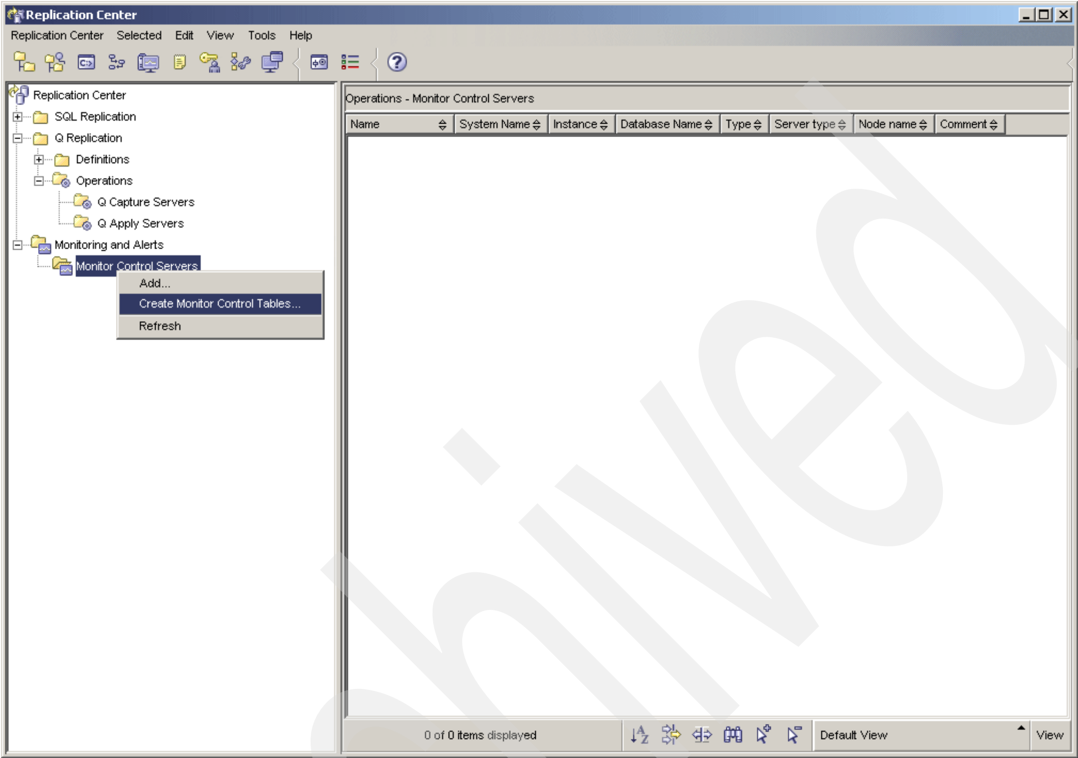


Figure 6-96 Create the Monitor Control Tables

[3] Otherwise, ensure that MONDB is cataloged on the database server where the Replication Center is invoked.

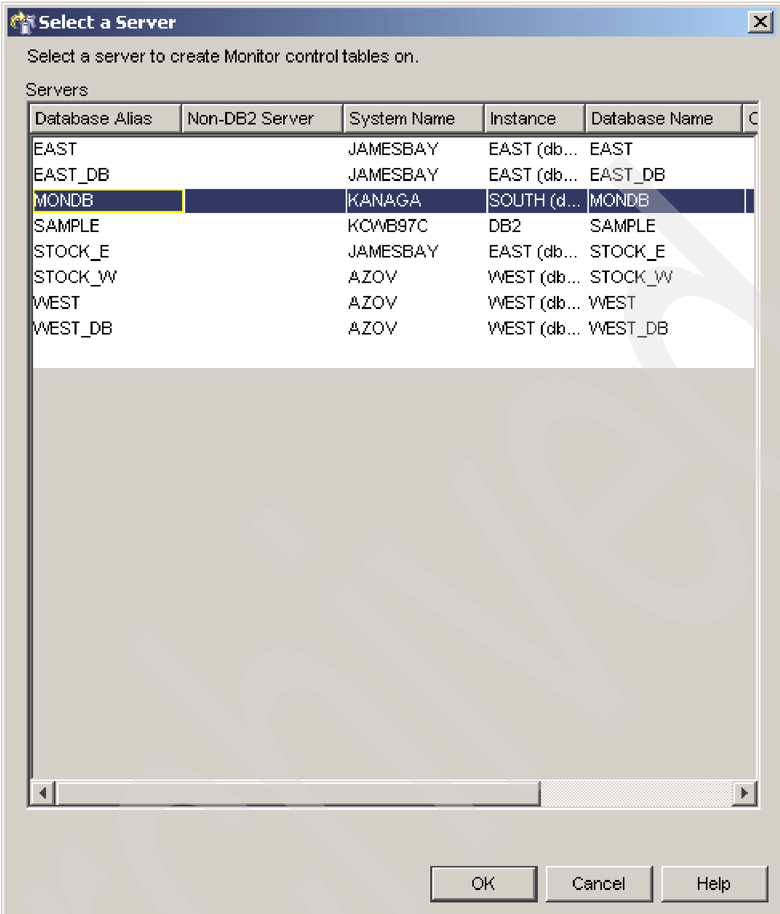


Figure 6-97 Select a Server to create the Monitor Control Tables

Select **MONDB** on Kanaga as the target for the Monitor Control Tables, and click **OK** to proceed to Figure 6-98 in order to provide appropriate information for these tables.

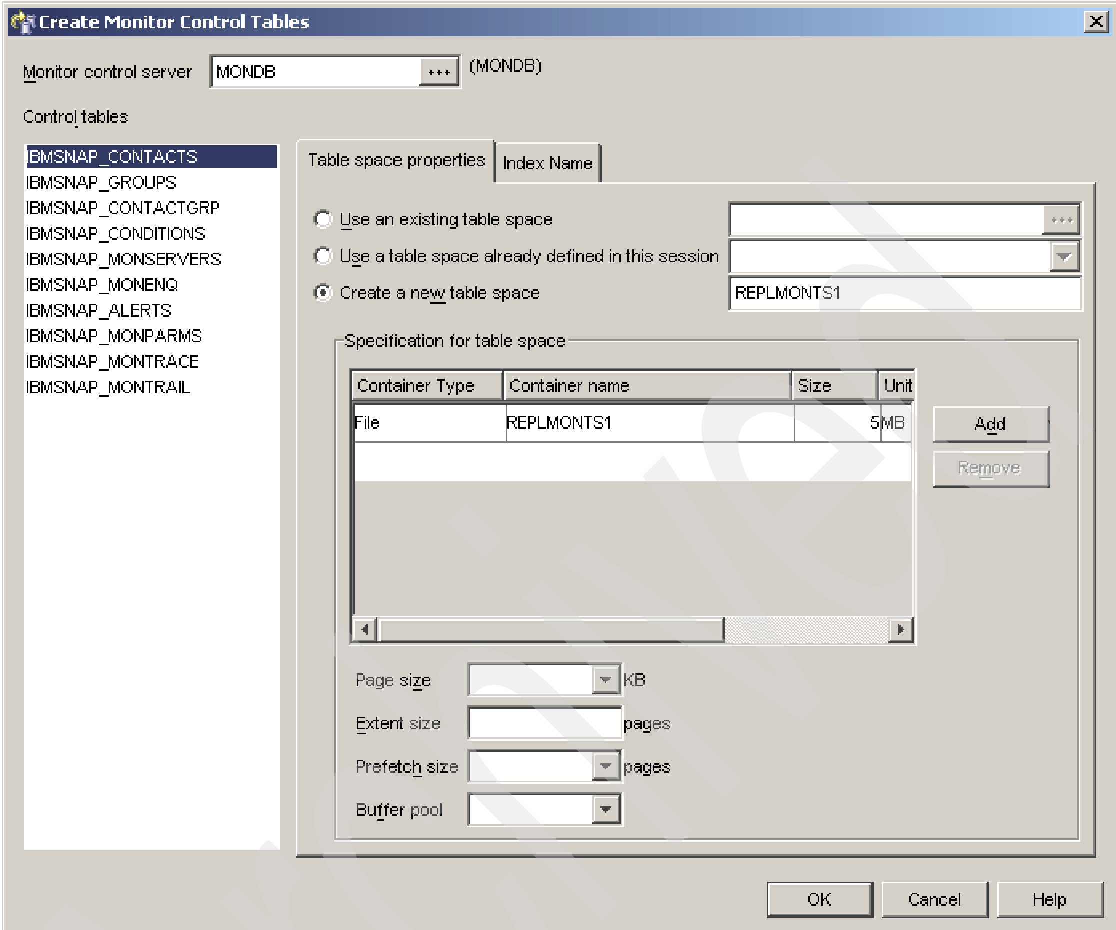


Figure 6-98 Create Monitor Control Tables in MONDB database

Select the **IBMSNAP.CONTACTS** and supply the tablespace properties such as "Create a new table space" named REPLMONTS1, as shown in Figure 6-98. If information is not provided for attributes such as Page size, Extent size, etc., it will default.

Click the **Index Name** tab to proceed to Figure 6-99 to provide index details for this table.

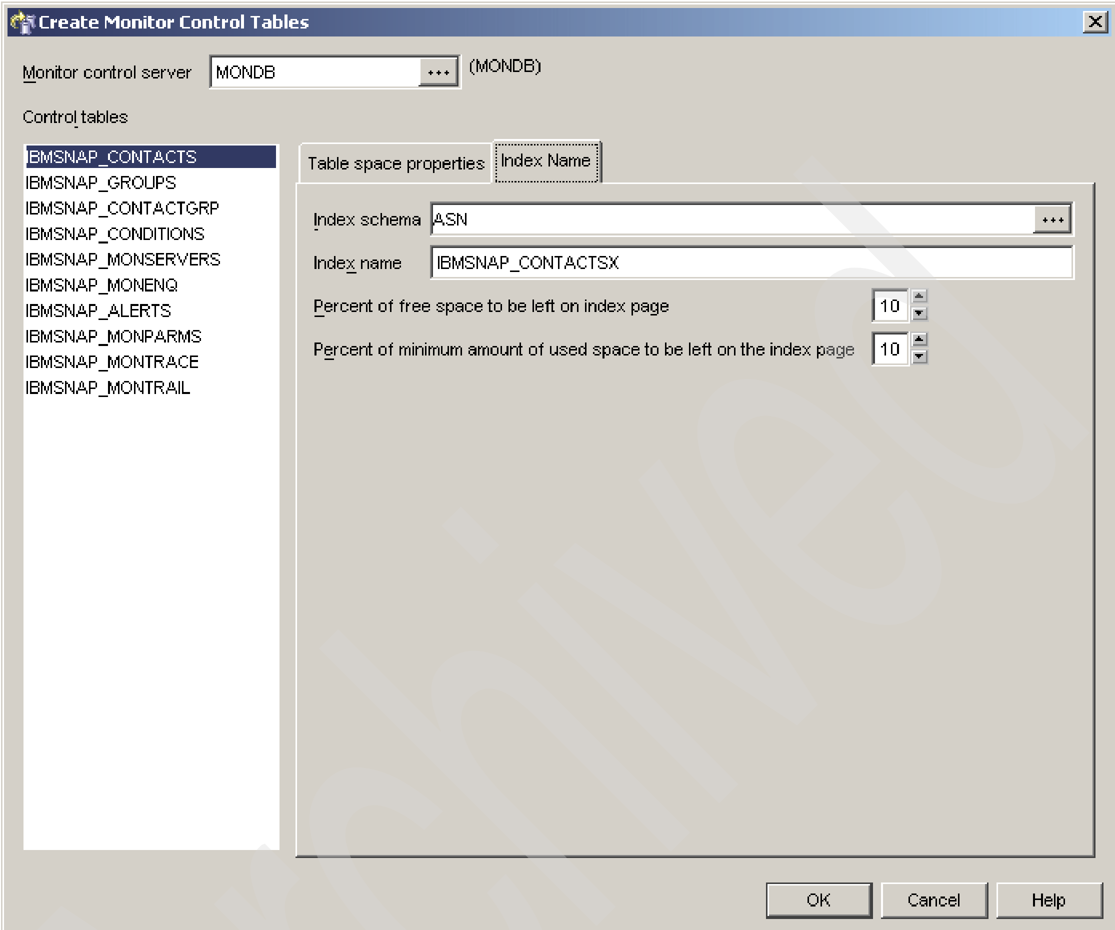


Figure 6-99 Create Monitor Control Tables - Index details

Supply the index details as shown in Figure 6-99, and repeat the steps for all the other monitor control tables. Click **OK** when complete to generate the scripts to create the control tables, as shown in Figure 6-100.

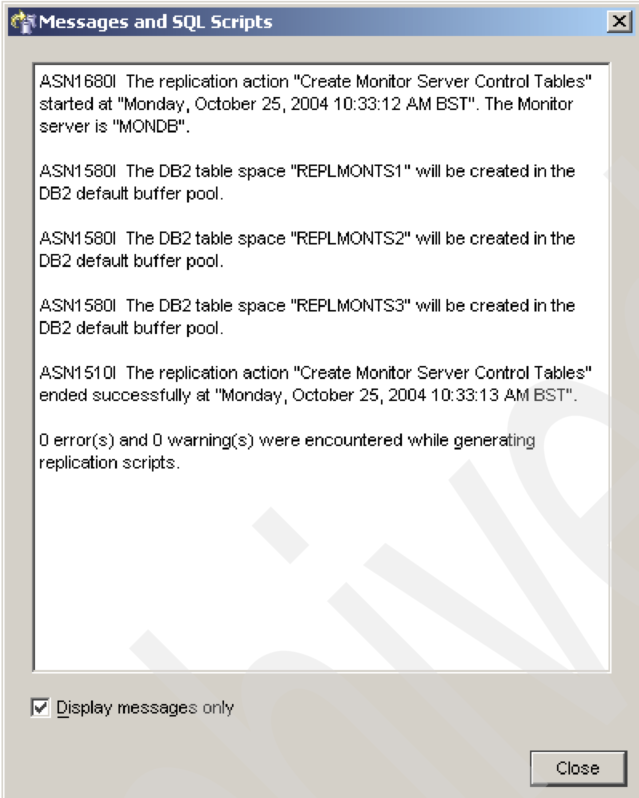


Figure 6-100 Messages and SQL Scripts

Figure 6-100 shows the successful generation of the script required to create the monitor control tables. Click **Close** to be provided with the choice of executing the script right away or saving it for later execution, as shown in Figure 6-101.

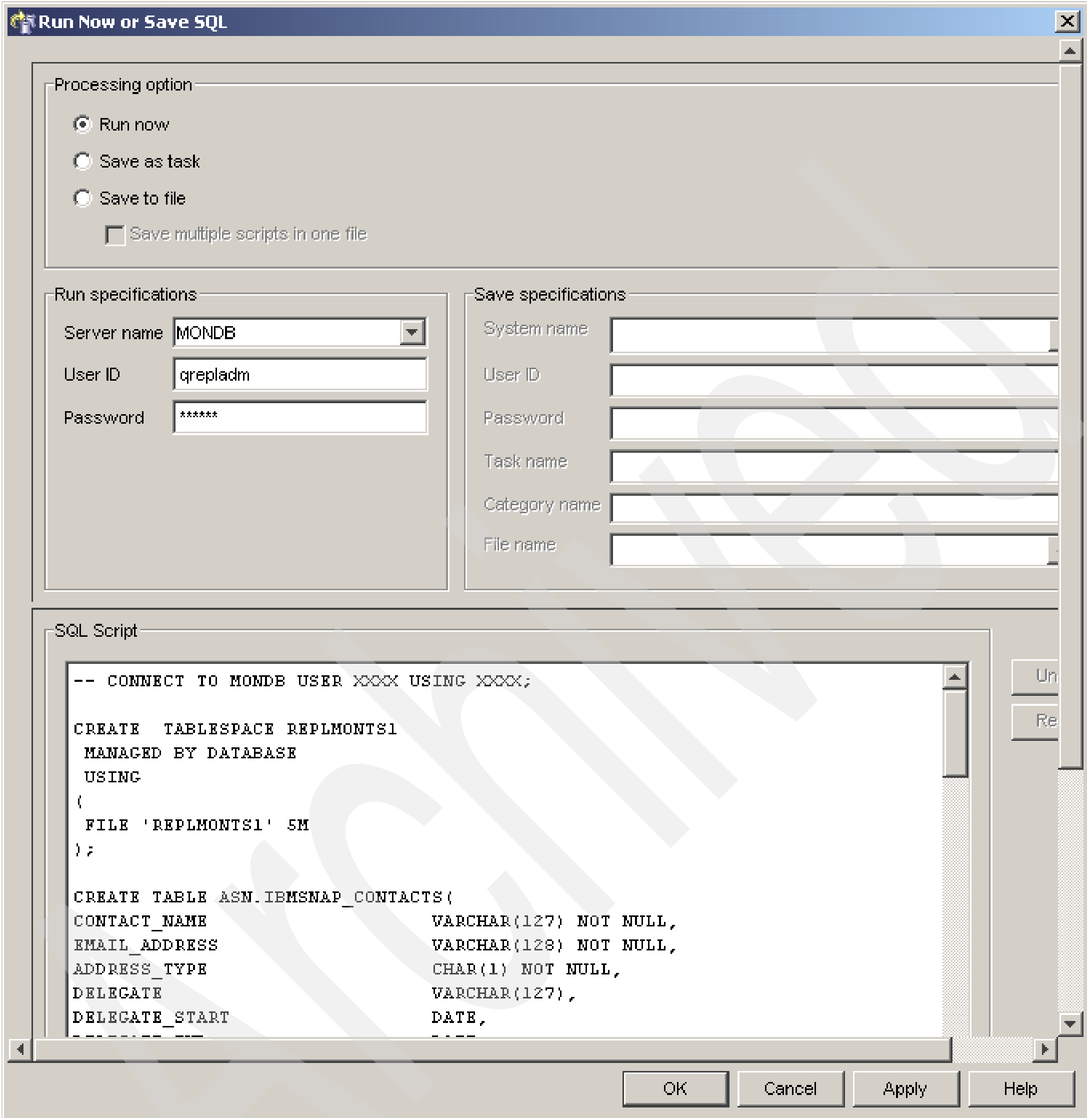


Figure 6-101 Run Now or Save SQL

Select **Run Now** and click **OK** to execute the generated script immediately. Figure 6-102 displays the successful execution of the generated script on the MONDB database server.

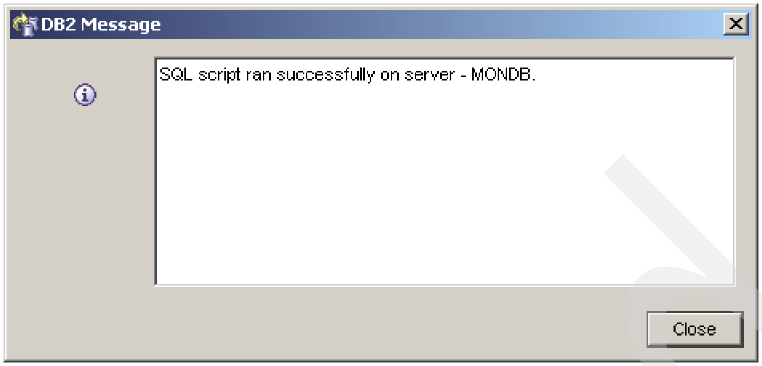


Figure 6-102 SQL script ran successfully message

Step 7b: Create alert conditions and contact list for Capture

The alert conditions to be monitored for and the contact list to be used in the event of alerts related to Q Capture are defined in this step. Figure 6-103 shows the Replication Center with the **MONDB** folder selected. The display pane shows folders Monitors and Contacts.

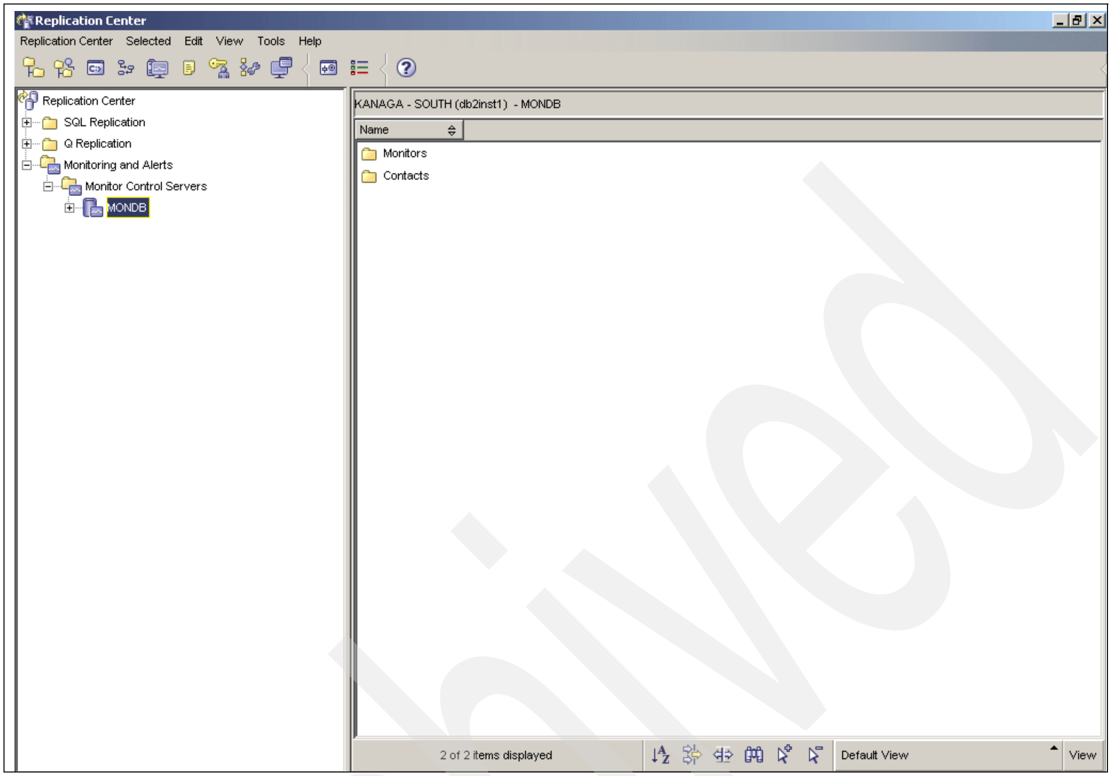


Figure 6-103 Monitor Control Servers - MONDB

Expand the **MONDB** folder as shown in Figure 6-104.

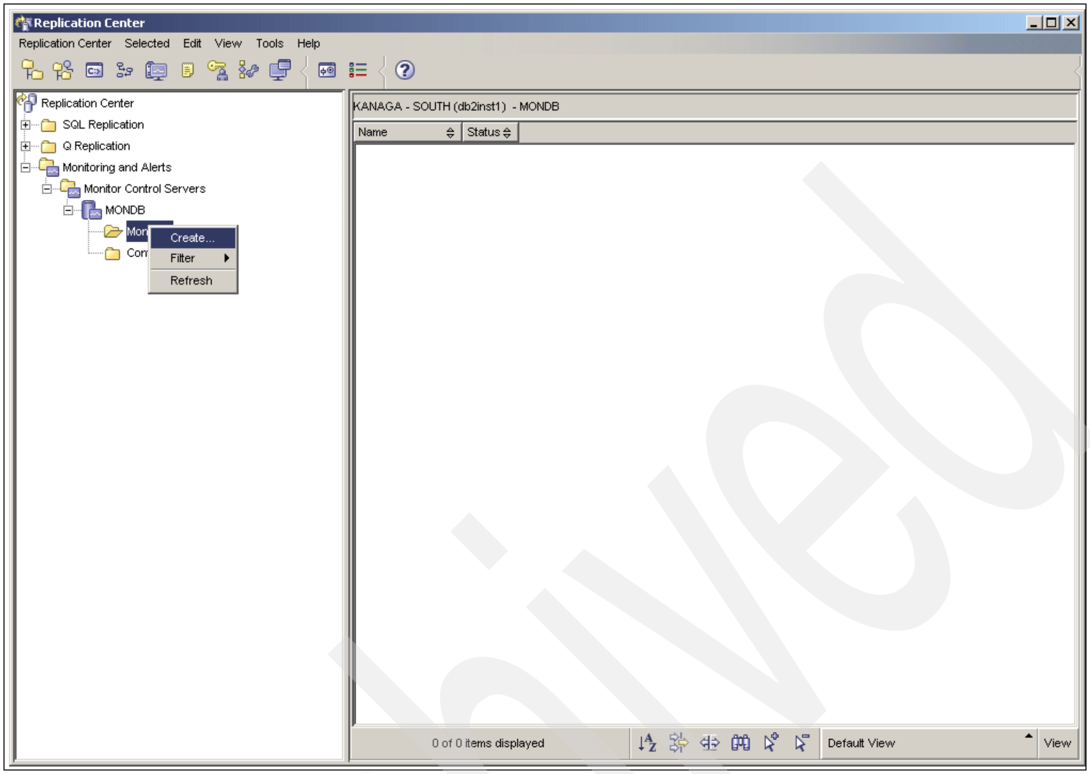


Figure 6-104 Create Monitors

Right-click the **Monitors** folder and select **Create...** (as shown in Figure 6-104) to proceed to the Create Monitor Wizard shown in Figure 6-105.

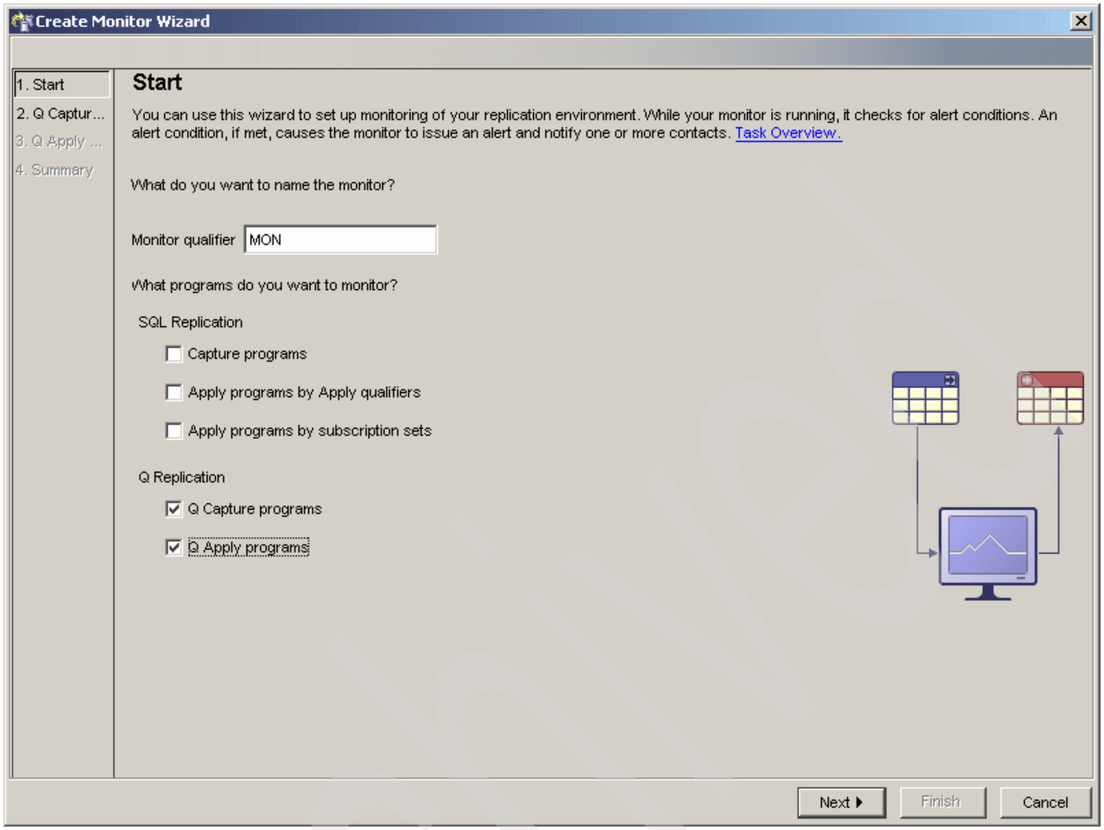


Figure 6-105 Create Monitor Wizard

A monitor qualifier is a grouping of monitors—one can have multiple monitors to monitor multiple programs. These multiple monitors are identified via the monitor qualifiers. We chose to monitor Q Capture and Q Apply programs in Q Replication using a single monitor qualifier MON entered in the Monitor qualifier field. We therefore selected the **Q Capture programs** and **Q Apply programs** settings under Q Replication, and clicked **Next** to select the alert conditions for Q Capture programs in Figure 6-106.

> **Note:** The Replication Alert Monitor has the schema name of ASN, which cannot be changed. Since the Replication Alert Monitor can monitor SQL replication as well as Q Replication, the control tables start with IBMSNAP instead of IBMQREP.

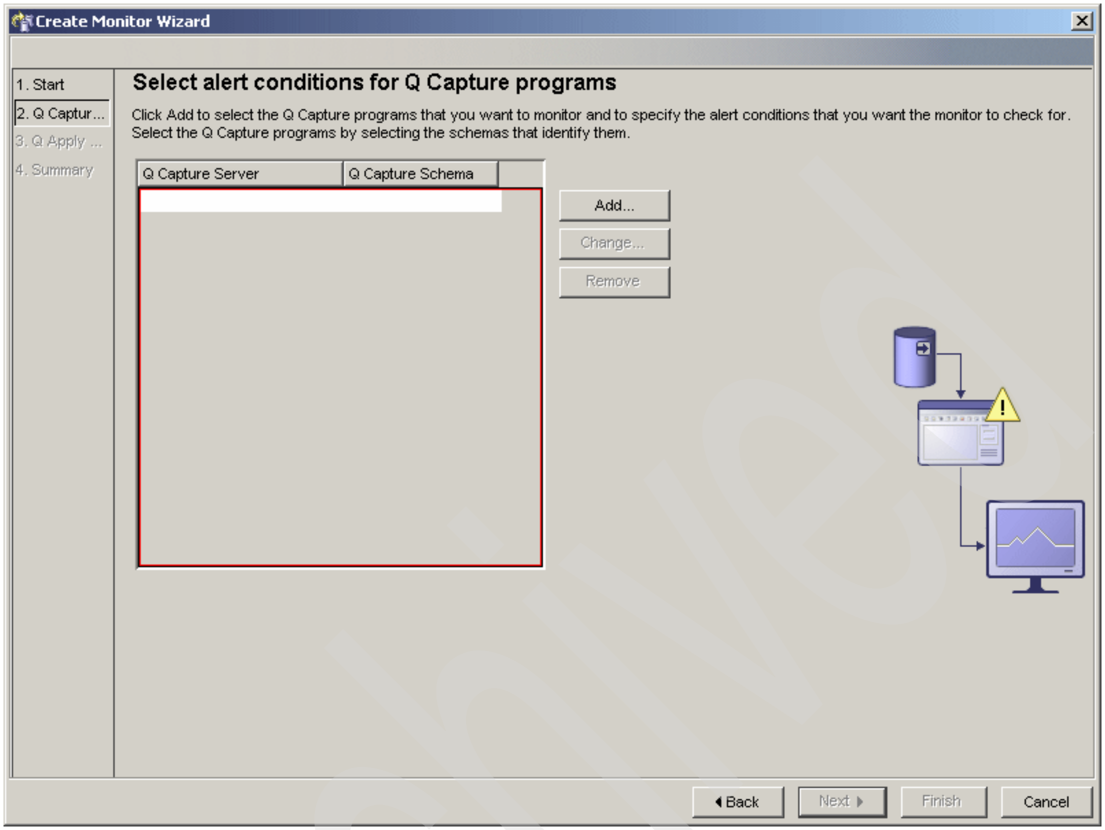


Figure 6-106 Select alert conditions for Q Capture programs

The Q Capture Server list is empty since this is the first time the create monitor has been invoked; therefore, candidate Q Capture Servers need to be added. This is done by clicking the **Add...** button in Figure 6-106 to select the Q Capture program and the alert conditions to be monitored, as shown in Figure 6-107.

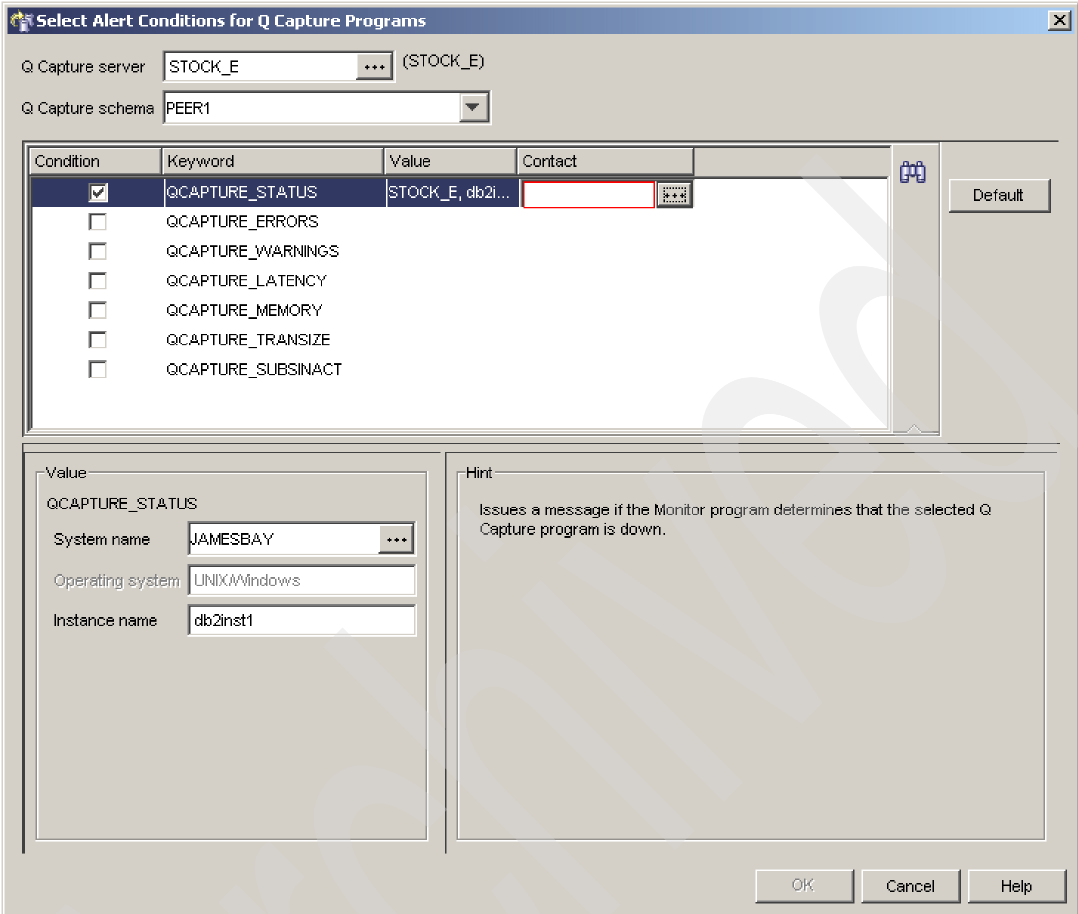


Figure 6-107 Select Q Capture program and alert conditions

Supply information about the Q Capture Server (STOCK_E reference B.7 in Table 6-1 on page 549), Q Capture schema (PEER1 reference B.22 in Table 6-3 on page 552), and check the alert conditions to monitor, such as QCAPTURE_STATUS. Some alert conditions require additional parameters, such as System name and Instance name.

Click the **...** button under the column Contact to specify the contact person for each alert condition, as shown in Figure 6-108.

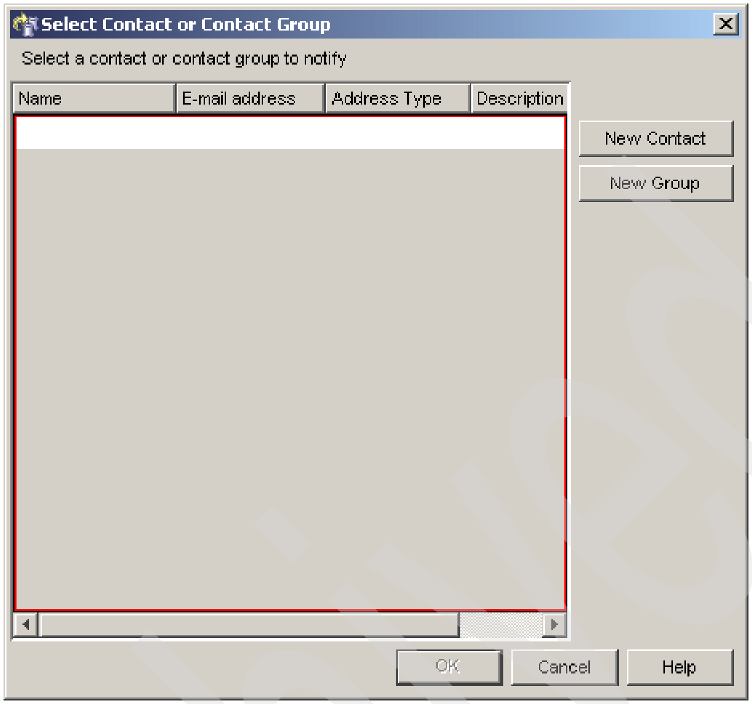


Figure 6-108 Select Contact or Contact Group

Since no contact information list is available, new contacts need to be created by clicking the **New Contact** button in Figure 6-108, which displays Figure 6-109.

> **Note:** Contact groups may also be created here, which are collections of contacts that receive the same alerts.

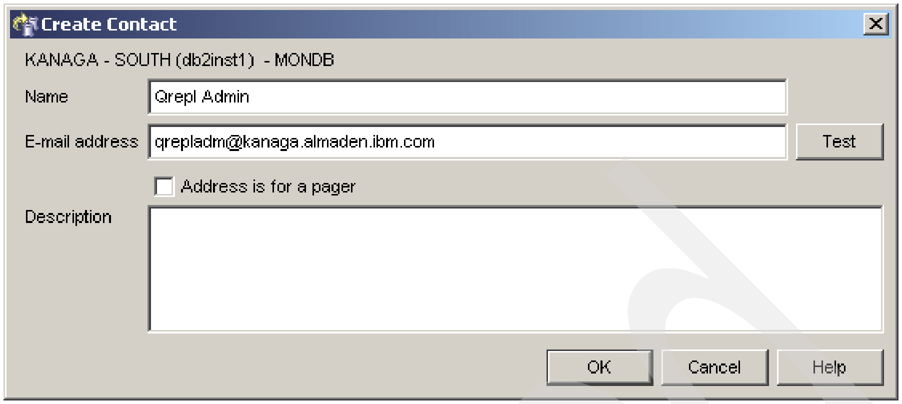


Figure 6-109 Create Contact

Specify any name for the contact in Figure 6-109 in the Name field (reference M.5d in Table 6-4 on page 553), and provide an e-mail address of the contact person in the E-mail address field (reference M.5e in Table 6-4 on page 553), where the alert message should be delivered by the Replication Alert Monitor.

The Test button provides a quick way of determining whether the mail server is running or the address entered is correct and can be used.

Check the **Address is for a pager** if appropriate.

Click **OK** to generate the script for defining the contact information specified, as shown in Figure 6-110.

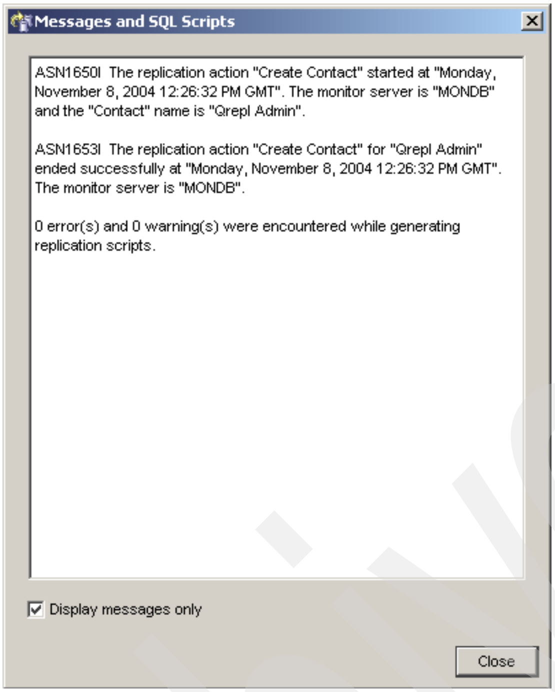


Figure 6-110 Message and SQL Scripts

Check that the errors are 0. Click **Close** to display Figure 6-111 to decide whether to execute the script or save it for later execution.

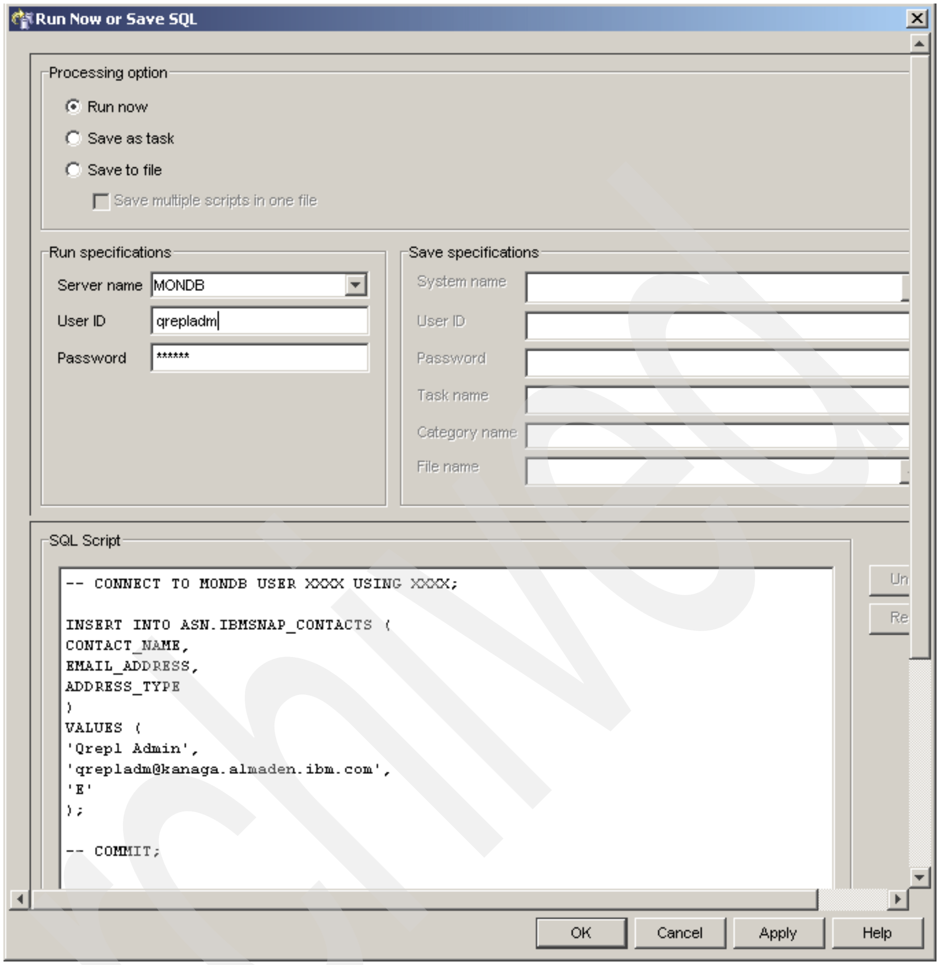


Figure 6-111 Run Now or Save SQL

Select the **Run now** option and click **OK** to display the message that the scripts to generate the contact information objects executed successfully, as shown in Figure 6-112.

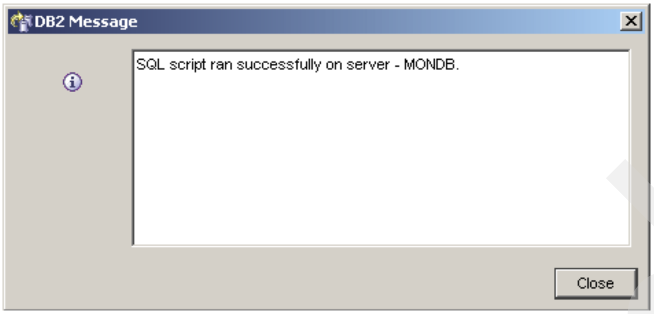


Figure 6-112 SQL script ran successfully message

Click **Close** to continue to Figure 6-113, which shows the newly added contact information.

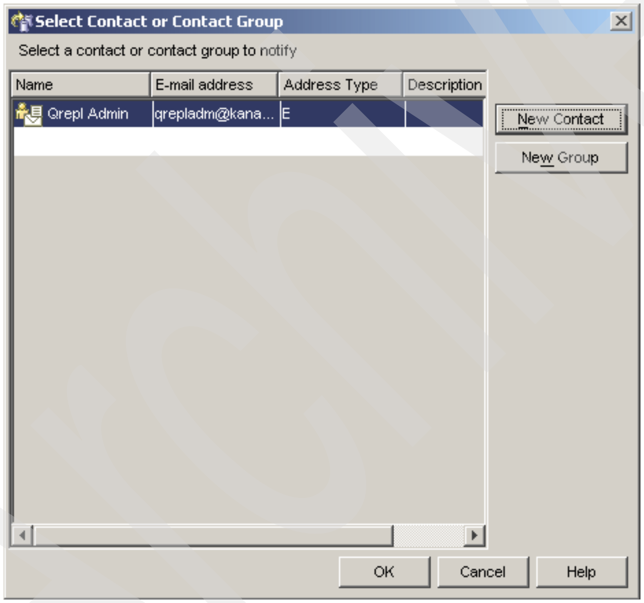


Figure 6-113 Select Contact or Contact Group

Select the contact just created and click **OK** to proceed to Figure 6-114, which lists the rest of the alert conditions for possible selection.

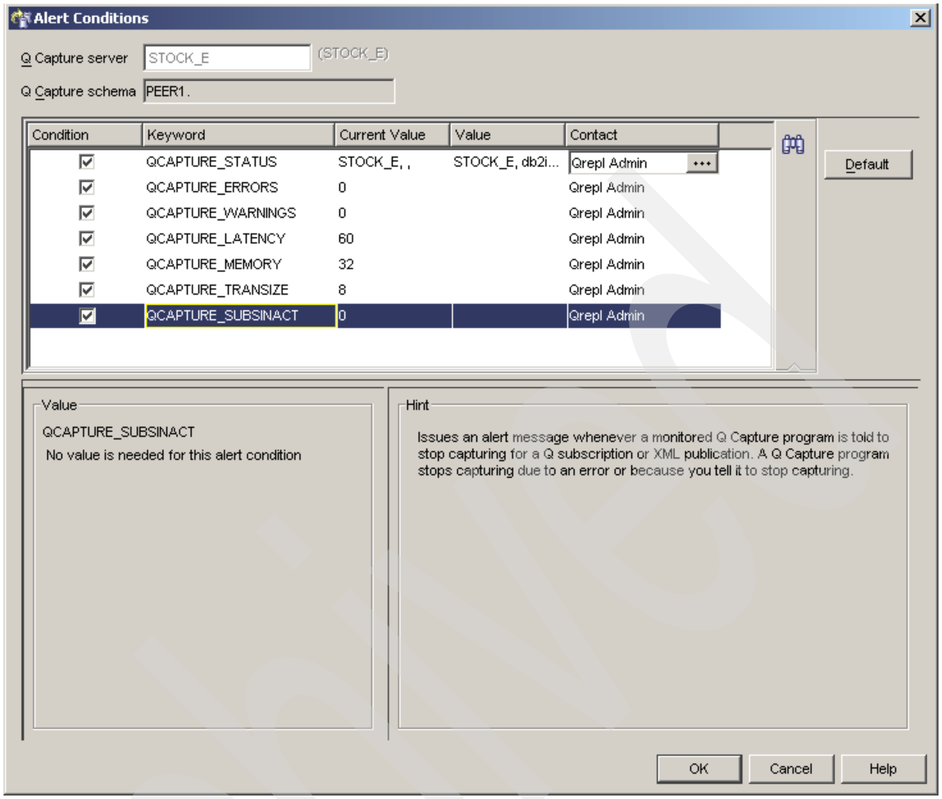


Figure 6-114 Alert Conditions

We selected all the alert conditions and chose the same contact name created earlier for each of them. Click **OK** to proceed to Figure 6-115.

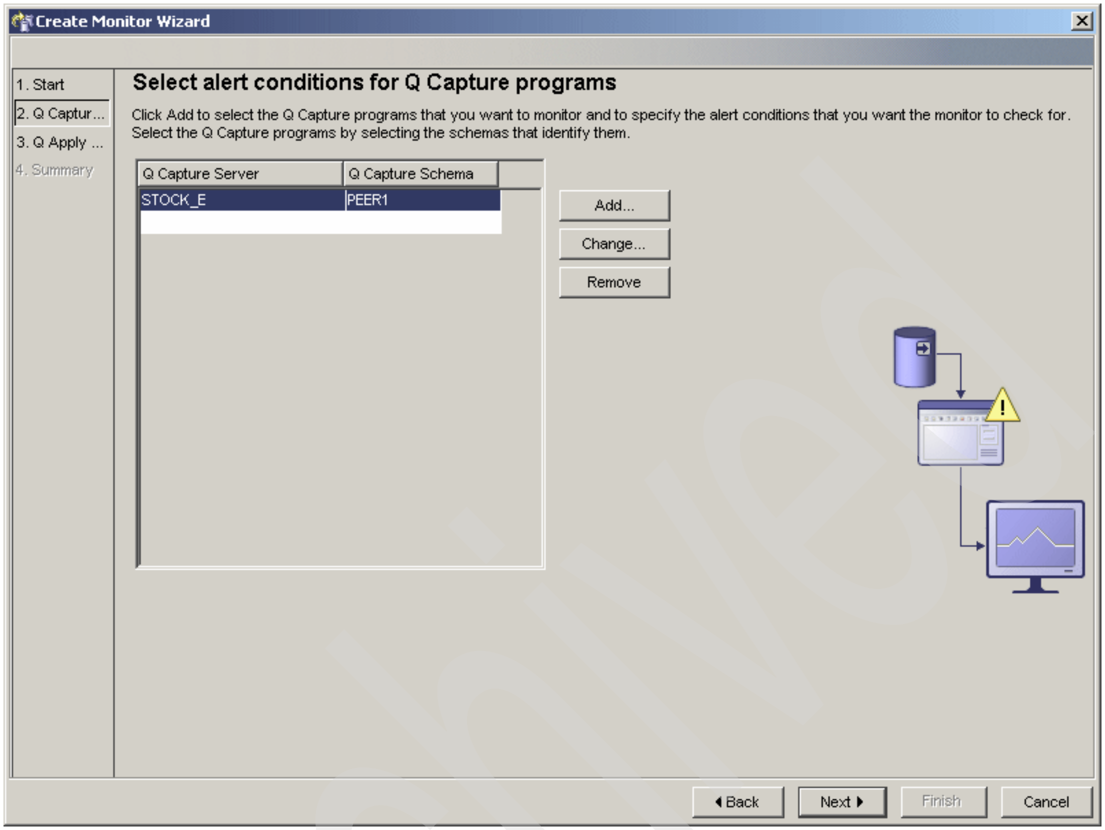


Figure 6-115 Select alert conditions for Q Capture programs

Additional Q Capture Servers may be added in Figure 6-115 by clicking the **Add...** button. Since we needed to monitor both the Q Capture servers in our peer-to-peer environment, we clicked **Add...** to include the STOCK_W Q Capture Server, as shown in Figure 6-116.

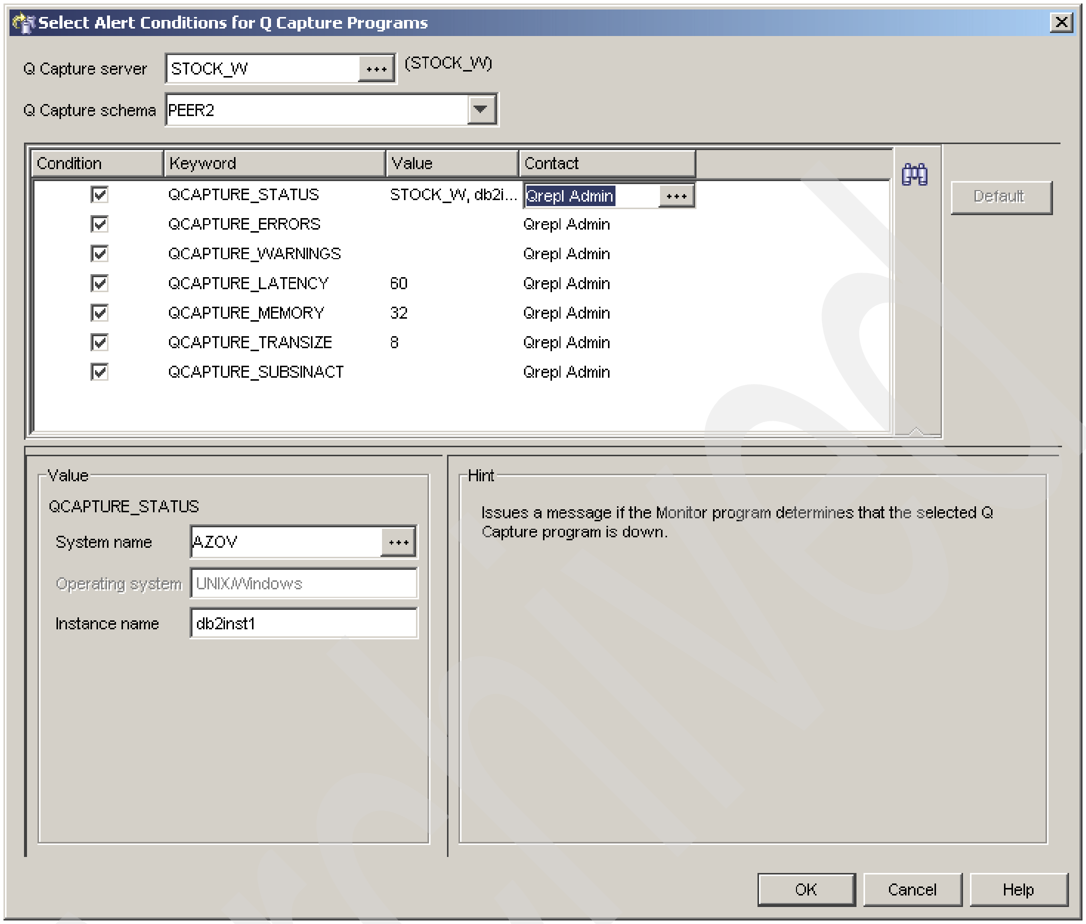


Figure 6-116 Select Alert Conditions for Q Capture Programs

Supply information for the STOCK_W Q Capture Server and the PEER2 Q Capture schema, and select all the alert conditions and associate the same contact name defined earlier with each one of them. The System name (AZOV.ALMADEN.IBM.COM) and Instance name (db2inst1) information need to be provided. Click **OK** to proceed to Figure 6-117, which completes the monitoring configuration for the Q Capture programs on the STOCK_E and STOCK_W database servers.

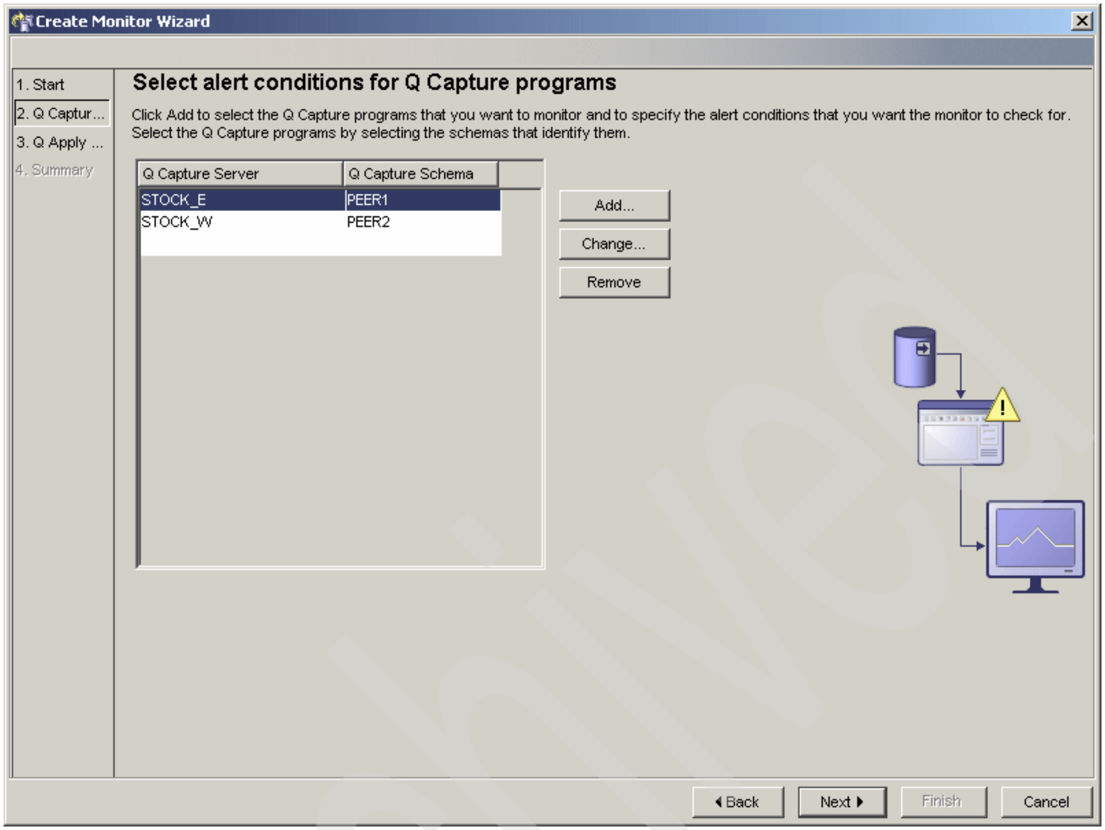


Figure 6-117   Select alert conditions for Q Capture programs

Click **Next** to set the alert conditions for the Q Apply programs.

Step7c: Create alert conditions and contact list for Apply

The same series of steps performed for setting the alert conditions for the Q Capture program as described in Figure 6-106 on page 678 through Figure 6-117 should also be performed for the Q Apply program, as shown in Figure 6-118 through Figure 6-122 on page 693.

Once all the alert conditions are provided, the Create Monitor Wizard generates a summary of all the information provided, as shown in Figure 6-123 on page 694.

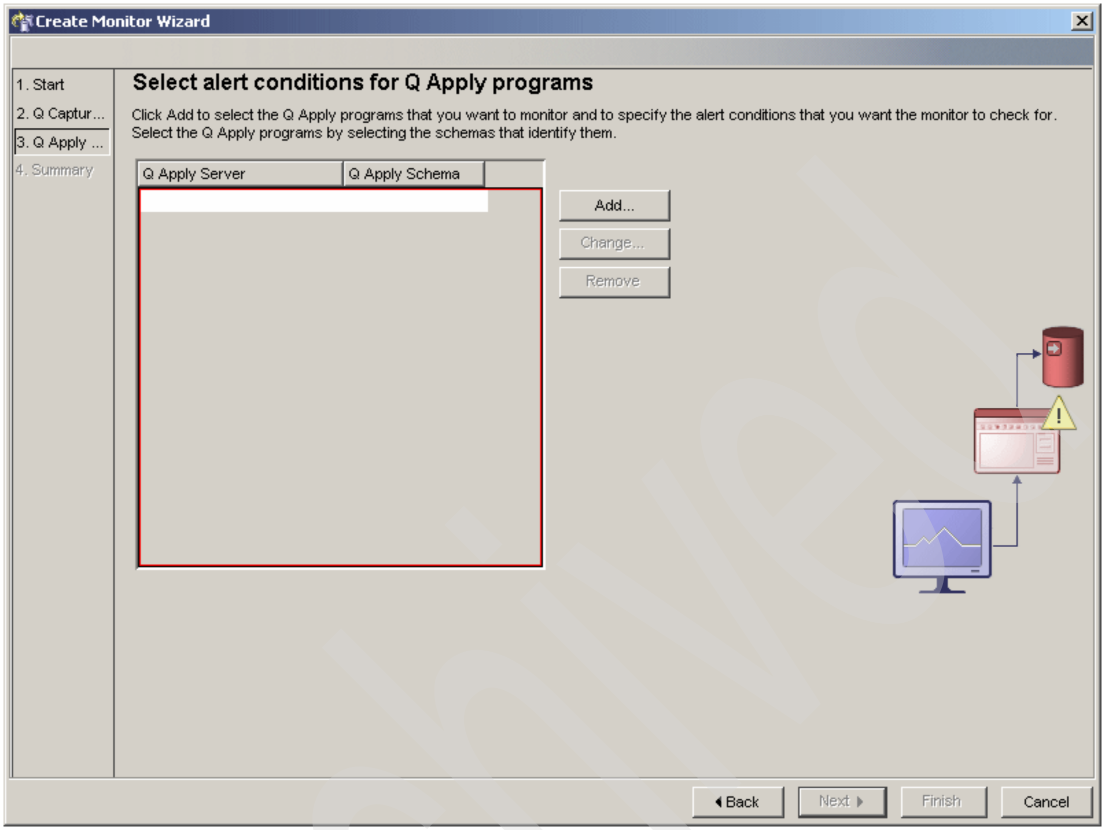


Figure 6-118 Create Monitor Wizard - Select alert conditions for Q Apply programs

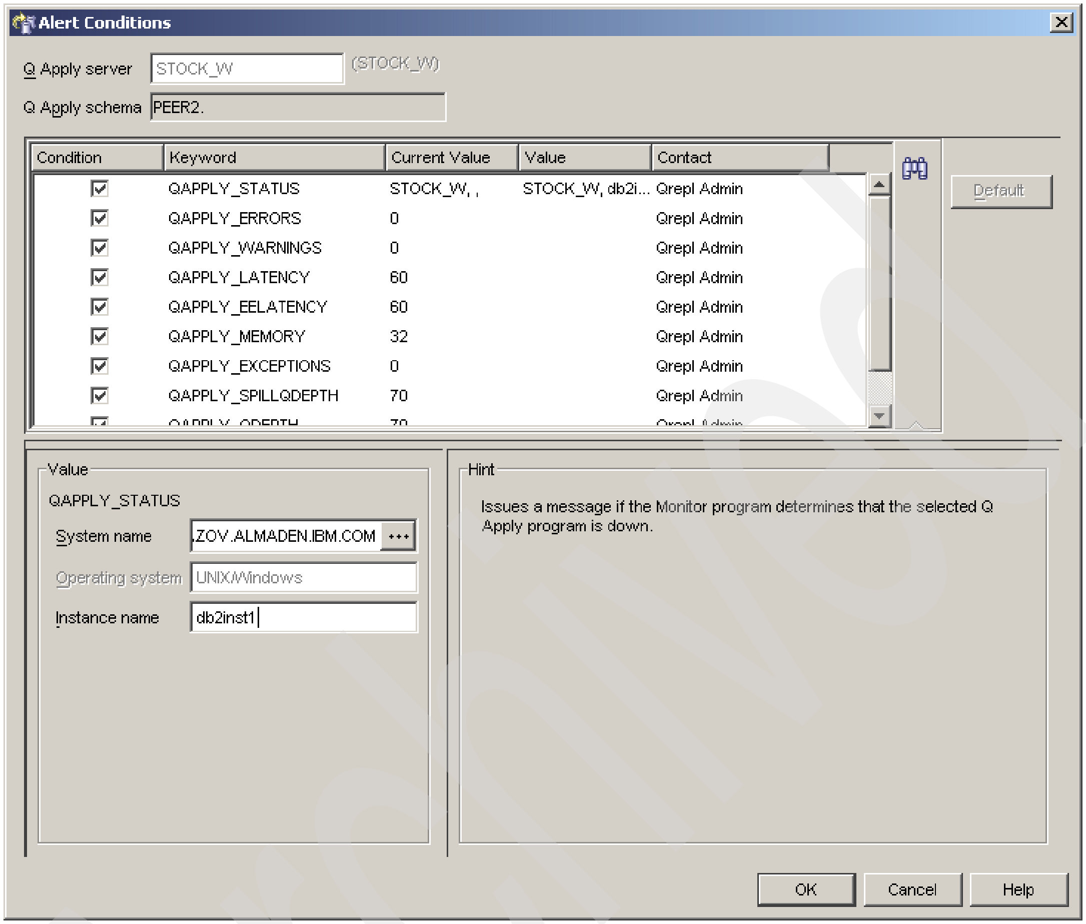


Figure 6-119 Alert Conditions

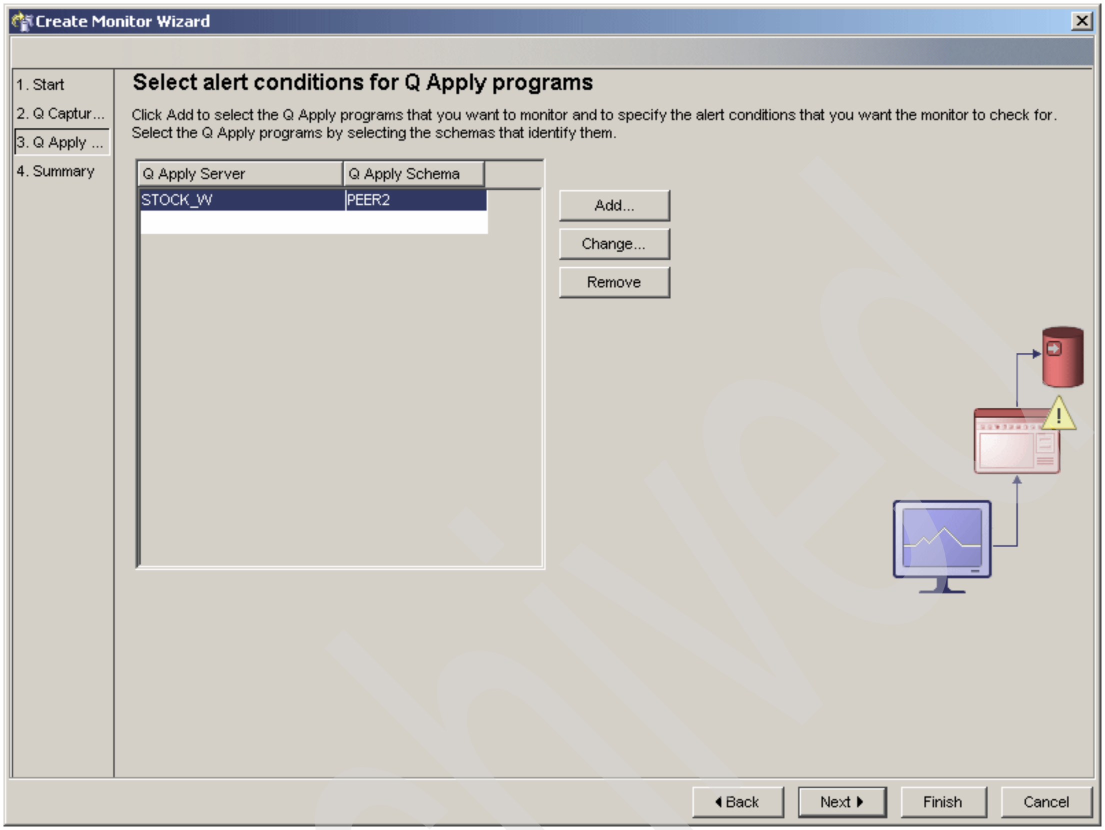


Figure 6-120   Create Monitor Wizard - Select alert conditions for Q Apply programs

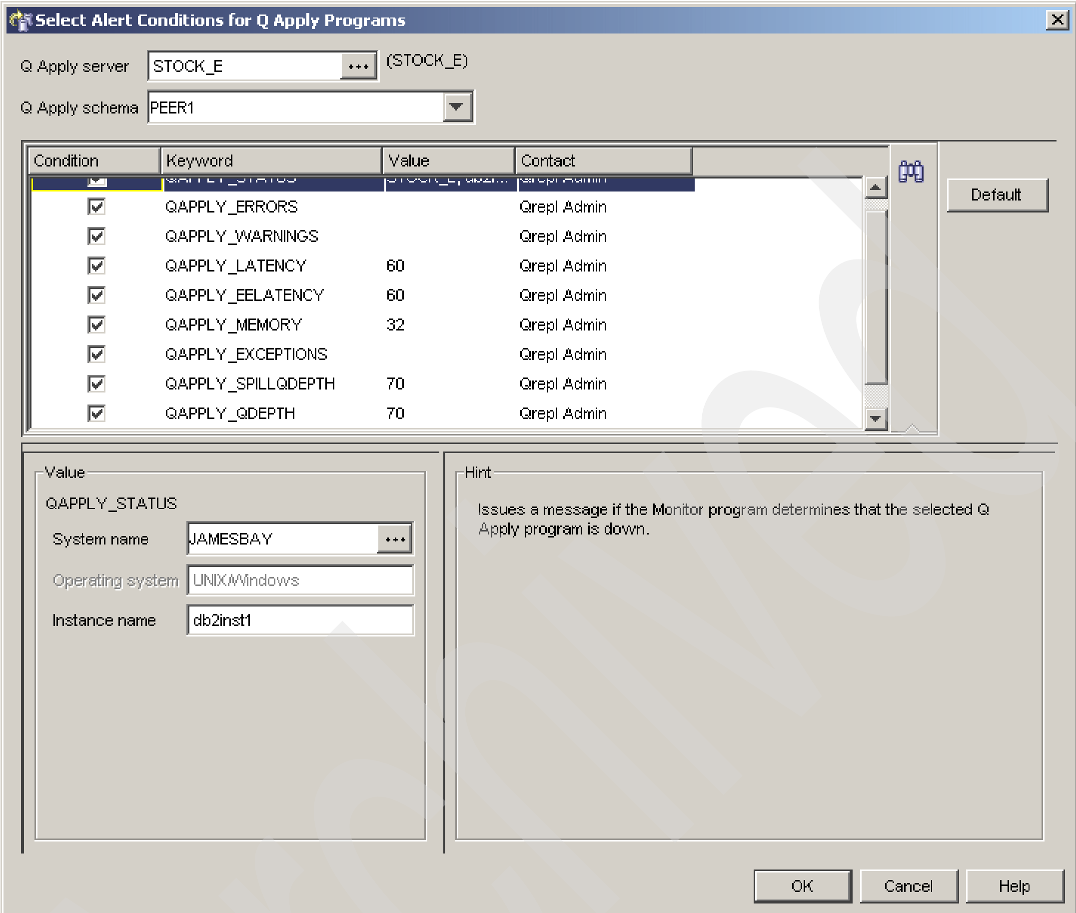


Figure 6-121 Select Alert Conditions for Q Apply Programs

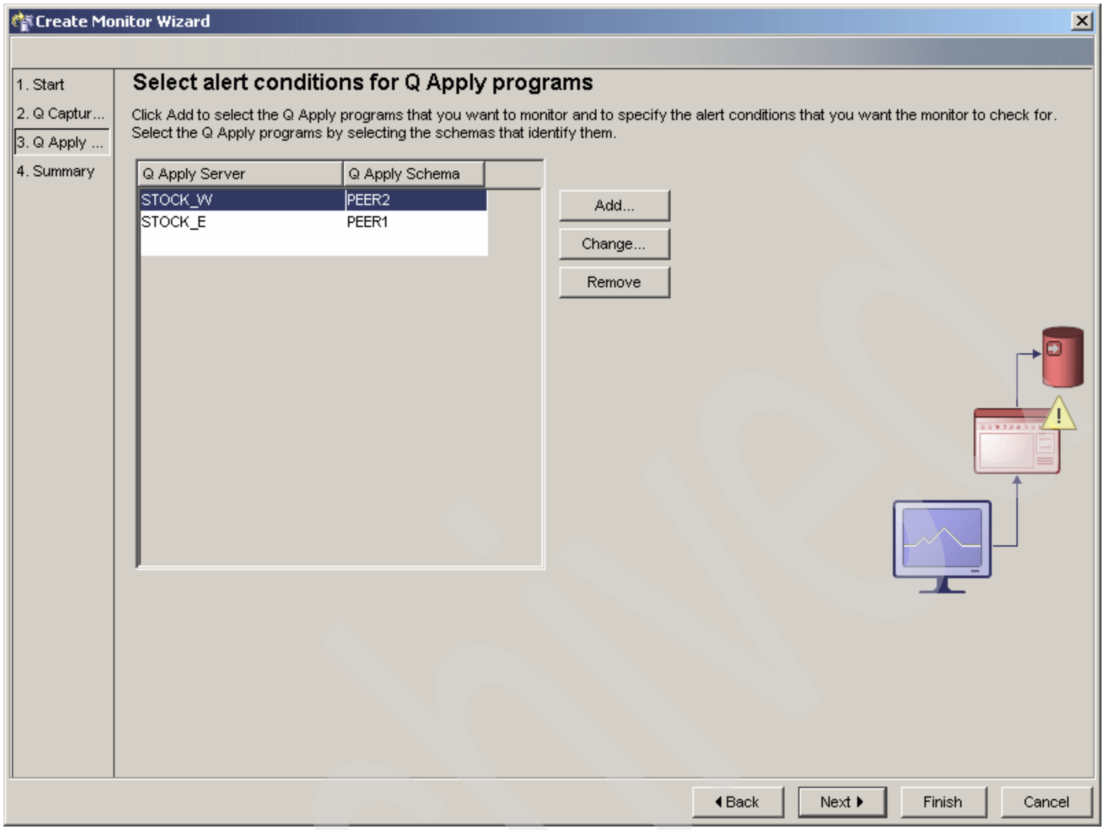


Figure 6-122 Create Monitor wizard - Select alert conditions for Q Apply programs

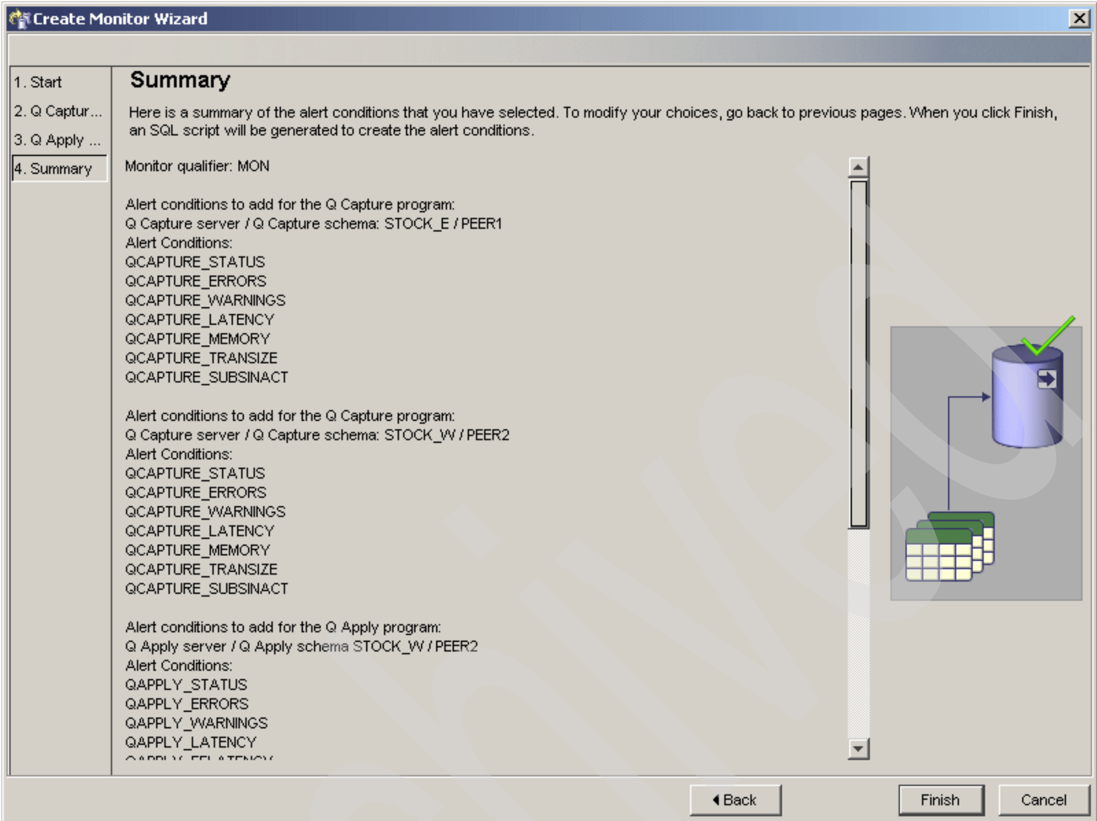


Figure 6-123 Summary

The summary of the alert conditions for the Q Capture and Q Apply programs shown in Figure 6-123 should be reviewed for correctness. Click the **Back** button to correct any errors, or click **Finish** to proceed to generating the script for defining the various alert condition objects specified, as shown in Figure 6-124.

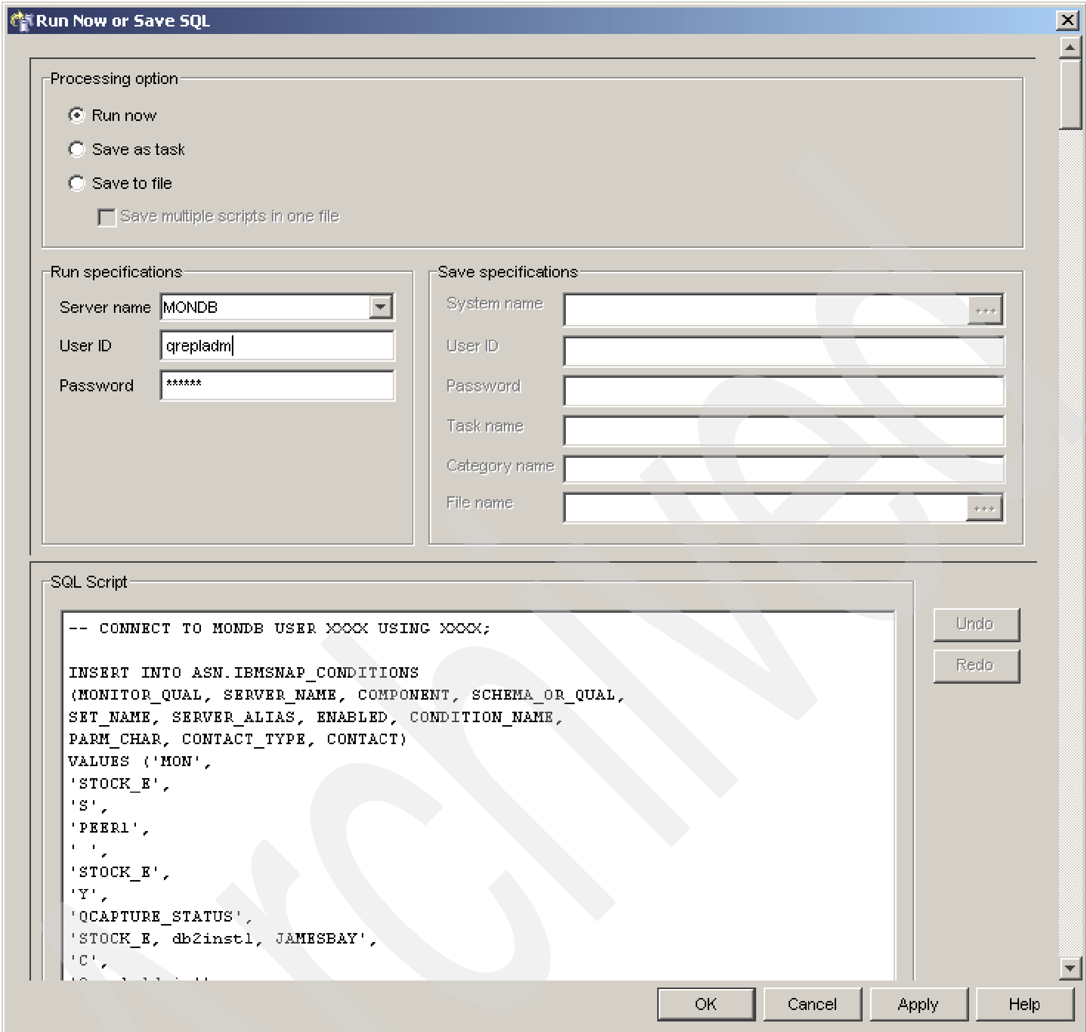


Figure 6-124 Run Now or Save SQL

Select the **Run now** option and click **OK** to display the message that the scripts executed successfully, as shown in Figure 6-125.

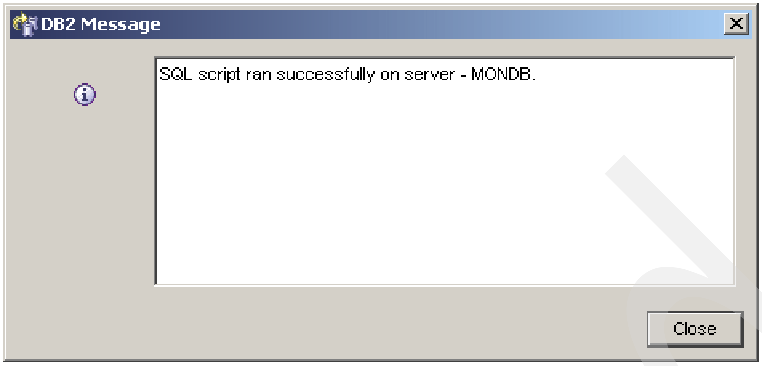


Figure 6-125 SQL script ran successfully message

Figure 6-126 shows the four monitored servers that we defined by navigating to the ASN folder for the MONDB Monitor Control Server. Figure 6-127 shows the contact information created in the Contacts folder.

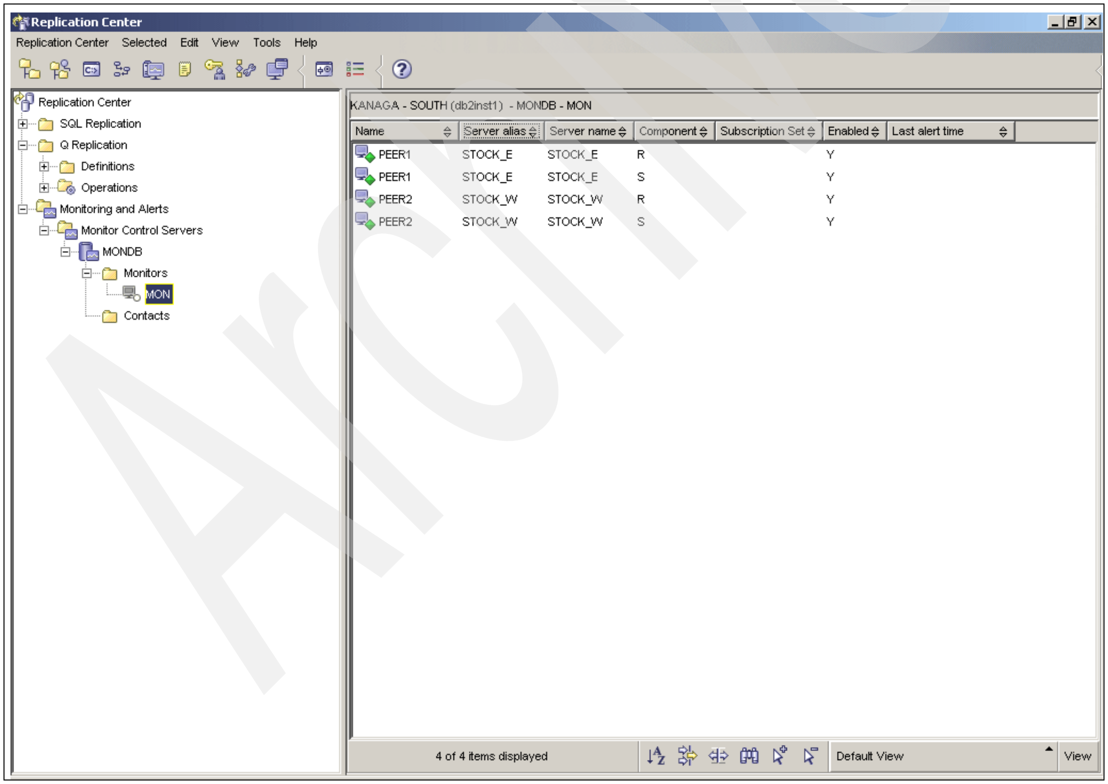


Figure 6-126 Monitored servers

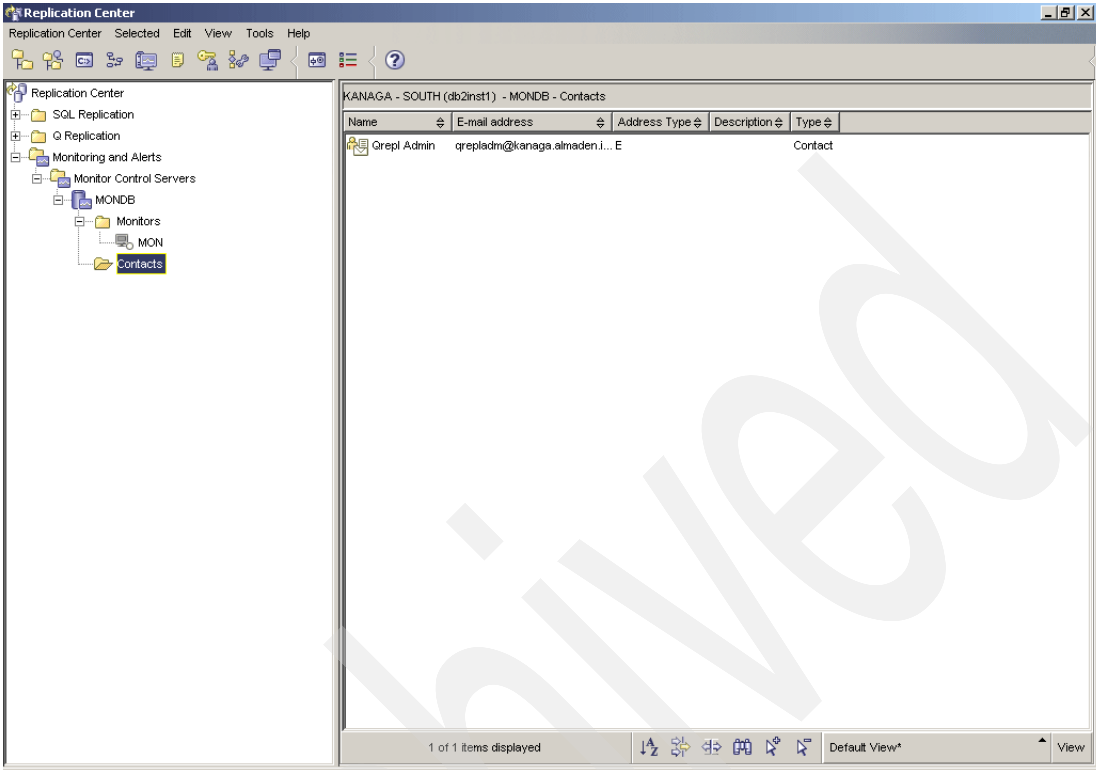


Figure 6-127 Replication Center - Contacts

Step 7d: Start the Alert Monitor

The Replication Alert Monitor can be started by selecting and right-clicking the **MON** monitor qualifier and selecting **Start Monitor**, as shown in Figure 6-128. This displays Figure 6-129, which allows one to configure various runtime parameters.

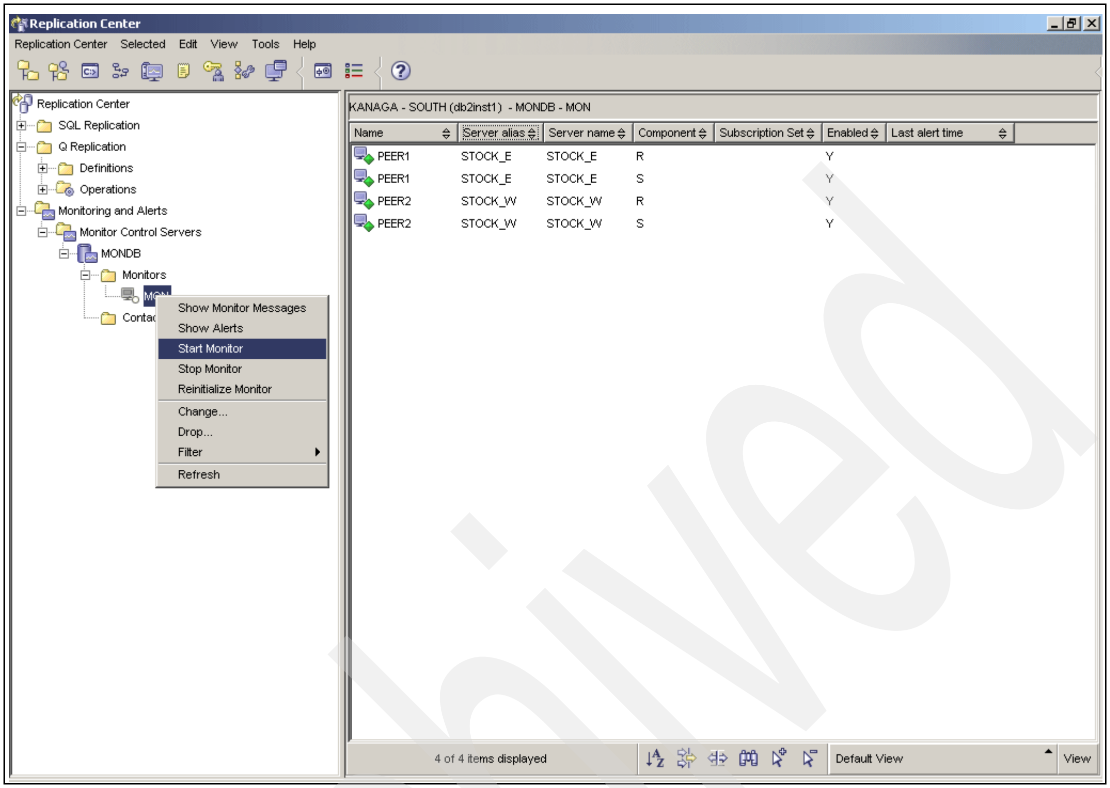


Figure 6-128 Start Monitor

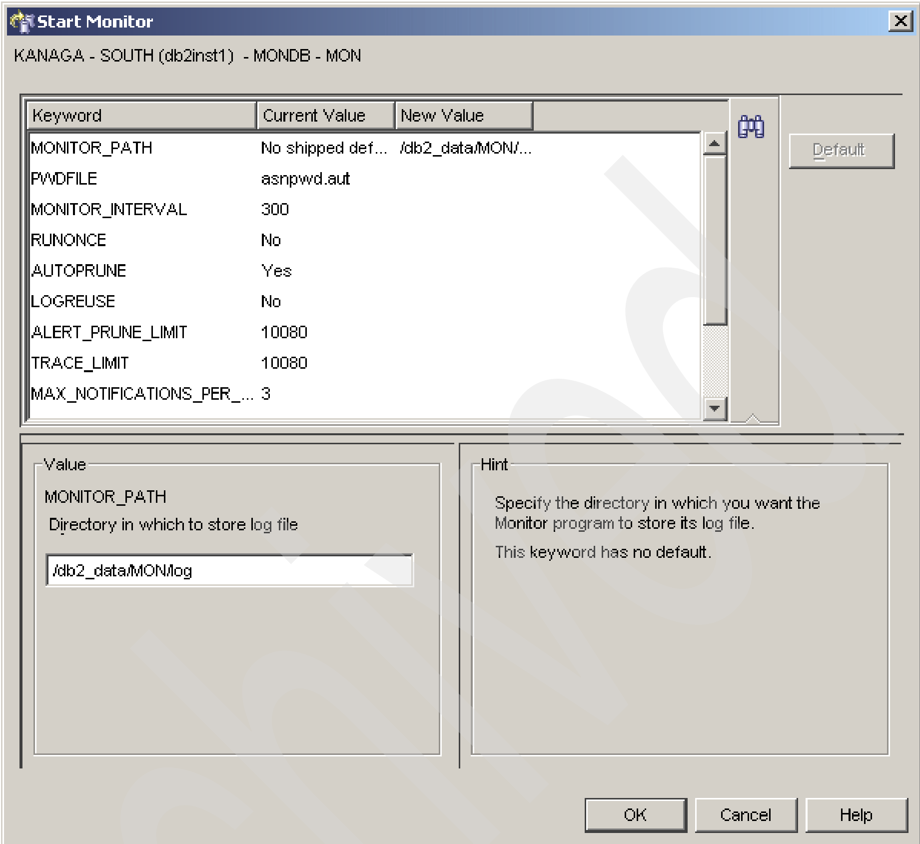


Figure 6-129 Start Monitor parameters

We specified the MONITOR_PATH directory to be /db2_data/MON/log where the Replication Alert Monitor programs store the log file, and let the other values default.

Since we were running the Replication Alert Monitor programs from a remote system, we needed to have a password file in the MONITOR_PATH directory that contained the remote database name, user ID, and password. This file should contain this information for all the remote databases, and is used by the monitor programs to connect to the remote server. The name of the password file (asnpwd.aut) is specified in the PWDFILE field. Example 6-48 shows the creation contents of the asnpwd.aut password file, which includes the user ID/password for accessing EAST_DB, WEST_DB, STOCK_E, and STOCK_W databases.

Example 6-48 Create password file

```
cd /db2_data/MON/log
asnpwd init
asnpwd add alias EAST_DB id qrepladm password "xxxx"
asnpwd add alias WEST_DB id qrepladm password "xxxx"
asnpwd add alias STOCK_E id qrepladm password "xxxx"
asnpwd add alias STOCK_W id qrepladm password "xxxx"
```

Attention: Ensure that these databases are cataloged on the monitor server Kanaga.

Make sure that these databases are catalogued on the monitor server.

Example 6-49 Catalog these databases on the monitor server Kanaga

```
db2 catalog tcpip node EAST_DB remote 9.1.39.79 server 50000
db2 catalog db EAST_DB at node EAST_DB
db2 catalog tcpip node WEST_DB remote 9.1.39.89 server 50000
db2 catalog db WEST_DB at node WEST_DB
db2 catalog tcpip node STOCK_E remote 9.1.39.79 server 50000
db2 catalog db STOCK_E at node EAST_DB
db2 catalog tcpip node STOCK_W remote 9.1.39.89 server 50000
db2 catalog db STOCK_W at node EAST_DB
db2 terminate
```

Click **OK** in Figure 6-129 on page 699 to generate the command for setting the Replication Alert Monitor parameters for immediate execution or to be saved for later execution, as shown in Figure 6-130.

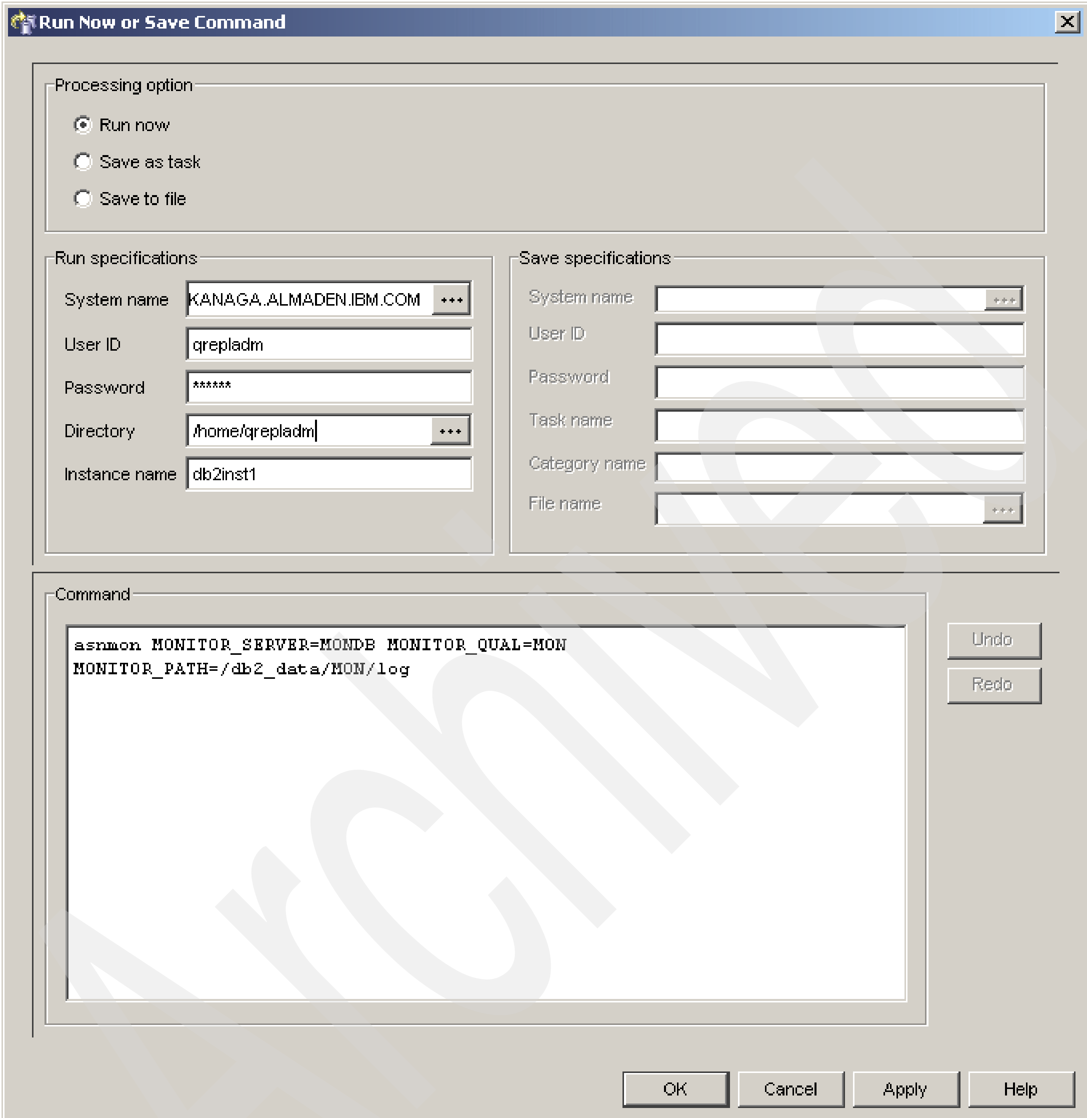


Figure 6-130 Run Now or Save Command

We selected **Run now** and clicked **OK** for the command to be executed immediately. Figure 6-131 displays the message indicating that the command was submitted.

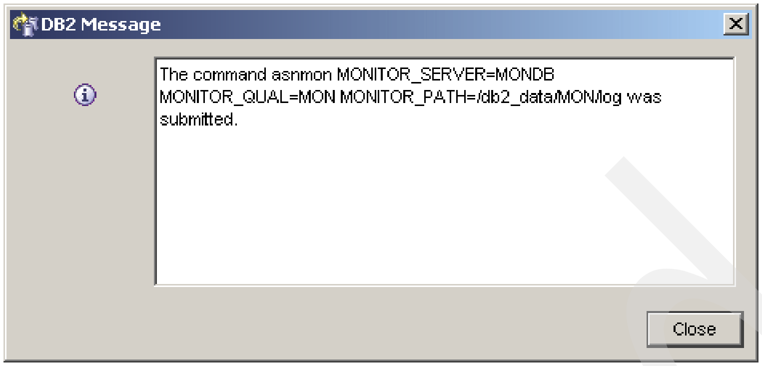


Figure 6-131 Command submitted successfully message

To determine whether the command was executed successfully, the status of the Replication Alert Monitor may be determined by one of the following three methods.

- Issue the AIX **ps** command as shown in Example 6-50.

Example 6-50 ps command

```
$ ps -ef | grep asn
qrepladm 36138     1   0   Nov 18      -  1:40 asnmon monitor_server=MONDB
monitor_qual=MON pwdfile=asnpwd.aut monitor_path=/db2_data/MON/log
```

- Select the **Monitors** folder in the Replication Center, as shown in Figure 6-132 on page 703.
- Check the message in the Replication Alert Monitor log file (/db2_data/MON/log directory), as shown in Example 6-51.

Example 6-51 Successful message in the Replication Alert Monitor log file

```
Monitor log file: /db2_data/MON/log/db2inst1.MONDB.MON.MON.log
2004-11-18-16.58.02.944151 <_asnMonitor> ASN5101I  MONITOR "MON" : "Initial".
The Replication Alert Monitor program started successfully.
2004-11-18-16.58.02.991213 <WorkerMain> ASN5102I  MONITOR "MON" :
"WorkerThread"
The Replication Alert Monitor program initialized successfully and is
monitoring "64" alert conditions
```

Step 7e: View the alerts

When the alert monitor is running, the status of the monitor qualifier (MON) will indicate `Up` in the Status column, as shown in Figure 6-132.

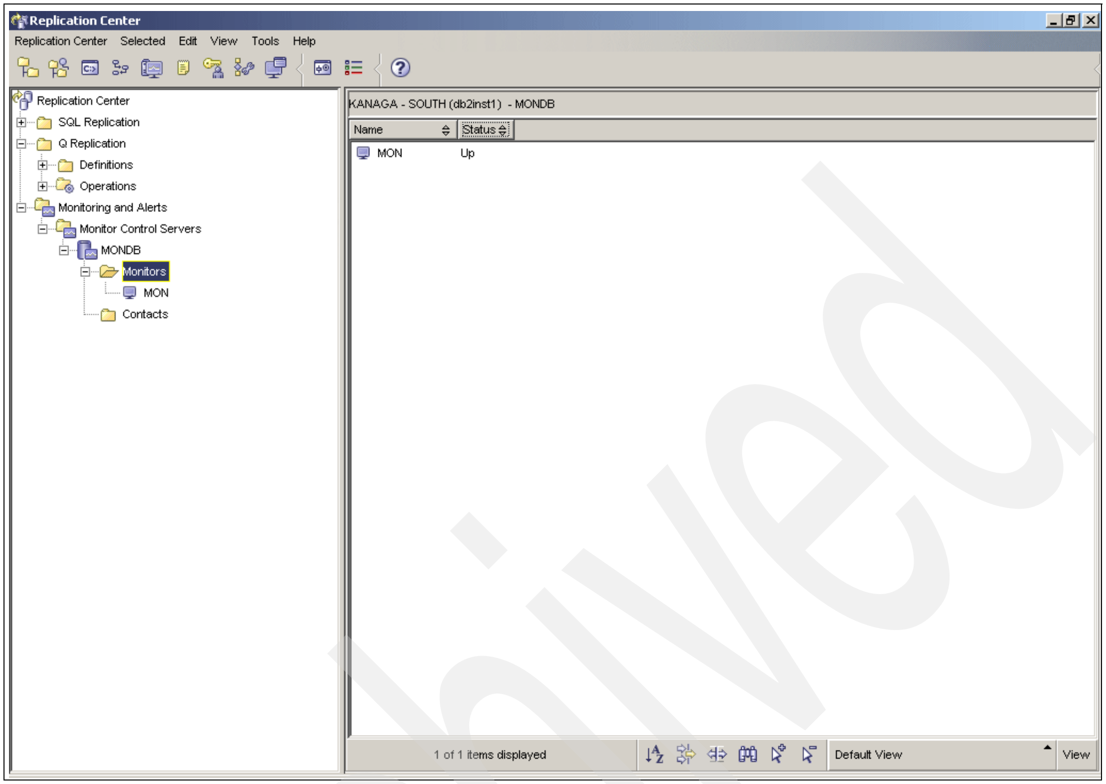


Figure 6-132 Monitor status

Highlighting the MON folder in the navigation pane provides an overview of the servers that have alerts, as shown in Figure 6-133.

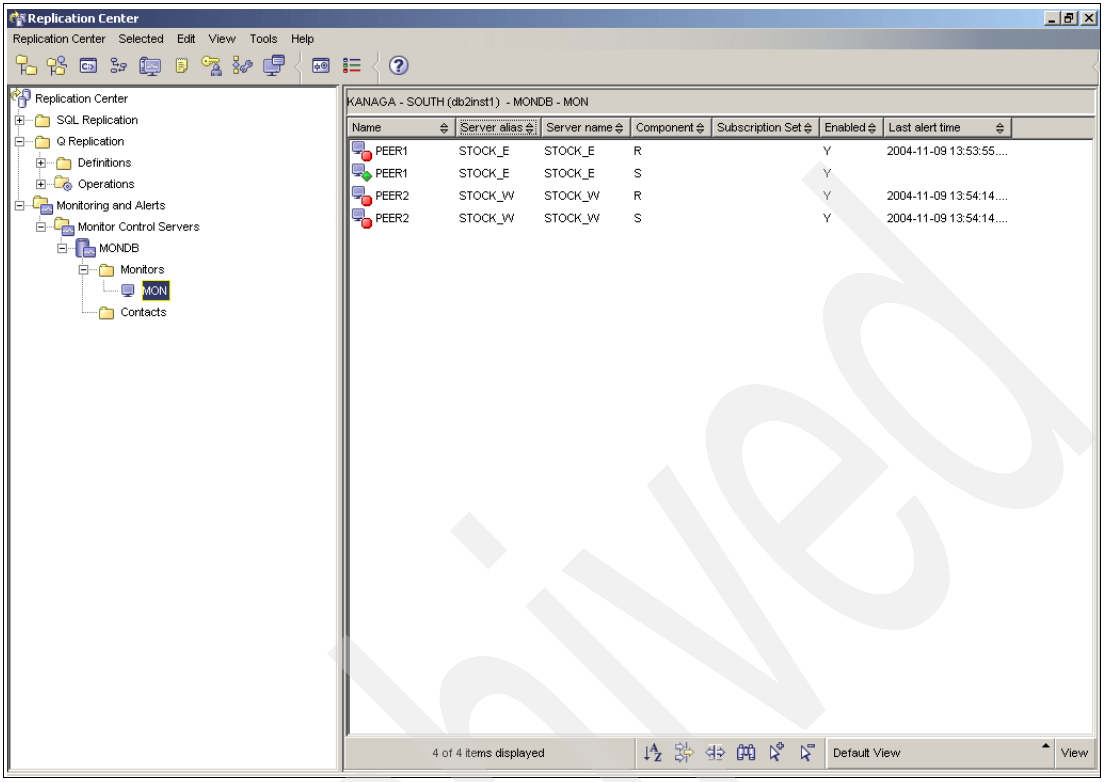


Figure 6-133 MON monitor qualifier alerts overview

In Figure 6-133, the four database servers are displayed in the right-hand display pane with colored icons next to them.

- A red button indicates that there is at least one alert.
- A green diamond indicates that there are no alerts.

The Last alert time column indicates the most recent timestamp of an alert. This timestamp should be monitored to determine if alerts are being generated—a frequently updating timestamp value is an indication that alerts being generated, while an unchanging timestamp indicates otherwise.

> **Note:** Even after correcting the alerts, the red button will not turn into a green diamond unless there are no more alert messages in the IBMSNAP_ALERTS table.

To view the alerts detected, select the server (with the red button) of interest and right-click it, and select **Show Alerts**, as shown in Figure 6-134. This will display Figure 6-135.

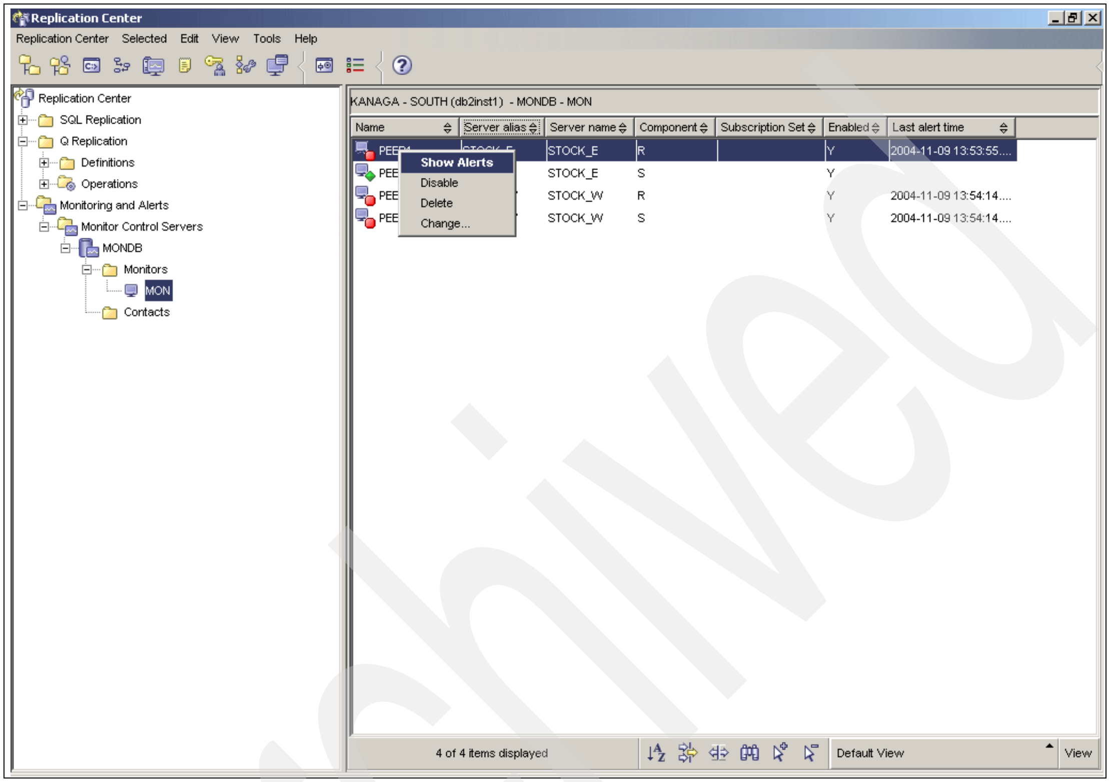


Figure 6-134 Replication Center - Show Alerts

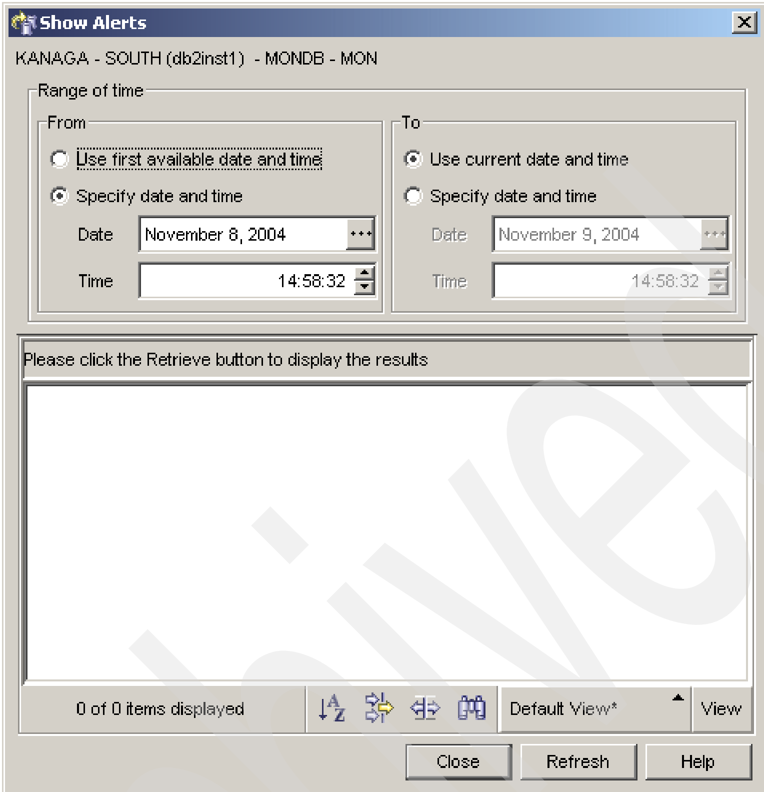


Figure 6-135 Show Alerts - Filter criteria

Figure 6-135 provides a filtering capability to limit the number of alerts to be viewed. After specifying the filter criteria, click the **Refresh** button to view the alerts conforming to the filtering criteria, as shown in Figure 6-136.

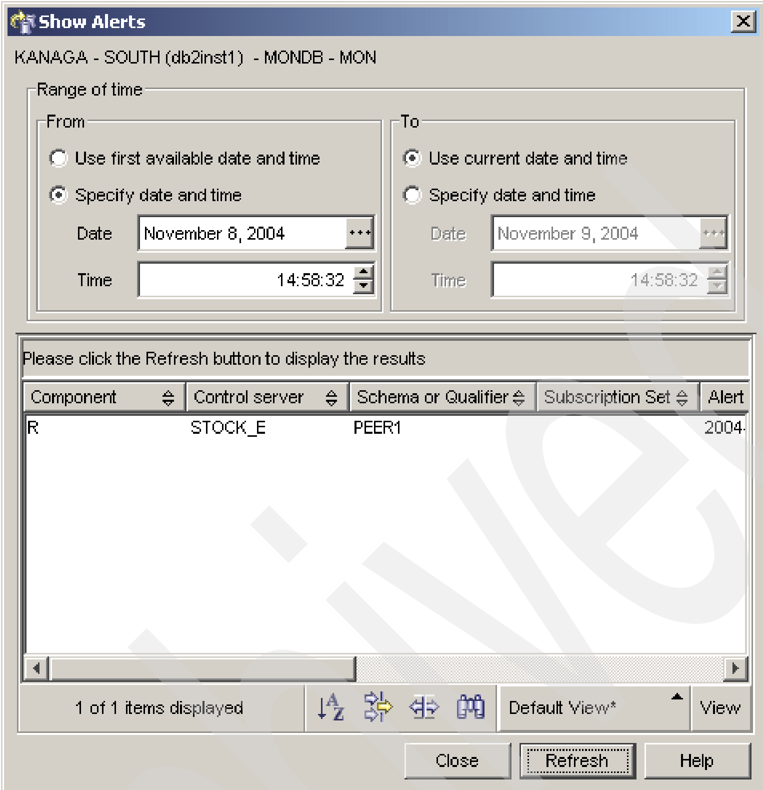


Figure 6-136 Show Alerts

Figure 6-136 shows the alert details in the lower part of the window.

All the alerts may also be viewed by querying the IBMSNAP_ALERTS table, as shown in Example 6-52.

Example 6-52 Show alerts through SQL

```
db2 CONNECT TO MONDB

   Database Connection Information

 Database server        = DB2/6000 8.2.0
 SQL authorization ID   = QREPLADM
 Local database alias   = MONDB

db2 "select monitor_qual as MON, occured_time, component as COMP, server_name
as SRV_NAME, server_alias as SRC_ALIAS, schema_or_qual as SCHEMA,
condition_name, alert_message from asn.ibmsnap_alerts order by occurred_time
desc" | more
```

```
MON OCCURED_TIME             COMP SRV_NAME SRV_ALIAS SCHEMA  CONDITION_NAME
-------------------------------------------------------------------------------
MON 2004-11-18-16.58.28.820031 R    STOCK_E STOCK_E  PEER1    QAPPLY_EXCEPTIONS

ALERT_MESSAGE
---------------
ASN5173W  MONITOR "MON" : "WorkerThread". An exception was found for the Q
Apply program. The server is "STOCK_E". The schema is "PEER1". The receive
queue is "PEER2_TO_PEER1_Q". The reason code is "P2PVERLOSER". The SQLCODE is
"0". The SQLSTATE is "". The SQLERRMC is "message old row version loses, no new
row found ".
```

6.5.9 Step 7: Set up monitoring environment using commands

The monitoring environment may be performed using GUI, as described in 6.5.8, “Step 7: Set up monitoring environment using the GUI” on page 665; or by using ASNCLP commands from the Unix command line. This section describes the configuration of the same monitoring environment 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 6-53.

Example 6-53 Set up the Java environment for user qrepladm

```
PATH=$PATH:/usr/java14/bin


CP=/db2_data/db2inst1/sqllib/java/Common.jar

CP=$CP:/db2_data/db2inst1/sqllib/tools/db2cmn.jar

CP=$CP:/db2_data/db2inst1/sqllib/tools/db2replapis.jar

CP=$CP:/db2_data/db2inst1/sqllib/tools/db2qreplapis.jar

CP=$CP:/db2_data/db2inst1/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 setup was successful. Enter quit to exit the Repl> command line, as shown in Example 6-54.

Example 6-54 asnclp setup configuration

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

> **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.

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 6-55.

Example 6-55 Execute commands from a file

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

The ASNCLP commands in the following sections are categorized according to the following tasks

The steps involved are mostly the same as with the GUI approach, as follows:

1. Create the Replication Alert Monitor control tables.
2. Create the Replication Alert Monitor contact.
3. Create the Replication Alert Monitor alert conditions.
4. Start the Replication Alert Monitor program.

These steps are described in detail in the following subsections.

Create the Replication Alert Monitor Control Tables

Example 6-56 lists the ASNCLP commands for creating the Replication Alert Monitor Control Tables on the Monitor control server.

Example 6-56 Create Replication Alert Monitor Control Tables in MONDB

```
SET SERVER MONITOR TO DB MONDB ID qrepladm PASSWORD "xxxxxx" ;
SET RUN SCRIPT NOW STOP ON SQL ERROR ON ;
```

```
CREATE CONTROL TABLES FOR MONITOR CONTROL SERVER;
```

The Replication Alert Monitor creates ten control tables, as shown in Example 6-57.

Example 6-57 Configuring replication alert monitor control tables

```
$ db2 list tables for schema ASN

Table/View      Schema Type  Creation time
---------------------- ----- --------------------------
IBMSNAP_ALERTS      ASN T     2004-10-25-10.35.12.009521
IBMSNAP_CONDITIONS     ASN T     2004-10-25-10.35.14.323779
IBMSNAP_CONTACTGRP     ASN T     2004-10-25-10.35.13.819100
IBMSNAP_CONTACTS       ASN T     2004-10-25-10.35.11.094370
IBMSNAP_GROUPS         ASN T     2004-10-25-10.35.13.326681
IBMSNAP_MONENQ         ASN T     2004-10-25-10.35.15.561326
IBMSNAP_MONPARMS       ASN T     2004-10-25-10.35.12.793137
IBMSNAP_MONSERVERS     ASN T     2004-10-25-10.35.15.059507
IBMSNAP_MONTRACE       ASN T     2004-10-25-10.35.15.793706
IBMSNAP_MONTRAIL       ASN T     2004-10-25-10.35.16.300199

  10 record(s) selected.
```

Create the Replication Alert Monitor Contact

Example 6-58 shows the ASNCLP commands for creating contact information.

Example 6-58 Create replication alert monitor contact

```
#############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 6-4 on page 553
############################################################
##QREPLADM from reference M.5c in Table 6-4 on page 553
##Qrepl Admin from reference M.5d in Table 6-4 on page 553
##qrepladm@kanaga.almaden.ibm.com from reference M.5e in Table 6-4 on page 553
#############################################################

SET SERVER MONITOR TO DB MONDB ID qrepladm PASSWORD "xxxxxx" ;
SET RUN SCRIPT NOW STOP ON SQL ERROR ON ;

CREATE CONTACT "Qrepl Admin" EMAIL "qrepladm@kanaga.almaden.ibm.com" ;
```

Create Replication Alert Monitor alert conditions

Example 6-59 shows the ASNCLP commands for creating alert conditions for Q Capture and Q Apply programs on the STOCK_E and STOCK_W database servers.

Example 6-59 Create Replication Alert Monitor alert conditions

```
#############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 6-1 on page 549, Table 6-3 on page 552 and Table 6-4 on page 553
############################################################
##MONDB from reference M.5a in Table 6-4 on page 553
##STOCK_W from reference B.7a in Table 6-1 on page 549
##QREPLADM from reference M.5c in Table 6-4 on page 553
##QREPLADM from reference A.7a in Table 6-1 on page 549
##QREPLADM from reference B.7a in Table 6-1 on page 549
##Capture schema PEER2 from reference B.22 in Table 6-3 on page 552
##MON from reference M.5f in Table 6-4 on page 553
##Apply schema PEER2 from reference B.23 in Table 6-3 on page 552
##STOCK_E from reference A.7a in Table 6-1 on page 549
##Capture schema PEER1 from reference A.22 in Table 6-3 on page 552
##Apply schema PEER1 from reference A.23 in Table 6-3 on page 552
#############################################################

ASNCLP SESSION SET TO Q REPLICATION ;

SET SERVER MONITOR TO DB MONDB ID qrepladm PASSWORD "xxxxxx" ;
SET SERVER CAPTURE TO DB STOCK_W ID qrepladm PASSWORD "xxxxx" ;
SET SERVER TARGET TO DB STOCK_W ID qrepladm PASSWORD "xxxxxx" ;

SET RUN SCRIPT NOW STOP ON SQL ERROR ON;

CREATE ALERT CONDITIONS FOR QCAPTURE SCHEMA PEER2 MONITOR QUALIFIER MON NOTIFY
CONTACT "Qrepl Admin" (STATUS DOWN, ERRORS, WARNINGS, LATENCY 60, MEMORY 32,
TRANSACTION SIZE 8, SUBSCRIPTIONS INACTIVE) ;

CREATE ALERT CONDITIONS FOR QAPPLY SCHEMA PEER2 MONITOR QUALIFIER MON NOTIFY
CONTACT "Qrepl Admin" (STATUS DOWN, ERRORS, WARNINGS, LATENCY 60, EELATENCY 60,
MEMORY 32, EXCEPTIONS, SPILL QUEUES DEPTH 70,QUEUE DEPTH 70) ;

SET SERVER CAPTURE TO DB STOCK_E ID qrepladm PASSWORD "xxxxxx" ;
SET SERVER TARGET TO DB STOCK_E ID qrepladm PASSWORD "xxxxxx" ;

CREATE ALERT CONDITIONS FOR QCAPTURE SCHEMA PEER1 MONITOR QUALIFIER MON NOTIFY
CONTACT "Qrepl Admin" (STATUS DOWN, ERRORS, WARNINGS, LATENCY 60, MEMORY 32,
TRANSACTION SIZE 8, SUBSCRIPTIONS INACTIVE) ;
```

```
CREATE ALERT CONDITIONS FOR QAPPLY SCHEMA PEER1 MONITOR QUALIFIER MON NOTIFY
CONTACT "Qrepl Admin" (STATUS DOWN, ERRORS, WARNINGS, LATENCY 60, EELATENCY 60,
MEMORY 32, EXCEPTIONS, SPILL QUEUES DEPTH 70,QUEUE DEPTH 70) ;
```

Since we chose to monitor alert conditions QCAPTURE_STATUS and QAPPLY_STATUS, the system names and instance owner name must be inserted into the PARM_CHAR column of the IBMSNAP_CONDITIONS table, as shown in Example 6-60.

Example 6-60 Insert values to IBMSNAP_CONDITIONS

```
#############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 6-1 on page 549, Table 6-3 on page 552 and Table 6-4 on page 553
############################################################
##MONDB from reference M.5a in Table 6-4 on page 553
##STOCK_E from reference A.7a in Table 6-1 on page 549
##STOCK_W from reference B.7a in Table 6-1 on page 549
##db2inst1 for STOCK_E from reference A.3 in Table 6-1 on page 549
##jamesbay.almaden.ibm.com from reference A.1 in Table 6-1 on page 549
##db2inst1 for STOCK_W from reference B.3 in Table 6-1 on page 549
##azov.almaden.ibm.com from reference B.1 in Table 6-1 on page 549
#############################################################

connect to mondb ;

update asn.ibmsnap_conditions set
parm_char='STOCK_E,db2inst1,jamesbay.almaden.ibm.com' where
server_name='STOCK_E' and condition_name='QCAPTURE_STATUS' ;
update asn.ibmsnap_conditions set
parm_char='STOCK_E,db2inst1,jamesbay.almaden.ibm.com' where
server_name='STOCK_E' and condition_name='QAPPLY_STATUS' ;

update asn.ibmsnap_conditions set
parm_char='STOCK_W,db2inst1,azov.almaden.ibm.com' where server_name='STOCK_W'
and condition_name='QCAPTURE_STATUS' ;
update asn.ibmsnap_conditions set
parm_char='STOCK_W,db2inst1,azov.almaden.ibm.com' where server_name='STOCK_W'
and condition_name='QAPPLY_STATUS' ;

connect reset ;
```

Create a password file

Since we were running the Replication Alert Monitor programs from a remote system, we needed to have a password file in the MONITOR_PATH directory that contained the remote database name, user ID, and password. This file should contain this information for all the remote databases, and is used by the monitor programs to connect to the remote server. Example 6-48 on page 700 shows the creation contents of the asnpwd.aut password file, which includes the user ID/password for accessing EAST_DB, WEST_DB, STOCK_E, and STOCK_W databases.

Attention: Ensure that these databases are cataloged on the monitor server Kanaga.

Catalog the databases on the monitor server

Make sure that these databases are catalogued on the monitor server, as shown in Example 6-49 on page 700.

Start the Replication Alert Monitor program

At this point, the Replication Alert Monitor Control table IBMSNAP_MONPARMS[4] has no row. Therefore, prior to starting the Replication Alert Monitor program, the appropriate parameters must be inserted into this table, as shown in Example 6-61.

Example 6-61 Insert parameters into IBMSNAP_MONPARMS table

```
#############################################################
##The variables in the commands are shown unbold, and
##these values are substituted from the templates described
##in Table 6-4 on page 553
############################################################
##MONDB from reference M.5a in Table 6-4 on page 553
##MON from reference M.5f in Table 6-4 on page 553
##/db2_data/MON/log from reference M.6 in Table 6-4 on page 553
#############################################################
connect to mondb

insert into asn.ibmsnap_monparms
(MONITOR_QUAL, ALERT_PRUNE_LIMIT, AUTOPRUNE, EMAIL_SERVER, LOGREUSE, LOGSTDOUT,
NOTIF_PER_ALERT, NOTIF_MINUTES, MONITOR_ERRORS, MONITOR_INTERVAL, MONITOR_PATH,
RUNONCE, TERM, TRACE_LIMIT)
values
('MON', 10080, 'Y','','N','N',1,1,'',300,'/db2_data/MON/log','N','N',10080)
```

The Replication Alert Monitor can be started by executing the ASNXLP command shown in Example 6-62.

[4] Contains parameters that control the operations of the Replication Alert Monitor.

Example 6-62 Start replication alert monitor program

```
asnmon MONITOR_SERVER=MONDB MONITOR_QUAL=MON MONITOR_PATH=’/db2_data/MON/log’
PWDFILE=’asnpwd.aut’
```

Attention: Unlike Q Capture and Q Apply, there are no parameters in the IBMSNAP_MONPARMS table to specify MONITOR_PATH and PWDFILE. Therefore, ensure that these options are specified on every invocation of the Replication Alert Monitor program.

View the contents of the Replication Alert Monitor log (/db2_data/MON/log directory) to determine that the Replication Alert Monitor program initialized successfully, as shown in Example 6-63.

Example 6-63 Replication Alert Monitor log file

```
2004-11-18-16.58.02.944151 <_asnMonitor> ASN5101I  MONITOR "MON" : "Initial".
The Replication AlertMonitor program started successfully.
2004-11-18-16.58.02.991213 <WorkerMain> ASN5102I  MONITOR "MON" :
"WorkerThread". The Replication Alert Monitor program initialized successfully
and is monitoring "16" alert conditions.
```

The status of the Replication Alert Monitor threads may be viewed via the ASNCLP command shown in Example 6-64.

Example 6-64 Replication Alert Monitor threads status

```
$ asnmcmd monitor_server=MONDB monitor_qual=MON status
2004-11-22-09.51.24.482889 ASN0520I  "AsnMcmd" : "MON" : "Initial" : The STATUS
command response: "HoldLThread" thread is in the "is waiting" state.
2004-11-22-09.51.24.484636 ASN0520I  "AsnMcmd" : "MON" : "Initial" : The STATUS
command response: "AdminThread" thread is in the "is resting" state.
2004-11-22-09.51.24.484707 ASN0520I  "AsnMcmd" : "MON" : "Initial" : The STATUS
command response: "WorkerThread" thread is in the "is resting" state.
```

Note: The status of Replication Alert Monitor threads can only be checked through the command line, and not via the GUI.

6.6 Failover considerations

Luna’s Q replication peer-to-peer topology involves two AIX servers (Jamesbay and Azov), where either server can have updates occurring, and there is no concept of a primary or secondary server since the two servers are considered to be peers.

The requirement is for the surviving server to take over the workload of the failed server. When the failed server is restored, the requirement is for the peer-to-peer relationship to be reestablished between the two servers.

Attention: This section provides a very high-level overview of some of the considerations involved with failover processing associated with peer-to-peer replication. Depending upon the particular environment, the process involved in ensuring satisfactory failover processing can get quite complex. A full discussion of these considerations is beyond the scope of this book.

When the failover occurs to the surviving server, it is possible for some of the changes that occurred at the failed server not to be replicated over to the surviving 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 6-137), or messages in the WebSphere MQ queue that did not get transmitted to the surviving server (item 2 in Figure 6-137). These un-replicated changes should be considered to be data loss at the surviving server (at least until the failed server is restored). If there are messages in the receive queue on the secondary server that have not been drained (item 4 in Figure 6-137) 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.

Important: Identifying the data loss suffered can be critical for certain types of applications. Approaches for identifying the data loss suffered are beyond the scope of this book.

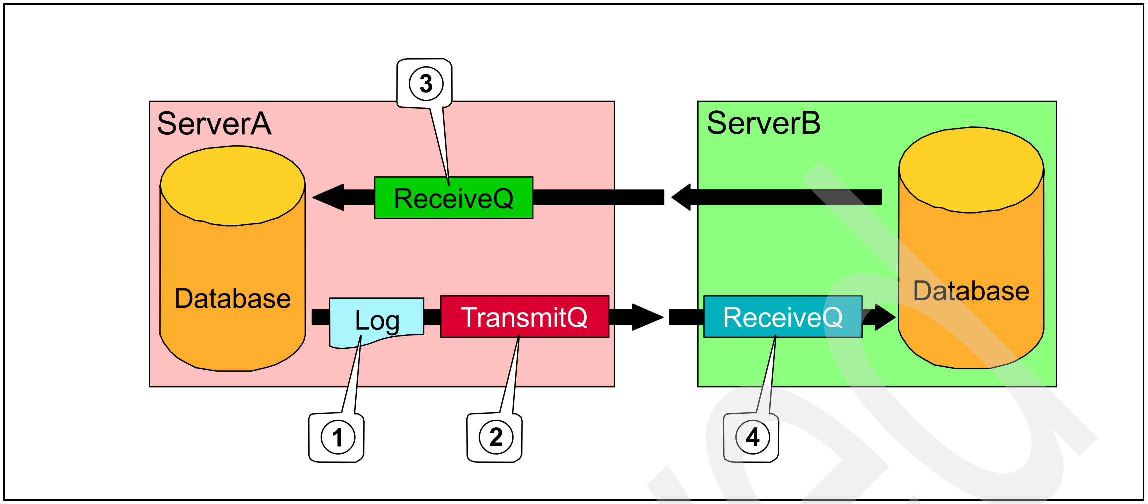


Figure 6-137 Potential sources of conflicts

The peer-to-peer scenario, by definition, has the potential for conflicts occurring at either server during normal processing, as well as failover and switchback[5]. The timestamp-based conflict detection and resolution algorithm ensures that the most recent changes to the data are the winners in the event of conflicts.

Note: The triggering event for failover is assumed to originate external to the Q replication environment.

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

1. The Q Apply program will soon catch up with any messages sent by the failed server, and will have no additional processing to perform until switchback.
2. The transmit queue will store all messages sent to the failed server up to the value specified by MAXDEPTH. The current depth of the queue may be determined by executing the command shown in Example 6-65.

Example 6-65 Determine current depth of the queue

```
$ runmqsc QM_PEER2
dis ql(QM_PEER2) curdepth

1 : dis ql(QM_PEER2) curdepth
AMQ8409: Display Queue details.
```

[5] Switchback is the process of restoring the Q replication environment to its normal operating environment after failover processing has occurred.

```
QUEUE(QM_PEER2)                    CURDEPTH(0)
```

3. Upon the transmit queue reaching its capacity (MAXDEPTH), the Q Capture program will deactivate all subscriptions or Q Capture will shut down, depending on ERROR_ACTION 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 of the transmit queue should be increased to a size capable of accumulating messages for the duration that the failed server is unavailable. The value of MAXDEPTH depends on the:

- 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 failed server outage is expected to be large, and may cause the transmit queue to fill up, then Q Capture must be shut down before the transmit queue fills up. Example 6-66 shows how the states of the subscriptions can be determined so that Q Capture is turned off before subscriptions are deactivated.

Example 6-66 Check the subscriptions states

```
$ CONNECT TO STOCK_W

$ SELECT SUBNAME, STATE, HAS_LOADPHASE FROM PEER2.IBMQREP_SUBS

SUBNAME                 STATE HAS_LOADPHASE
-------------------------------------------
ACCOUNT0001             A     E
CUSTOMER0001            A     E
STOCKS0001              A     E

  3 record(s) selected.

$ CONNECT RESET
```

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 failed 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 the updates on the DB2 log (after a Q Capture shutdown) exceeds an acceptable switchback time or the failed 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.

6.7 Switchback 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 Luna's peer-to-peer topology this involves restoring the failed server as a peer of the surviving server. Switchback should involve minimum data loss.

Important: Identifying the data loss suffered can be critical for certain types of applications. Approaches for identifying the data loss suffered are beyond the scope of this book.

Attention: This section provides a very high-level overview of some of the considerations involved with switchback processing associated with peer-to-peer replication. Depending on the particular customer environment, the process involved in ensuring satisfactory switchback processing can get quite complex. A full discussion of these considerations is beyond the scope of this book.

The effort involved in restoring the normal operating environment depends on the expected duration of the outage of the failed server.

- Short outage

  If the duration of the failed server outage is short and Q Capture was not shut down on the surviving server, then the peer-to-peer replication configuration implemented will resynchronize and resume normal operations simply by restarting the failed server, and warm starting Q Capture on the failed server.

- Extended outage

  If the duration of the failed server outage was estimated to be so long that Q Capture was shut down to prevent the queues from filling up, then after restarting the failed server, adequate time must be provided so that the messages already stored in the WebSphere MQ queues can be processed by Q Apply on the failed server. This may be determined by starting Q Apply with the command shown in Example 6-67, which causes the Q Apply program to terminate after all receive queues are emptied once. After the messages have been processed, Q Capture can be restarted with the command shown in Example 6-68, which causes Q Capture to stop after reaching the end of the DB2 active log.

Example 6-67 Determine that Q Apply has finished processing messages

```
asnqapp apply_server=.... autostop=y .....
```

Example 6-68 Determine that Q Capture has finished reading the log

```
asnqcap capture_server=.... autostop=y .....
```

If, despite adequate planning, Q Capture on the surviving server deactivated the subscriptions, then a careful analysis is required to determine the appropriate recovery action, such as:

– If the Q Subscription was configured for automatic load, then reactivation of the subscription will cause the target table to be emptied and reloaded. For a large table this could take significant time and resources. The CAPSTART signal must be issued to the Q Capture program on the surviving side to designate source and target tables.

– If the Q Subscription was configured for a manual load, then a CAPSTART signal must be issued. Reconfirm that the tables are synchronized, and then a LOADDONE signal is issued to complete subscription activation.

– If appropriate, the existing subscriptions can be dropped and recreated.

Note: It is assumed that the failed server with WebSphere II is restarted successfully to the point-of-failure.

A

WebSphere MQ overview

In this appendix we provide a high-level overview of WebSphere MQ with particular emphasis on its relationship to Q replication.

The topics covered are:

- WebSphere MQ overview
- Q replication objects

WebSphere MQ overview

WebSphere MQ allows application programs to use message queuing to participate in message-driven processing. Application programs can communicate across different platforms by using the appropriate message queuing software products. For example, HP-UX and z/OS® applications can communicate through WebSphere MQ for HP-UX and WebSphere MQ for z/OS, respectively. The applications are shielded from the mechanics of the underlying communications.

WebSphere MQ products implement a common application programming interface called the message queue interface (MQI) wherever the applications run. This makes it easier to port application programs from one platform to another. WebSphere MQ also provides a high-level message handling API called the Application Messaging Interface (AMI) that makes it easier for programmers to deploy an intelligent network for business process integration within and between enterprises. The AMI moves many message-handling functions normally performed by messaging applications into the middleware layer, where a set of policies defined by the enterprise is applied on the application's behalf.

With message queuing, the exchange of messages between the sending and receiving programs is independent of time. This means that the sending and receiving application programs are decoupled—the sender can continue processing without having to wait for the receiver to acknowledge receipt of the message. The target application does not even have to be running when the message is sent. It can retrieve the message after it is has been started. When messages arrive on a queue, they can automatically start an application using triggering. If necessary, the applications can be stopped when the message (or messages) have been processed.

With message queuing, the exchange of messages between the sending and receiving programs is independent of time. This means that the sending and receiving application programs are decoupled—the sender can continue processing without having to wait for the receiver to acknowledge receipt of the message. The target application does not even have to be running when the message is sent. It can retrieve the message after it is has been started.

When messages arrive on a queue, they can automatically start an application using triggering. If necessary, the applications can be stopped when the message (or messages) have been processed.

In the following sections we briefly describe:

- Messages and queues
- WebSphere MQ objects
- Clients and servers

- Extending queue manager facilities
- Security
- Transactional support

Messages and queues

Messages and queues are the basic components of a message queuing system.

- A message is a string of bytes that is meaningful to the applications that use it.

  Messages are used to transfer information from one application program to another, or between different parts of the same application. The applications can be running on the same platform, or on different platforms.

  WebSphere MQ messages have two parts:

  – Application data

    The content and structure of the application data is defined by the application programs that use it.

  – Message descriptor

    The message descriptor identifies the message and contains additional control information, such as the type of message and the priority assigned to the message by the sending application. The format of the message descriptor is defined by WebSphere MQ.

  The default maximum message length is 4 MB, but may be increased to 100 MB. In practice, the message length might be limited by:

  – The maximum message length defined for the receiving queue

  – The maximum message length defined for the queue manager

  – The maximum message length defined by the queue

  – The maximum message length defined by either the sending or receiving application

  – The amount of storage available for the message

  It might take several messages to send all the information that an application requires.

  Application programs send and receive messages using MQI calls. For example, to put a message onto a queue, an application:

  a. Opens the required queue by issuing an MQI MQOPEN call.
  b. Issues an MQI MQPUT call to put the message onto the queue.

  Another application can retrieve the message from the same queue by issuing an MQI MQGET call.

- A queue is a data structure used to store messages.

  Each queue is owned by a queue manager. The queue manager is responsible for maintaining the queues it owns, and for storing all the messages it receives onto the appropriate queues. The messages might be put on the queue by application programs, or by a queue manager as part of its normal operation. WebSphere MQ Version 5.3 supports queues over 2 GB in size.

  Queues can be characterized by the way they are created:

  – Predefined queues are created by an administrator using the appropriate MQSC or PCF commands.

    Predefined queues are permanent; they exist independently of the applications that use them and survive WebSphere MQ restarts.

  – Dynamic queues are created when an application issues an MQOPEN request specifying the name of a model queue.

  The queue created is based on a template queue definition, which is called a model queue. A model queue may be created by using the MQSC command DEFINE QMODEL. The attributes of a model queue (such as the maximum number of messages that can be stored on it) are inherited by any dynamic queue that is created from it. Model queues have an attribute that specifies whether the dynamic queue is to be permanent or temporary. Permanent queues survive application and queue manager restarts, while temporary queues are lost on restart.

  Suitably authorized applications can retrieve messages from a queue according to the following retrieval algorithms:

  – First-in-first-out (FIFO).

  – Message priority, as defined in the message descriptor. Messages that have the same priority are retrieved on a FIFO basis.

  – A program request for a specific message—this is how Q replication retrieves messages.

  The MQGET request from the application determines the method used.

WebSphere MQ objects

WebSphere MQ includes a number of object types such as queue managers, queues, process definitions, channels, namelists, and authentication information objects. The manipulation or administration of objects includes:

- Starting and stopping queue managers
- Creating objects, particularly queues, for applications

- Working with channels to create communication paths to queue managers on other (remote) systems
- Creating clusters of queue managers to simplify the overall administration process, and to balance workload

The naming convention adopted for WebSphere MQ objects depends on the object. Each instance of a queue manager is known by its name. This name must be unique within the network of interconnected queue managers, so that one queue manager can unambiguously identify the target queue manager to which any given message is sent. For the other types of object, each object has a name associated with it and can be referred to by that name. These names must be unique within one queue manager and object type. For example, a queue and a process may have the same name, but two queues cannot have the same name.

WebSphere MQ objects can be created, altered, or displayed using:

- Control commands, which are typed in from a keyboard
- MQSC commands, which can be typed in from a keyboard or read from a file
- Programmable Command Format (PCF) messages, which can be used in an automation program
- WebSphere MQ Administration Interface (MQAI) calls in a program
- For WebSphere MQ for Windows NT® and Windows 2000 only:
  - MQAI Component Object Model (COM) calls in a program
  - Active Directory Service interface (ADSI) calls in a program
  - The WebSphere MQ Explorer snap-in and WebSphere MQ Services snap-in running under the Microsoft® Management Console (MMC)
  - The Windows NT Default Configuration application

The properties of an object are defined by its attributes. Some may be specified, while others may only be viewed. For example, the maximum message length that a queue can accommodate is defined by its MaxMsgLength attribute; this may be specified when creating the queue. The DefinitionType attribute specifies how the queue was created, and may only be displayed.

WebSphere MQ has two ways of referring to an attribute:

- Using its PCF name, for example, MaxMsgLength
- Using its MQSC command name, for example, MAXMSGL

There are four types of queue objects in WebSphere MQ, as follows:

- Local queue object

A local queue object identifies a local queue belonging to the queue manager to which the application is connected. All queues are local queues in the sense that each queue belongs to a queue manager and, for that queue manager, the queue is a local queue.

- Remote queue object

  A remote queue object identifies a queue belonging to another queue manager. This queue must be defined as a local queue to that queue manager. The information specified when defining a remote queue object allows the local queue manager to find the remote queue manager, so that any messages destined for the remote queue go to the correct queue manager. Before applications can send messages to a queue on another queue manager, a transmission queue and channels must be defined between the queue managers, unless one or more queue managers have been grouped together into a cluster[1].

- Alias queue object

  An alias queue allows applications to access a queue by referring to it indirectly in MQI calls. When an alias queue name is used in an MQI call, the name is resolved to the name of either a local or a remote queue at runtime. This allows you to change the queues that applications use without changing the application in any way; you merely change the alias queue definition to reflect the name of the new queue to which the alias resolves. An alias queue is not a queue, but an object that you can use to access another queue.

- Model queue object

  A model queue defines a set of queue attributes that are used as a template for creating a dynamic queue. Dynamic queues are created by the queue manager when an application issues an MQOPEN request specifying a queue name that is the name of a model queue. The dynamic queue that is created in this way is a local queue whose attributes are taken from the model queue definition. The dynamic queue name can be specified by the application, or the queue manager can generate the name and return it to the application. Dynamic queues defined in this way can be temporary queues, which do not survive product restarts, or permanent queues, which do.

Queues are defined to WebSphere MQ using the:

- MQSC command DEFINE
- PCF Create Queue command

[1] A cluster is a group of queue managers set up in such a way that the queue managers can communicate directly with one another over a single network without complex transmission queue, channel, and queue definitions.

The commands specify the type of queue and its attributes. For example, a local queue object has attributes that specify what happens when applications reference that queue in MQI calls. Examples of attributes are:

- Whether applications can retrieve messages from the queue (GET enabled)
- Whether applications can put messages on the queue (PUT enabled)
- Whether access to the queue is exclusive to one application or shared between applications
- The maximum number of messages that can be stored on the queue at the same time (maximum queue depth)
- The maximum length of messages that can be put on the queue

WebSphere MQ uses some local queues for specific purposes related to its operation. These queues must be defined before WebSphere MQ can use them.

- Initiation queues

  Initiation queues are queues that are used in triggering. A queue manager puts a trigger message on an initiation queue when a trigger event occurs. A trigger event is a logical combination of conditions that is detected by a queue manager. For example, a trigger event might be generated when the number of messages on a queue reaches a predefined depth. This event causes the queue manager to put a trigger message on a specified initiation queue. This trigger message is retrieved by a trigger monitor, a special application that monitors an initiation queue. The trigger monitor then starts the application program that was specified in the trigger message. If a queue manager is to use triggering, at least one initiation queue must be defined for that queue manager.

- Transmission queues

  Transmission queues are queues that temporarily store messages that are destined for a remote queue manager. At least one transmission queue must be defined for each remote queue manager to which the local queue manager is to send messages directly. These queues are also used in remote administration. Each queue manager can have a default transmission queue. When a queue manager that is not part of a cluster puts a message onto a remote queue, the default action, if there is no transmission queue with the same name as the destination queue manager, is to use the default transmission queue.

- Cluster transmission queues

  Each queue manager within a cluster has a cluster transmission queue called SYSTEM.CLUSTER.TRANSMIT.QUEUE. A definition of this queue is created by default when a queue manager is defined. A queue manager that is part of the cluster can send messages on the cluster transmission queue to any other queue manager that is in the same cluster. During name resolution, the cluster transmission queue takes precedence over the default transmission queue. When a queue manager is part of a cluster, the default action is to use the SYSTEM.CLUSTER.TRANSMIT.QUEUE, except when the destination queue manager is not part of the cluster.

- Dead-letter queues

  A dead-letter (undelivered-message) queue is a queue that stores messages that cannot be routed to their correct destinations. This occurs when, for example, the destination queue is full. The supplied dead-letter queue is called SYSTEM.DEAD.LETTER.QUEUE. For distributed queuing, define a dead-letter queue on each queue manager involved.

- Command queues

  The command queue, SYSTEM.ADMIN.COMMAND.QUEUE, is a local queue to which suitably authorized applications can send MQSC commands for processing. These commands are then retrieved by a WebSphere MQ component called the command server. The command server validates the commands, passes the valid ones on for processing by the queue manager, and returns any responses to the appropriate reply-to queue. A command queue is created automatically for each queue manager when that queue manager is created.

- Reply-to queues

  When an application sends a request message, the application that receives the message can send back a reply message to the sending application. This message is put on a queue, called a reply-to queue, which is normally a local queue to the sending application. The name of the reply-to queue is specified by the sending application as part of the message descriptor.

- Event queues

  Instrumentation events can be used to monitor queue managers independently of MQI applications. When an instrumentation event occurs, the queue manager puts an event message on an event queue. This message can then be read by a monitoring application, which might inform an administrator or initiate some remedial action if the event indicates a problem.

  Note: Trigger events are quite different from instrumentation events in that trigger events are not caused by the same conditions, and do not generate event messages.

Now we present a brief review of queue managers, process definitions, channels, clusters, namelists, authentication information objects, and system default objects.

Queue managers

A queue manager provides queuing services to applications, and manages the queues that belong to it. It ensures that:

- Object attributes are changed according to the commands received.
- Special events, such as trigger events or instrumentation events, are generated when the appropriate conditions are met.
- Messages are put on the correct queue, as requested by the application making the MQPUT call. The application is informed if this cannot be done, and an appropriate reason code is given.

Each queue belongs to a single queue manager and is said to be a local queue to that queue manager. The queue manager to which an application is connected is said to be the local queue manager for that application. For the application, the queues that belong to its local queue manager are local queues.

A remote queue is a queue that belongs to another queue manager. A remote queue manager is any queue manager other than the local queue manager. A remote queue manager can exist on a remote machine across the network, or might exist on the same machine as the local queue manager.

WebSphere MQ supports multiple queue managers on the same machine. The attributes of a queue manager object may be retrieved using the MQI call MQINQ.

Process definitions

A process definition object defines an application that starts in response to a trigger event on a WebSphere MQ queue manager. The process definition attributes include the application ID, the application type, and data specific to the application.

Channels

Channels are objects that provide a communication path from one queue manager to another. Channels are used in distributed queuing to move messages from one queue manager to another. They shield applications from the underlying communications protocols. The queue managers might exist on the same, or different, platforms. For queue managers to communicate with one another, one channel object must be defined at the queue manager that is to send messages; and another, complementary one, at the queue manager that is to receive them.

Clusters

In a traditional WebSphere MQ network using distributed queuing, every queue manager is independent. If one queue manager needs to send messages to another queue manager, it must define a transmission queue, a channel to the remote queue manager, and a remote queue definition for every queue to which it wants to send messages.

A cluster is a group of queue managers set up in such a way that the queue managers can communicate directly with one another over a single network, without the need for transmission queue, channel, and remote queue definitions.

Namelists

A namelist is a WebSphere MQ object that contains a list of other WebSphere MQ objects. Typically, namelists are used by applications such as trigger monitors, where they are used to identify a group of queues. The advantage of using a namelist is that it is maintained independently of applications; it can be updated without stopping any of the applications that use it. Also, if one application fails, the namelist is not affected, and other applications can continue using it. Namelists are also used with queue manager clusters to maintain a list of clusters referred to by more than one WebSphere MQ object.

Authentication information objects

The queue manager authentication information object forms part of WebSphere MQ support for Secure Sockets Layer (SSL) security. It provides the definitions needed to check certificate revocation lists (CRLs) using LDAP servers. CRLs allow Certification Authorities to revoke certificates that can no longer be trusted.

System default objects

The system default objects are a set of object definitions that are created automatically whenever a queue manager is created. These object definitions may be copied and modified for use in applications at installation. Default object names have the stem SYSTEM.DEFAULT (for example, the default local queue is SYSTEM.DEFAULT.LOCAL.QUEUE, and the default receiver channel is SYSTEM.DEFAULT.RECEIVER).

These default object names cannot be renamed—default objects of these names are required. When an object is defined, any attributes that are not specified explicitly are copied from the appropriate default object. For example, if a local queue is defined, those attributes that are not specified are taken from the default queue SYSTEM.DEFAULT.LOCAL.QUEUE.

Clients and servers

WebSphere MQ supports client-server configurations for its applications.

- A WebSphere MQ client is a component that allows an application running on a system to issue MQI calls to a queue manager running on another system. The output from the call is sent back to the client, which passes it back to the application.
- A WebSphere MQ server is a queue manager that provides queuing services to one or more clients. All the WebSphere MQ objects (for example, queues) exist only on the queue manager machine (the WebSphere MQ server machine), and not on the client. A WebSphere MQ server can also support local WebSphere MQ applications.

  The difference between a WebSphere MQ server and an ordinary queue manager is that a WebSphere MQ server has a dedicated communications link with each client.

When linked to a server, client WebSphere MQ applications can issue most MQI calls in the same way as local applications. The client application issues an MQCONN call to connect to a specified queue manager. Any additional MQI calls that specify the connection handle returned from the connect request are then processed by this queue manager.

Extending queue manager facilities

The facilities provided by a queue manager can be extended by:

- User exits

  User exits provide a mechanism to insert custom code into a queue manager function. The user exits supported include:

  - Channel exits change the way that channels operate.
  - Data conversion exits create source code fragments that can be put into application programs to convert data from one format to another.
  - The cluster workload exit function performed is defined by the provider of the exit. Call definition information is given in WebSphere MQ Queue Manager Clusters.
- Installable services

  Installable services are more extensive than exits in that they have formalized interfaces (an API) with multiple entry points. An implementation of an installable service is called a service component. One may use the components supplied with the WebSphere MQ product, or write one's own component to perform the required functions.

The following installable services are currently provided:

– Authorization service

  The authorization service allows one to build a custom security facility. The default service component that implements the service is the Object Authority Manager (OAM). By default, the OAM is active, and no configuration is required. One may use the authorization service interface to create other components to replace or augment the OAM.

– Name service

  The name service enables applications to share queues by identifying remote queues as though they were local queues. A default service component that implements the name service is provided with WebSphere MQ. It uses the Open Software Foundation (OSF) Distributed Computing Environment (DCE). One may write a custom name service component in certain cases, such as when DCE is not installed. By default, the name service is inactive.

– API exits

  API exits let one write code that changes the behavior of WebSphere MQ API calls, such as MQPUT and MQGET, and then insert that code immediately before or immediately after those calls. The insertion is automatic—the queue manager drives the exit code at the registered points.

Security

In WebSphere MQ, there are three methods of providing security, as follows:

- The Object Authority Manager (OAM) facility

  Authorization for using MQI calls, commands, and access to objects is provided by the Object Authority Manager (OAM), which is enabled by default. Access to WebSphere MQ entities is controlled through WebSphere MQ user groups and the OAM. A command line interface is provided to enable administrators to grant or revoke authorizations as required.

- DCE security

  Channel exits that use the DCE Generic Security Service (GSS) are provided by WebSphere MQ.

- Channel security using Secure Sockets Layer (SSL)

  The Secure Sockets Layer (SSL) protocol provides industry-standard channel security, with protection against eavesdropping, tampering, and impersonation. SSL uses public key and symmetric techniques to provide message privacy and integrity and mutual authentication.

Transactional support

An application program can group a set of updates into a unit of work. These updates are usually logically related and must all be successful for data integrity to be preserved. If one update succeeds while another fails, data integrity is lost.

When a unit of work completes successfully, all updates made within that unit of work are made permanent and irreversible. However, if the unit of work fails, all updates are instead backed out. This process is sometimes called syncpoint coordination.

- A local unit of work is one in which the only resources updated are those of the WebSphere MQ queue manager. Here syncpoint coordination is provided by the queue manager itself using a single-phase commit process.
- A global unit of work is one in which resources belonging to other resource managers, such as XA-compliant databases, are also updated. Here, a two-phase commit procedure must be used, and the unit of work can be coordinated by the queue manager itself, or externally by another XA-compliant transaction manager such as IBM TXSeries® or BEA Tuxedo.

For further details about WebSphere MQ, refer to *WebSphere MQ System Administration Guide*, SC34-6068-02.

Q replication objects

Depending on the type of replication or publishing to be performed, various WebSphere® MQ objects need to be defined.

The WebSphere® MQ objects required by the Q Capture and Q Apply programs are as follows:

- Queue manager

  One queue manager is required on each system.
- Send queue

  This is a queue that directs data messages from a Q Capture program to a Q Apply program or user application. In remote configurations, this is the local definition on the source system of the receive queue on the target system. Each send queue should be used by only one Q Capture program.
- Receive queue

  This is a queue that receives data and informational messages from a Q Capture program to a Q Apply program or user application. This is a local queue on the target system.

- Administration queue

  This 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

  This 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

  This is a model queue that you define 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 your model queue definition, and then deletes them. The spill queue must have a specific name, IBMQREP.SPILL.MODELQ.

> **Note:** Channels and transmission queues are not defined during Q replication configuration, but are needed for operation and are defined in WebSphere MQ. Dead letter queues are optional in Q replication.

Figure A-1, Figure A-2, and Figure A-3 show the WebSphere MQ objects that are required for unidirectional replication between remote servers, event publishing between remote servers, and bidirectional or peer-to-peer replication between two remote servers, respectively.

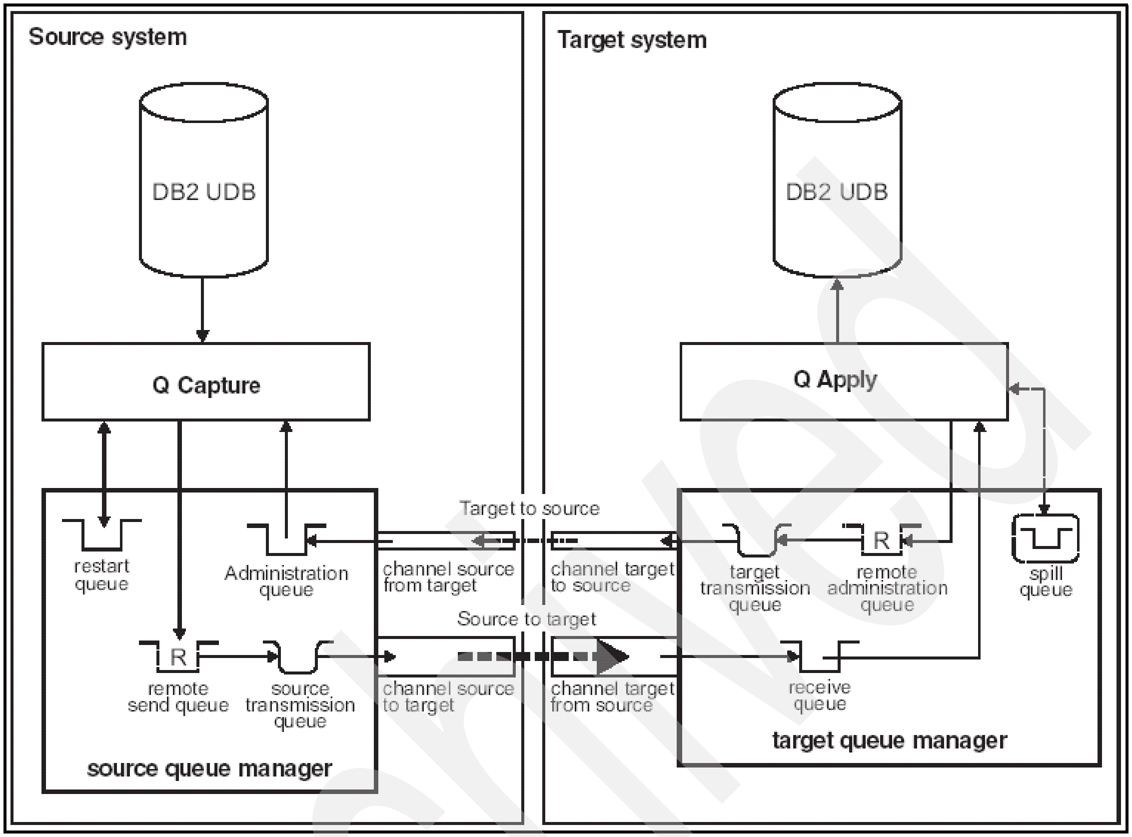


Figure A-1 Required for unidirectional Q replication between remote servers

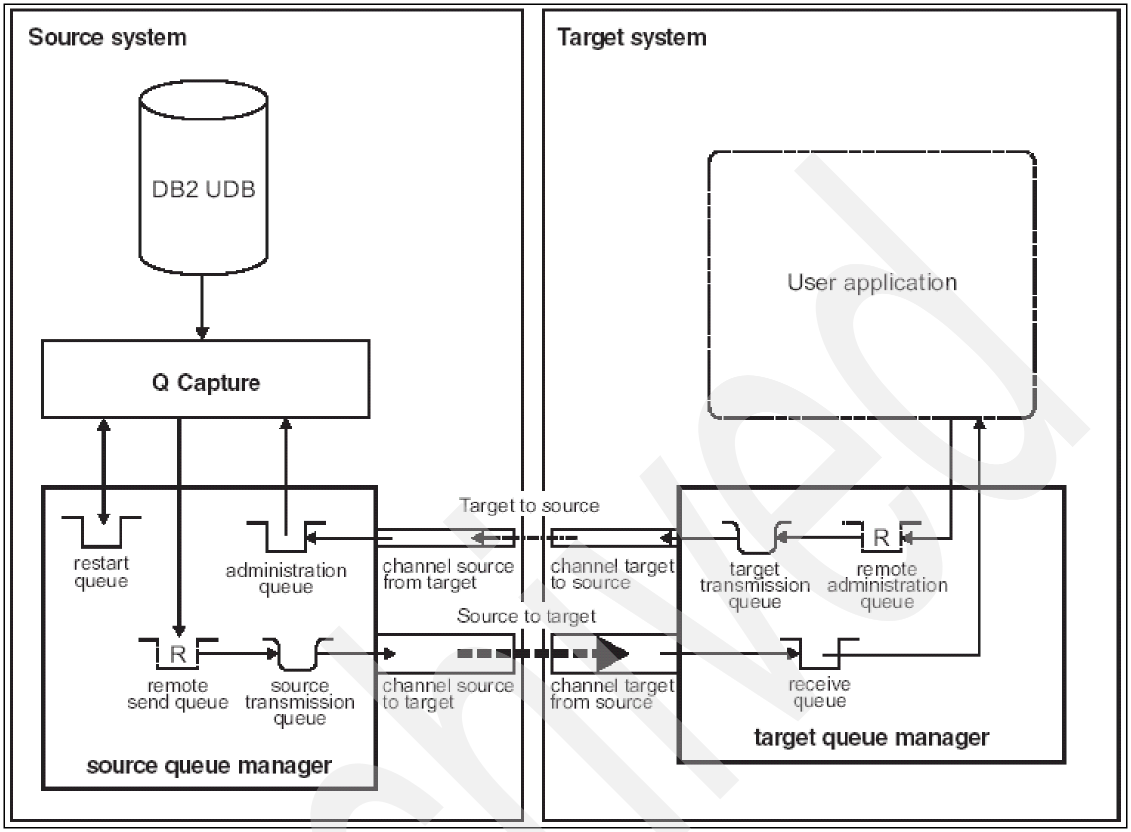


Figure A-2 Objects required for event publishing between remote servers

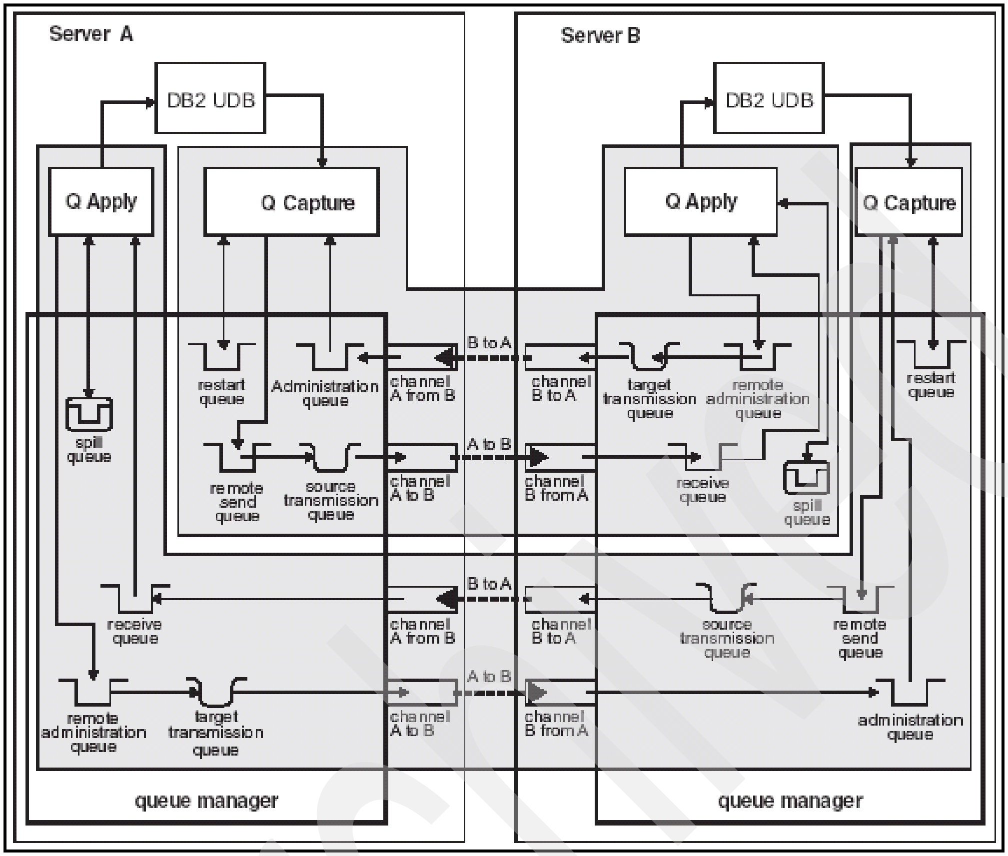


Figure A-3 Required for bidirectional or two-server peer-to-peer Q replication

The following considerations apply to the relationship between WebSphere MQ objects and their corresponding definitions in Q replication.

- The objects may be defined in WebSphere MQ and Q replication in either sequence—they both need to exist prior to operation. However, the recommendation is to define the WebSphere MQ objects first in WebSphere MQ before configuring them in Q replication.
- The maximum size of a message (MAX_MESSAGE_SIZE) that the Q Capture program can put on a send queue must be equal to or less than the WebSphere MQ Series maximum message length (MAXMSGL).

For further details, refer to *IBM DB2 Information Integrator Replication and Event Publishing Guide and Reference,* SC18-7568-00.

B

Template for topology and configuration information

In this appendix we provide the set of templates to be used for collecting relevant information for implementing bidirectional and peer-to-peer replication in z/OS and AIX environments. The elements in the template are briefly described.

The topics covered are:

- Host and DB2 system information template
- WebSphere MQ information template
- Q replication configuration information template
- Replication Alert Monitor configuration information template

Introduction

The information needed for implementing bidirectional and peer-to-peer replication involving two servers may be considered as falling into three categories, as follows:

- Host and DB2 system information
- WebSphere MQ information
- Q replication configuration information
- Replication Alert Monitor configuration information

Each of the elements and any associated parameters that may be customized for the Q replication environment are described briefly in the following sections.

Host and DB2 system information template

Table B-1 is the template for Host and DB2 system related information.

Table B-1 Host and DB2 system information

Description	Server A	Server B
Host name	(A.1)	(B.1)
Host IP address	(A.2)	(B.2)
DB2 subsystem name or instance owner	(A.3)	(B.3)
DB2 group name (z/OS only)	(A.4)	(B.4)
DB2 location name (z/OS only)	(A.5)	(B.5)
DB2 tcp port	(A.6)	(B.6)
Database server/alias information ▸ Database server/alias ▸ Q Capture user ID and password ▸ Q Apply user ID and pass Q Apply schema	 ▸ (A.7a) ▸ (A.7b) ▸ (A.7c)	 ▸ (B.7a) ▸ (B.7b) ▸ (B.7c)
User ID group (Unix only)	(A.8)	(B.8)
Other configuration user IDs requiring access	(A.9)	(B.9)
Logical database for control tables (z/OS only—must preexist) ▸ STOGROUP ▸ Volumes ▸ VCAT	(A.10) ▸ (A.10a) ▸ (A.10b) ▸ (A.10b)	(B.10) ▸ (B.10a) ▸ (B.10b) ▸ (B.10b)

The legend for each of the elements and any customizable associated parameters follows.

- Host name: The name of the current host system. It can be obtained by issuing the 'hostname' or the 'uname -n' command on AIX.
- Host IP Address: The 32-bit numeric address of the Network Interface Card (NIC) that is used to communicate with the server. A server may have multiple NICs and thus multiple IP addresses.
- DB2 subsystem name or instance owner: The user ID that owns the DB2 instance, generally db2inst1 on Unix platforms.
- DB2 group name (z/OS only).
- DB2 location name (z/OS only).
- DB2 tcp port: The TCP/IP port that is used to communicate with the DB2 database, by default 50000 on Unix. This value can be found in the file /etc/services on UNIX. On z/OS, the port number can be displayed by issuing the console command:

  ```
  =xxxx DIS DDF
  where '=' is the recognition character for the DB2 subsystem, and 'xxxx' is
  the DB2 subsystem name
  ```

- Database server/alias information: The name under which the DB2 database was cataloged.
- Database server/alias: The name under which the DB2 database was cataloged.
- Q Capture user ID and password: The user ID that will be used to operate the Q Capture program.
- Q Apply user ID and password: The user ID that will be used to operate the Q Apply program.
- User ID group (Unix only): The group name for Q replication IDs.
- Logical database for control tables (z/OS only)
  - STOGROUP: The name of the storage group
  - VOLUMES: The list of volumes associated with the storage group
  - VCAT: The name of the VCAT

WebSphere MQ information template

Table B-2 is the template for WebSphere MQ-related information.

Table B-2 WebSphere MQ information

Description	Server A	Server B
Queue Manager ► DEADQ (optional) ► MAXUMSGS (or use default) ► MAXMSGL (or use default)	(A.11) ► (A.11a) ► (A.11b) ► (A.11c)	(B.11) ► (B.11a) ► (B.11b) ► (B.11c)
Listener port	(A.12) 1450	(B.12) 1451
TransmitQ ► USAGE ► PUT ► GET ► DEFPSIST ► TRIGGER ► TRIGTYPE ► TRIGDATA ► INITQ ► MAXMSGL (or use default) ► MAXDEPTH (or use default)	(A.13) ► XMITQ ► ENABLED ► ENABLED ► YES ► YES ► FIRST ► (A.13a)=(A14) ► (A.13b) SYSTEM.CHANNEL.INITQ ► (A.13c) 4194304 ► (A.13d) 5000	(B.13) ► XMITQ ► ENABLED ► ENABLED ► YES ► YES ► FIRST ► (B.13a)=(B14) ► (B.13b) SYSTEM.CHANNEL.INITQ ► (B.13c) 4194304 ► (B.13d) 5000
SDR channel ► CHLTYPE ► TRPTYPE ► XMITQ ► CONNAME ► HBINT (or use default)	(A.14) ► SDR ► (A.14a) TCP ► (A.14b)=(A.13) ► (A.14c) ► (A.14d) 300	(B.14) ► SDR ► (B.14a) TCP ► (B.14b)=(B.13) ► (B.14c) ► (B.14d) 300
RCV channel ► CHLTYPE ► TRPTYPE ► HBINT (or use default)	(A.15)=(B.14) ► RCVR ► (A.15a) TCP ► (A.15b) 300	(B.15)=(A.14) ► RCVR ► (B.15a) TCP ► (B.15b) 300
RestartQ ► PUT ► GET ► SHARE ► DEFSOPT ► DEFPSIST ► MAXMSGL (or use default) ► MAXDEPTH (or use default)	(A.16) ► ENABLED ► ENABLED ► YES ► SHARED ► YES ► (A.16a) 4194304 ► (A.16b) 5000	(B.16) ► ENABLED ► ENABLED ► YES ► SHARED ► YES ► (B.16a) 4194304 ► (B.16b) 5000

Description	Server A	Server B
AdminQ ▶ PUT ▶ GET ▶ SHARE ▶ DEFSOPT ▶ DEFPSIST ▶ MAXMSGL (or use default) ▶ MAXDEPTH (or use default)	(A.17) ▶ ENABLED ▶ ENABLED ▶ YES ▶ SHARED ▶ YES ▶ (A.17a) 4194304 ▶ (A.17b) 5000	(B.17) ▶ ENABLED ▶ ENABLED ▶ YES ▶ SHARED ▶ YES ▶ (B.17a) 4194304 ▶ (B.17b) 5000
SendQ (remote) ▶ PUT ▶ DEFPSIST ▶ XMITQ ▶ RNAME ▶ RQMNAME	(A.18) ▶ ENABLED ▶ YES ▶ (A.18a)=(A.13) ▶ (A.18b)=(B.19) ▶ (A.18c)=(B.11)	(B.18) ▶ ENABLED ▶ YES ▶ (B.18a)=(B.13) ▶ (B.18b)=(A.19) ▶ (B.18c)=(A.11)
ReceiveQ (local) ▶ PUT ▶ GET ▶ SHARE ▶ DEFSOPT ▶ DEFPSIST ▶ INDXTYPE ▶ MSGDLVSQ ▶ MAXMSGL (or use default) ▶ MAXDEPTH (or use default)	(A.19) = (B.18b) ▶ ENABLED ▶ ENABLED ▶ YES ▶ SHARED ▶ YES ▶ MSGID ▶ PRIORITY ▶ (A.19a) 4194304 ▶ (A.19b) 5000	(B.19) = (A.18b) ▶ ENABLED ▶ ENABLED ▶ YES ▶ SHARED ▶ YES ▶ MSGID ▶ PRIORITY ▶ (B.19a) 4194304 ▶ (B.19b) 5000
SpillQ ▶ DEFTYPE ▶ DEFSOPT ▶ MSGDLVSQ ▶ MAXMSGL (or use default) ▶ MAXDEPTH (or use default)	IBMQREP.SPILL.MODELQ ▶ PERMDYN ▶ SHARED ▶ FIFO ▶ (A.20a) 4194304 ▶ (A.20b) 5000	IBMQREP.SPILL.MODELQ ▶ PERMDYN ▶ SHARED ▶ FIFO ▶ (B.20a) 4194304 ▶ (B.20b) 5000
AdminQ (remote) ▶ PUT ▶ DEFPSIST ▶ XMITQ ▶ RNAME ▶ RQMNAME	(A.21) = (B.17) ▶ ENABLED ▶ YES ▶ (A.21a)=(A.13) ▶ (A.21b)=(B.17) ▶ (A.21c)=(B.11)	(B.21) = (A.17) ▶ ENABLED ▶ YES ▶ (B.21a)=(B.13) ▶ (B.21b)=(A.17) ▶ (B.21c)=(A.11)

The legend for each of the elements and any customizable associated parameters follows.

- Queue Manager: The name of the local Queue Manager on each system. The Queue Manager owns all MQ objects.
  - DEADQ:

– MAXUMSGS:
– MAXMSGL:

- Listener port: The TCP/IP port that will be used for inter-Queue Manager communications. This value will be specified when starting the listener and when creating the sender channel.
- TransmitQ: A local queue that is used to transmit messages from the local Queue Manager to a remote Queue Manager. The transmit queue is generally named after the remote Queue Manager to which it will send messages.

  – TRIGDATA:

  – INITQ:

  – MAXMSGL:

  – MAXDEPTH: Default is 5000 for Linux, UNIX, and Windows; and 999,999,999 for z/OS

- SDR channel: The definition of one end of the communications channel that will be used to communicate with a remote queue manager. Each channel is uni-directional and is defined by its two ends (one on each Queue Manager). Both ends of the channel are named the same, and usually attempt to describe the direction of flow, for example, QM1_TO_QM2. The sender channel is defined on the Queue Manager that is the source of the communications.

  – TRPTYPE:
  – XMITQ:
  – CONNAME:
  – HBINT:

- RCV channel: The definition of one end of the communications channel that will be used to communicate with a remote queue manager. Each channel is uni-directional and is defined by its two ends (one on each Queue Manager). Both ends of the channel are named the same, and usually attempt to describe the direction of flow, for example, QM1_TO_QM2. The receiver channel is defined on the Queue Manager that is the target of the communications.

  – CHLTYPE:
  – TRPTYPE:
  – HBINT:

- RestartQ: A local queue that is defined on the Queue Manager where the Q Capture program will operate. This queue is used only by the Q Capture program, and stores the information necessary for the Q Capture to restart after a Q Capture shutdown.

  – MAXMSGL:

  - MAXDEPTH: Default is 5000 for Linux, UNIX, and Windows; and 999,999,999 for z/OS

- AdminQ: A local queue that is defined on the Queue Manager where the Q Capture program will operate. This queue is used by the Q Capture program to get messages sent to it by the Q Apply program operating on the remote server. Administrative messages are sent via this queue.
  - MAXMSGL:
  - MAXDEPTH: Default is 5000 for Linux, UNIX, and Windows; and 999,999,999 for z/OS

- SendQ (remote): The remote queue that is used by the Q Capture program to send data and administrative messages to the Q Apply program. This queue is a remote definition of the receive queue.
  - XMITQ
  - RNAME:
  - RQMNAME:

- ReceiveQ (local): A local queue that is read by the Q Apply program to get data and administrative messages sent by the Q Capture program on the remote server. This queue is a local definition of the send queue.
  - MAXMSGL:
  - MAXDEPTH: Default is 5000 for Linux, UNIX, and Windows; and 999,999,999 for z/OS.

- SpillQ: Special queues that are dynamically created by the Q Apply program for each table that is being loaded. During the load process any transactions that are received for the loading table are temporarily stored in the spill queue until after the load completes. They are then applied to the loaded table. The spill queue is created according to the model created as IBMQREP.SPILL.MODELQ.
  - MAXMSGL:
  - MAXDEPTH: Default is 5000 for Linux, UNIX, and Windows; and 999,999,999 for z/OS.

- AdminQ (remote): A remote queue created on the Q Apply Queue Manager. It is the remote definition of the admin queue on the Q Capture Queue Manager. This queue allows the Q Apply to send administrative messages to the Q Capture program.
  - XMITQ
  - RNAME:
  - RQMNAME:

Q replication configuration information template

Table B-3 is the template for Q replication configuration related information.

Table B-3 Q replication configuration information

Description	Server A	Server B
QCAPTURE Schema	(A.22)	(B.22)
QAPPLY Schema	(A.23)	(B.23)
Q Capture path (log files)	(A.24)	(B.24)
Q Apply path (log files)	(A.25)	(B.25)
Q Replication type (U/B/P)	(A.26)	(B.26)
Replication Queue Map name ► Max message length (KB) ► Error handling ► Num apply agents ► Memory buffer for Recvq (MB) ► Allow QCapture to send heartbeat ► Heartbeat interval (sec)	(A.27) ► (A.27a) 64 ► (A.27b) ► (A.27c) 16 ► (A.27d) 2 ► (A.27e) yes ► (A.27f) 60	(B.27) ► (B.27a) 64 ► (B.27b) ► (B.27c) 16 ► (B.27d) 2 ► (B.27e) yes ► (B.27f) 60
Q Subscriptions parameters ► Target table schema/creator ► Target table index schema/name ► Target tablespaces ► Check for conflicts setting ► Conflict resolution action ► Error response action ► Initial load option ► Source server for initial load ► Qsub start automatic	► (A.28a) ► (A.28b) ► (A.28c) ► (A.28d) ► (A.28e) ► (A.28f) ► (A.28g) ► (A.28h) ► (A.28i)	► (B.28a) ► (B.28b) ► (B.28c) ► (B.28d) ► (B.28e) ► (B.28f) ► (B.28g) ► (B.28h) ► (B.28i)
Q Subscriptions list	(A.29)	(B.29)

The legend for each of the elements and any customizable associated parameters follows.

- QCAPTURE Schema: A DB2 schema name that will be used when creating the control tables for the Q Capture program. If Q Capture and Q Apply will run on the same server they must use the same schema name. The default value is ASN.
- QAPPLY Schema: A DB2 schema name that will be used when creating the control tables for the Q Apply program. If Q Capture and Q Apply will run on the same server they must use the same schema name. The default value is ASN.

- Q Capture path (log files): The file system directory where the Q Capture program will write its message log file.
- Q Apply path (log files): The file system directory where the Q Apply program will write its message log file and where its password file is located. The password file is necessary for accessing remote database servers.
- Q replication type: The type of Q Replication to be used. Available options are Uni-Directional, Bi-Directional, Peer-to-Peer two servers, and Peer-to-Peer three or more servers.
- Replication Queue Map name: The name of a mapping between two servers. A Q Map consists of a send queue (Q Capture side), receive queue (Q Apply side), and admin queue (Q Apply side).
  - Max message length (KB):
  - Error handling:
  - Num apply agents:
  - Memory buffer for Recvq (MB):
  - Allow Qcap to send heartbeat:
  - Heartbeat interval (sec):
- Q Subscriptions parameters:
  - Target table schema/creator:
  - Target table index schema/name:
  - Target tablespace:
  - Check for conflicts setting:
  - Conflict resolution action:
  - Error response action:
  - Initial loadoption:
  - Source server for initial load:
  - Qsub start automatic:
- Q Subscriptions lists:

Replication Alert Monitor configuration information template

Table B-4 is the template for Replication Alert Monitor related information.

Table B-4 Replication Alert Monitor configuration information

Description	Designated server
Host name	(M.1)
Host IP address	(M.2)

Description	Designated server
DB2 subsystem name or instance owner	(M.3)
DB2 tcp port	(M.4)
Monitor database server/alias information ► Database server/alias ► db2instance ► User ID/password ► Contact name ► Contact e-mail address ► Monitor qualifier ► Schema name	 ► (M.5a) ► (M.5b) ► (M.5c) ► (M.5d) ► (M.5e) ► (M.5f) ► ASN
Monitor log path (not for z/OS)	(M.6)
DB2 group name (z/OS only)	(M.7)
DB2 location name (z/OS only)	(M.8)

The legend for each of the elements and any customizable associated parameters follows.

- Host name: The name of the current host system. It can be obtained by issuing the 'hostname' or the 'uname -n' command on AIX.
- Host IP address: The 32-bit numeric address of the Network Interface Card (NIC) that is used to communicate with the server. A server may have multiple NICs and thus multiple IP addresses.
- DB2 subsystem name or instance owner: The user ID that owns the DB2 instance, generally db2inst1 on Unix platforms.
- DB2 tcp port: The TCP/IP port that is used to communicate with the DB2 database, by default 50000 on Unix. This value can be found in the file /etc/services.
- Monitor database server/alias information:
  - Database server/alias: The name under which the DB2 database was catalogued.
  - db2instance: The name of the DB2 instance; on Unix this is the same as the owner of the instance (db2inst1).
  - User ID/password: User ID used to operate the Replication Alert Monitor program.
  - Contact name: The name of the contact person to whom the alert is sent.
  - Contact e-mail address: The e-mail address or pager address to use for alert notification.

  - Monitor qualifier: The qualifier used to group monitors.
  - Schema name: ASN. This value is not currently editable.
- Monitor log path: The file system directory where the Replication Alert Monitor program will write its message log file.
- DB2 group name (z/OS only):
- DB2 location name (z/OS only):

C

Dead letter queues in a Q replication environment

In this appendix we provide a brief description of WebSphere MQ's dead letter queues and describe their relationship to Q replication.

The topics covered are:

- Dead letter queues overview
- Q replication and dead letter queues

Dead letter queues overview

A dead-letter queue (DLQ), sometimes referred to as an undelivered-message queue, is a holding queue for messages that cannot be delivered to their destination queues. Every queue manager in a network should have an associated DLQ. Messages can be put on the DLQ by queue managers, message channel agents (MCAs), and applications.

All messages on the DLQ must be prefixed with a dead-letter header structure, MQDLH. Messages put on the DLQ by a queue manager or a message channel agent always have an MQDLH—applications putting messages on the DLQ must supply an MQDLH. The Reason field of the MQDLH structure contains a reason code that identifies why the message is on the DLQ.

All WebSphere MQ environments need a routine to process messages on the DLQ regularly. WebSphere MQ supplies a default routine, called the dead-letter queue handler (the DLQ handler), which can be invoked via the **`runmqdlq`** command. Instructions for processing messages on the DLQ are supplied to the DLQ handler by means of a user-written rules table (that is, the DLQ handler matches messages on the DLQ against entries in the rules table—when a DLQ message matches an entry in the rules table, the DLQ handler performs the action associated with that entry).

Attention: A user-written rules table is *not* provided for Q replication messages. Should one choose to define a DLQ for the Q replication environment, an appropriate DLQ handler rules table should be defined.

For further details, refer to *WebSphere MQ System Administration Guide*, SC340-6068-02.

Q replication and dead letter queues

The Q Apply checks for messages from Q Capture to arrive with an increasing number (dense numbering[1]) without gaps in them. When Q Apply detects a gap in the message numbers, it stops processing and issues the message shown in Example C-1 in the Q Apply log. It then polls the receive queue for the missing message at regular intervals and logs that message `ASN7522 The Q Apply program is polling receive queue "EAST_TO_WEST_Q", replication queue map "E_TO_W_MAP"` for message ID

[1] The Q Capture and Q Apply programs use a dense numbering system to identify and retrieve missing messages. Each message is assigned a positive integer with no gaps between numbers. A dense numbering sequence is a monotonic sequence with contiguous numbers, for example, 2,3,4,5,6,....

"515245504187F4A6000000000000000000000000000000001"“ (not shown here) as well in the Q Apply log.

Example: C-1 Gap in message number on the receive queue

```
2004-11-02-13.44.34.920606 <browser::readNextDenseMsg> ASN7551E  "Q Apply" :
"WEST" : "BR00000" : The Q Apply program detected a gap in message numbers on
receive queue "EAST_TO_WEST_Q", replication queue map "E_TO_W_MAP". It read
message ID "515245504187F4A600000000000000000000000000000002B", but expected to
find message ID "515245504187F4A6000000000000000000000000000000001". The Q Apply
program cannot process any messages until if finds the expected message.
```

Associating a dead letter queue can cause gaps in messages in the receive queue since undeliverable messages would be diverted to it.

The following sections describe how to configure a dead-letter queue in a Q replication environment, how a queue full condition causes an overflow to the dead letter queue, and how to configure a DLQ handler.

The following examples document tasks that describe the effect of dead letter queues:

1. List the attributes of the default configuration of a queue manager.
2. Create a dead letter queue.
3. Associate the created dead letter queue with the queue manager.
4. Decrease the maximum depth of the receive queue to demonstrate queue overflow.
5. Stop the receive queue to allow messages to accumulate in it.
6. Insert rows into the table to cause the receive queue to fill without overflowing to the dead letter queue.
7. Insert additional rows to cause the overflow to the dead letter queue to occur.
8. Restart the receive queue so that messages in it are processed.
9. Display contents of the queues to show messages still in the dead letter queue.
10. Modify receive queue configuration to its original value.
11. Stop the receive queue to cause messages to accumulate again.
12. Configure the dead letter queue message handler.
13. Display the contents of both the receive queue and the dead letter queue.
14. Restart the receive queue.
15. Display the current depth of the receive queue.

These steps are discussed briefly, and should provide the reader with a better understanding of how dead letter queues operate.

Example C-2 lists the attributes of the default queue manager with no dead letter queue defined for it, as highlighted in bold.

Example: C-2 Display queue manager details

```
runmqsc

dis qmgr

AMQ8408: Display Queue Manager details.
   DESCR( )                                 DEADQ( )
   DEFXMITQ( )                              CHADEXIT( )
   CLWLEXIT( )                              CLWLDATA( )
   REPOS( )                                 REPOSNL( )
   SSLKEYR(/var/mqm/qmgrs/QM_WEST/ssl/key)
   SSLCRLNL( )                              SSLCRYP( )
   COMMANDQ(SYSTEM.ADMIN.COMMAND.QUEUE)     QMNAME(QM_WEST)
   CRDATE(2004-11-04)                       CRTIME(16.33.47)
   ALTDATE(2004-11-04)                      ALTTIME(16.33.47)
   QMID(QM_WEST_2004-11-04_16.33.47)        TRIGINT(999999999)
   MAXHANDS(256)                            MAXUMSGS(10000)
   AUTHOREV(DISABLED)                       INHIBTEV(DISABLED)
   LOCALEV(DISABLED)                        REMOTEEV(DISABLED)
   PERFMEV(DISABLED)                        STRSTPEV(ENABLED)
   CHAD(DISABLED)                           CHADEV(DISABLED)
   CLWLLEN(100)                             MAXMSGL(4194304)
   CCSID(819)                               MAXPRTY(9)
   CMDLEVEL(530)                            PLATFORM(UNIX)
   SYNCPT                                   DISTL(YES)

end
```

Example C-3 creates a dead letter queue WEST_DEADQ with its MAXDEPTH attribute set to 50000. All the attributes of the WEST_DEADQ are displayed as well using the **dis ql(WEST_DEADQ)** command. The CURDEPTH shows a value of zero, indicating that there are no messages in this queue.

Example: C-3 Create dead letter queue and display its attributes

```
runmqsc

define ql(WEST_DEADQ) MAXDEPTH(50000)
AMQ8006: WebSphere MQ queue created.

dis ql(WEST_DEADQ)
AMQ8409: Display Queue details.
```

```
   DESCR(WebSphere MQ Default Local Queue)
   PROCESS( )                              BOQNAME( )
   INITQ( )                                TRIGDATA( )
   CLUSTER( )                              CLUSNL( )
   QUEUE(WEST_DEADQ)                       CRDATE(2004-11-05)
   CRTIME(16.15.46)                        ALTDATE(2004-11-05)
   ALTTIME(16.15.46)                       GET(ENABLED)
   PUT(ENABLED)                            DEFPRTY(0)
   DEFPSIST(NO)                            MAXDEPTH(50000)
   MAXMSGL(4194304)                        BOTHRESH(0)
   SHARE                                   DEFSOPT(SHARED)
   HARDENBO                                MSGDLVSQ(PRIORITY)
   RETINTVL(999999999)                     USAGE(NORMAL)
   NOTRIGGER                               TRIGTYPE(FIRST)
   TRIGDPTH(1)                             TRIGMPRI(0)
   QDEPTHHI(80)                            QDEPTHLO(20)
   QDPMAXEV(ENABLED)                       QDPHIEV(DISABLED)
   QDPLOEV(DISABLED)                       QSVCINT(999999999)
   QSVCIEV(NONE)                           DISTL(NO)
   DEFTYPE(PREDEFINED)                     TYPE(QLOCAL)
   SCOPE(QMGR)                             DEFBIND(OPEN)
   IPPROCS(0)                              OPPROCS(0)
   CURDEPTH(0)

end
```

Example C-4 associates the queue manager with the dead letter queue WEST_DEADQ, and its modified attributes are again displayed using the **dis qmgr** command. The DEADQ parameter shows the new association.

Example: C-4 Alter queue manager to associate dead letter queue

```
runmqsc

alter qmgr DEADQ(WEST_DEADQ)
AMQ8005: WebSphere MQ queue manager changed.

dis qmgr
AMQ8408: Display Queue Manager details.
   DESCR( )                                DEADQ(WEST_DEADQ)
   DEFXMITQ( )                             CHADEXIT( )
   CLWLEXIT( )                             CLWLDATA( )
   REPOS( )                                REPOSNL( )
   SSLKEYR(/var/mqm/qmgrs/QM_WEST/ssl/key)
   SSLCRLNL( )                             SSLCRYP( )
   COMMANDQ(SYSTEM.ADMIN.COMMAND.QUEUE)    QMNAME(QM_WEST)
   CRDATE(2004-11-04)                      CRTIME(16.33.47)
   ALTDATE(2004-11-05)                     ALTTIME(16.17.42)
   QMID(QM_WEST_2004-11-04_16.33.47)       TRIGINT(999999999)
```

```
   MAXHANDS(256)                          MAXUMSGS(10000)
   AUTHOREV(DISABLED)                     INHIBTEV(DISABLED)
   LOCALEV(DISABLED)                      REMOTEEV(DISABLED)
   PERFMEV(DISABLED)                      STRSTPEV(ENABLED)
   CHAD(DISABLED)                         CHADEV(DISABLED)
   CLWLLEN(100)                           MAXMSGL(4194304)
   CCSID(819)                             MAXPRTY(9)
   CMDLEVEL(530)                          PLATFORM(UNIX)
   SYNCPT                                 DISTL(YES)

end
```

Example C-5 shows the commands for reducing the MAXDEPTH parameter value from 50000 to 10 in order to demonstrate message flow when dead letter queues are defined.

Example: C-5 Decrease MAXDEPTH of Q replication receive queue

```
runmqsc

dis ql(EAST_TO_WEST_Q)
AMQ8409: Display Queue details.
   DESCR(LOCAL RECEIVE QUEUE - WEST FROM EAST)
   PROCESS( )                             BOQNAME( )
   INITQ( )                               TRIGDATA( )
   CLUSTER( )                             CLUSNL( )
   QUEUE(EAST_TO_WEST_Q)                  CRDATE(2004-11-04)
   CRTIME(16.33.55)                       ALTDATE(2004-11-04)
   ALTTIME(16.33.55)                      GET(ENABLED)
   PUT(ENABLED)                           DEFPRTY(0)
   DEFPSIST(YES)                          MAXDEPTH(50000)
   MAXMSGL(4194304)                       BOTHRESH(0)
   SHARE                                  DEFSOPT(SHARED)
   HARDENBO                               MSGDLVSQ(PRIORITY)
   RETINTVL(999999999)                    USAGE(NORMAL)
   NOTRIGGER                              TRIGTYPE(FIRST)
   TRIGDPTH(1)                            TRIGMPRI(0)
   QDEPTHHI(80)                           QDEPTHLO(20)
   QDPMAXEV(ENABLED)                      QDPHIEV(DISABLED)
   QDPLOEV(DISABLED)                      QSVCINT(999999999)
   QSVCIEV(NONE)                          DISTL(NO)
   DEFTYPE(PREDEFINED)                    TYPE(QLOCAL)
   SCOPE(QMGR)                            DEFBIND(OPEN)
   IPPROCS(2)                             OPPROCS(1)
   CURDEPTH(0)

alter ql(EAST_TO_WEST_Q) MAXDEPTH(10)
AMQ8008: WebSphere MQ queue changed.
```

```
dis ql(EAST_TO_WEST_Q) MAXDEPTH
AMQ8409: Display Queue details.
   QUEUE(EAST_TO_WEST_Q)                   MAXDEPTH(10)

end
```

Example C-6 shows the commands for stopping the receive message queue so that messages will accumulate. The state of the receive queue is then reviewed by querying the IBMQREP_RECVQUEUES table—showing it to be inactive (I).

Example: C-6   Stop the receive queue to let message accumulate and confirm its state

```
$ asnqacmd apply_server=WEST_DB apply_schema=WEST stopq=EAST_TO_WEST_Q
2004-11-05-16.26.31.085735 ASN0522I  "AsnQAcmd" : "WEST" : "Initial" : The
program received the "STOPQ" command.

--Confirm the state of the receive queue by querying the IBMQREP_RECVQUEUES
--table.

$ db2 connect to WEST_DB

   Database Connection Information

 Database server        = DB2/6000 8.2.0
 SQL authorization ID   = DB2INST1
 Local database alias   = WEST_DB

$ db2 select recvq, state from west.ibmqrep_recvqueues

RECVQ                                            STATE
------------------------------------------------ -----
EAST_TO_WEST_Q                                   I

  1 record(s) selected.
```

We inserted 10 rows into a table on the Q Capture side of the replication pair—a number that matches the maximum depth (MAXDEPTH value of 10) set for the receive queue. Example C-7 shows the current depth of the receive queue (CURDEPTH value of 10) and dead letter queue (CURDEPTH value of 0) to confirm that the receive queue had filled up without overflowing to the dead letter queue.

Example: C-7   Display queues after inserting rows in table to correspond to MAXDEPTH

```
runmqsc

dis ql(EAST_TO_WEST_Q) CURDEPTH
AMQ8409: Display Queue details.
```

```
   QUEUE(EAST_TO_WEST_Q)                        CURDEPTH(10)

dis ql(WEST_DEADQ) CURDEPTH
AMQ8409: Display Queue details.
   QUEUE(WEST_DEADQ)                            CURDEPTH(0)

end
```

We inserted additional rows into a table on the Q Capture side of the replication pair up to cause an overflow of the receive queue. Example C-8 shows the current depth of the receive queue (CURDEPTH value of 10) and dead letter queue (CURDEPTH value of 2) that confirms that overflow had occurred to the dead letter queue.

Example: C-8 Display queues after adding rows in table to cause spillover to DLQ

```
runmqsc

dis ql(EAST_TO_WEST_Q) CURDEPTH
AMQ8409: Display Queue details.
   QUEUE(EAST_TO_WEST_Q)                        CURDEPTH(10)

dis ql(WEST_DEADQ) CURDEPTH
AMQ8409: Display Queue details.
   QUEUE(WEST_DEADQ)                            CURDEPTH(2)

end
```

Example C-9 shows the command for restarting the receive queue so that processing of the messages may proceed. The status of the receive queue is displayed by viewing the contents of the IBMQREP_RECVQUEUES—its state is active (A).

Example: C-9 Restart receive queue to process rows and confirm its state

```
$ asnqacmd apply_server=WEST_DB apply_schema=WEST startq=EAST_TO_WEST_Q
2004-11-05-16.39.51.396542 ASN0522I  "AsnQAcmd" : "WEST" : "Initial" : The
program received the "STARTQ" command.


--Confirm the state of the receive queue.

$ db2 select recvq, state from west.ibmqrep_recvqueues

RECVQ                                                STATE
------------------------------------------------ -----
EAST_TO_WEST_Q                                       A
```

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

Example C-10 shows the commands for displaying the contents of both the receive queue and the dead letter queue after the receive queue had been restarted. It shows the current depth of the receive queue (CURDEPTH value of 0) and the dead letter queue (CURDEPTH value of 2), indicating that messages were cleared from the receive queue by the Q Apply program, but the messages in the dead letter queue remain.

Example: C-10 Display content of both queues

```
runmqsc

dis ql(EAST_TO_WEST_Q) CURDEPTH
AMQ8409: Display Queue details.
   QUEUE(EAST_TO_WEST_Q)                    CURDEPTH(0)

dis ql(WEST_DEADQ) CURDEPTH
AMQ8409: Display Queue details.
   QUEUE(WEST_DEADQ)                        CURDEPTH(2)

end
```

Example C-11 shows the command for restoring the MAXDEPTH parameter to its original value of 50000, along with a display of MAXDEPTH and CURDEPTH attributes.

Example: C-11 Restore receive queue to its original configuration

```
runmqsc

alter ql(EAST_TO_WEST_Q) MAXDEPTH(50000)
AMQ8008: WebSphere MQ queue changed.

dis ql(EAST_TO_WEST_Q) MAXDEPTH CURDEPTH
AMQ8409: Display Queue details.
   QUEUE(EAST_TO_WEST_Q)                    MAXDEPTH(50000)
   CURDEPTH(0)

end
```

Example C-12 shows the command for stopping the receive queue so that messages may accumulate again, and confirms the state of the receive queue to be inactive (I) by querying the IBMQREP_RECVQUEUES.

Example: C-12 Stop receive queues to let messages build up again and confirm its state

```
$ asnqacmd apply_server=WEST_DB apply_schema=WEST stopq=EAST_TO_WEST_Q
```

```
2004-11-05-16.53.02.410589 ASN0522I  "AsnQAcmd" : "WEST" : "Initial" : The
program received the "STOPQ" command.


--Confirm the state of the receive queue.

$ db2 select recvq, state from west.ibmqrep_recvqueues

RECVQ                                                STATE
------------------------------------------------ -----
EAST_TO_WEST_Q                                       I

  1 record(s) selected.
```

Example C-13 shows the command for configuring the dead letter queue message handler so that messages received because of a queue full condition will be replaced on their original target queue. The first attempt to put the message on its destination queue fails. The RETRY keyword sets the number of tries made to implement an action. The RETRYINT keyword of the control data controls the interval between attempts.

Example C-13 shows the DLQ handler having begun processing the dead letter queue (which had two messages in it).

Example: C-13 Configure dead letter queue message handler

```
File: dlqrule.txt
inputqm(' ') inputq(' ')

REASON(MQRC_Q_FULL) ACTION(RETRY) RETRY(5) RETRYINT(5)


$ runmqdlq WEST_DEADQ QM_WEST < dlqrule.txt
2004-11-0516.59.31 AMQ8708: Dead-letter queue handler started to process
INPUTQ(WEST_DEADQ).

end
```

Example C-14 shows the contents of the receive and dead letter queue a little bit later. The number of messages in the receive queue (CURDEPTH) is 3, while that of the dead letter queue is 0—this is the effect of the DLQ message handler. The third message is due to the heartbeat message sent from the Q Capture program. Depending on how quickly these steps are performed, one may see more than one heartbeat appear.

Example: C-14 Display the content of both the receive and dead letter queues

```
runmqsc
```

```
dis QL(EAST_TO_WEST_Q) CURDEPTH
AMQ8409: Display Queue details.
   QUEUE(EAST_TO_WEST_Q)                       CURDEPTH(3)
dis ql(WEST_DEADQ) CURDEPTH
AMQ8409: Display Queue details.
   QUEUE(WEST_DEADQ)                           CURDEPTH(0)

end
```

Example C-15 shows the command for restarting the receive queue so that processing of the messages may proceed.

Example: C-15 Restart receive queue and display its state

```
$ asnqacmd apply_server=WEST_DB apply_schema=WEST startq=EAST_TO_WEST_Q
2004-11-05-17.04.04.989322 ASN0522I  "AsnQAcmd" : "WEST" : "Initial" : The
program received the "STARTQ" command.

--Display the current depth of the receive queue and see that again the Q Apply
--program has --cleared all the messages from the queue.
```

Example C-16 shows the current depth of the receive queue (CURDEPTH value of 0), indicating that the Q Apply program has cleared all the messages in the receive queue.

Example: C-16 Display status of the receive queue

```
runmqsc

dis ql(EAST_TO_WEST_Q) CURDEPTH
AMQ8409: Display Queue details.
   QUEUE(EAST_TO_WEST_Q)                       CURDEPTH(0)

end
```

D

Common troubleshooting setup problems

In this appendix we describe the main troubleshooting tools available, propose a basic troubleshooting methodology, identify some of the commonly encountered problems during setup, and provide guidelines for resolving them.

The topics covered are:

- Troubleshooting tools
- Basic troubleshooting methodology
- Commonly encountered problems
- Synchronizing tables
- Subscription deactivation

Troubleshooting tools available

Since the Q replication environment includes DB2 and WebSphere MQ on different operating system platforms, a number of different sources may need to be monitored during troubleshooting. Sources include logs, commands, and GUI monitors and tasks. The following list of sources, while not comprehensive, should provide information to assist with troubleshooting a Q replication problem.

AIX operating system

The **ps** command shows the current status of the processes.

WebSphere MQ

For WebSphere MQ:

- Message log /var/mqm/qmgrs/queue_manager/errors/AMQERR##.log

  For example, /var/mqm/qmgrs/QM_EAST/errors/QMQERR01.log

- Commands

  – **runmqsc**: MQ command utility; displays status (for example, checking the status of the EAST_ADMINQ queue)

  ```
  runmqsc QM_EAST
  dis ql(EAST_ADMINQ)
  end
  ```

  – **dspmq**: Lists all Queue Managers, for example:

  ```
  dspmq
  QMNAME(QM_EAST) STATUS(Running)
  ```

  – **mqrc**: Examines MQ error codes, for example:

  ```
  mqrc 2085
     2085  0x00000825  MQRC_UNKNOWN_OBJECT_NAME
  ```

  – **dmpmqaut**: Dumps all MQ authorizations, for example:

  ```
  dmpmqaut -m QM_EAST -t queue -g qreplgrp
  profile:     EAST_ADMINQ
  object type: queue
  entity:      qreplgrp
  entity type: group
  authority:   get browse put inq
  ....
  ```

DB2 UDB

For DB2 UDB:

- **db2**: Command utility to display error codes and execute SQL
  - For example, db2> ? SQL1040
  - db2> select * from department
- DB2 dialog $HOME/sqllib/db2dump/db2diag.log
  - For example, /db2_data/db2inst1/sqllib/db2dump/db2diag.log
- DB2 Control Center with Health Center

Q replication

For Q replication:

- Q Capture log
  CAPTURE_PATH/db2instance.capture_server.capture_schema.QCAP.log

  For example, /db2_data/capture/db2inst1.EAST_DB.EAST.QCAP.log
- Q Apply log
  APPLY_PATH/db2instance.apply_server.apply_schema.QAPP.log

  For example, /db2_data/apply/db2inst1.WEST_DB.WEST.QAPP.log
- Commands
  - **asnqmfmt**: Examine messages in the local queues and remove messages from queues.
  - **asntdiff**: Determine differences between tables in a subscription.
  - **asntrep**: Resynchronize tables in a subscription.
  - **asnqccmd**: Q Capture command utility - display status.
  - **asnqacmd**: Q Apply command utility - display status.
  - **asnmcmd**: Q Monitor command utility - display status.
  - **asnpwd**: Create and modify password files.
  - **asnqanalyze**: Service tool to analyze the replication setup and produce HTML output.
- Replication Center

Basic troubleshooting methodology

The following basic steps are recommended for troubleshooting a problem:

1. Check if Q Capture and Q Apply programs are running on both servers.
2. Check the Q subscriptions states and receive queue states on both servers.
3. Look into Q Apply logs on both servers.

   Solve errors highlighted in the log, if any.
4. Investigate the Q Capture logs on both servers.

   Solve errors highlighted in the logs, if any.
5. Check the WebSphere MQ queues.
   a. Transmit queue's queue depth.
   b. Receive queue's queue depth.
   c. Ensure listeners are running on both servers.
   d. Ensure channels are running on both servers.

Commonly encountered problems

Some of the more commonly encountered problems may be broadly classified into setup errors and operational errors. These are briefly described here.

Setup errors

Table D-1 and Table D-2 list some of the common setup errors encountered and the resolution action relating to WebSphere MQ and Q replication, respectively. These return codes may be retrieved from one or more of the sources listed in “Troubleshooting tools available” on page 764.

Table D-1 WebSphere MQ common setup errors

Return code	Meaning	Resolution
2035	MQRC_NOT_AUTHORIZED	Make sure that the password file is created in the APPLY_PATH and has a correct entry for the source server. Make sure that the APPLY_PATH and PASSWORD_FILE are properly set in the APPLYPARMS table.
2058	MQRC_Q_MGR_NAME_ERROR	Check the value of the QMGR in the CAPPARMS table.

Return code	Meaning	Resolution
2085	MQRC_UNKNOWN_REMOTE_Q_MGR	Check the value of the RQMNAME for the remote queue (runmqsc).
2087	MQRC_UNKNOWN_OBJECT_NAME	Check the queue object name/spelling in the Control Tables to make sure that it was entered correctly when configuring replication.

Table D-2 Q replication setup errors

Return Code	Meaning	Resolution
1224	Number of threads per process has exceeded in AIX	export EXTSHM=ON db2set DB2ENVLIST=EXTSHM db2stop db2start

Operational errors

Some of the common operational errors encountered are as follows:

- Subscriptions stay in T state and are not working.
- System clocks are different on the two servers.
- Data is not replicating.
- Q Apply has detected a missing message.
- A CAPSTOP signal is issued on subscription state not I.

Each of these errors is discussed in the following sections with resolution suggestions.

Subscriptions stay in T[1] state and are not working

During activation of a new Q subscription, the Q Apply and Q Capture programs exchange control messages, which change the state of the subscription. A state of T should be very transitory. When the subscription remains in the T state for a long time, it indicates a problem.

The resolution to this problem is as follows:

1. Make sure you start all Q Capture programs before stating Q Apply programs.
2. Make sure that the listeners are running on both servers

```
$ps -ef | grep lsr
```

[1] An internal state that indicates that the Q Capture program read a CAPSTART signal in the log for this peer-to-peer Q subscription, and the Q subscription is being initialized within the peer-to-peer group.

```
db2inst1 55090    1   0   Nov 15      -  0:00 runmqlsr -m QM_PEER1 -t tcp
-p 1420
```

3. Make sure that the channels are working on both servers:

```
$ ps -ef | grep chl
db2inst1 29598 35280 0 15:56:39 -0:00 runmqchl -c PEER1_TO_PEER2 -m
QM_PEER1
```

   And/or:

```
$ runmqsc QM_PEER1
      dis chstatus(PEER1_TO_PEER2)
         1 : dis chstatus(PEER1_TO_PEER2)

         AMQ8417: Display Channel Status details.
         CHANNEL(PEER1_TO_PEER2)          XMITQ(QM_PEER2)
         CONNAME(9.1.39.89(1421))         CURRENT
         CHLTYPE(SDR)                     STATUS(RUNNING)
         RQMNAME(QM_PEER2)
```

4. Stop and restart Q Capture and Q Apply.

System clocks are different on the two servers

The Q Apply log indicates that there is a difference between the clocks on the Q Capture and Q Apply servers, as shown in Example 6-69.

Example 6-69 System clocks skew message in the Q Apply log

```
2004-10-26-11.09.26.494623 <QArow::p2pCheckRowFromFuture> ASN7614W  "Q Apply" :
"PEER1" : "BR00000AG004" : The local clock is at least "51" seconds behind the
clock of the system sending on recieve queue "PEER2_TO_PEER1_Q" for replication
queue map "P2_TO_P1_MAP".  This causes delays in processing P2P rows.

2004-10-26-11.10.24.551856 <appAgntMain> ASN8047D  "Q Apply" : "PEER1" :
"BR00000AG004" : Q Capture MQPUT timestamp occurs after Q Apply MQGET. Possible
clock skew between source and target servers.
```

This is relevant to peer-to-peer replication, which requires both the servers in the topology to have synchronized clocks for conflict resolution to be effective. Peer-to-peer replication relies on timestamp fields in each record to resolve conflicts. If the system clocks on the peer-to-peer servers get out of sync, it can affect Q replication.

The resolution to this problem is to resynchronize the time on both servers using the following AIX command:

```
$ setclock [ TimeServer ]
```

This command synchronizes the system clock on the server it runs on with the system clock of the server specified in the [TimeServer].

> **Attention:** Set the system clock forward only—run the command on the server whose clock is lagging.

Data is not replicating

A check of the target servers indicates that changes occurring at the source server are not being replicated.

The resolution to this problem is as follows:

1. Check the subscription states and receive queue states on both servers:

```
$ db2 CONNECT TO STOCK_E

   Database Connection Information

 Database server        = DB2/6000 8.2.0
 SQL authorization ID   = QREPLADM
 Local database alias   = STOCK_E


$ db2 "SELECT SUBNAME, STATE FROM PEER1.IBMQREP_SUBS"

SUBNAME             STATE
------------------- -----
ACCOUNT0001         A
CUSTOMER0001        A
STOCKS0001          A

  3 record(s) selected.

$ db2 "SELECT SUBNAME, SUBGROUP, STATE FROM PEER1.IBMQREP_TARGETS"

SUBNAME                 SUBGROUP   STATE
----------------------- --------   -----
ACCOUNT0002             P2P0001    A
CUSTOMER0002            P2P0001    A
STOCKS0002              P2P0001    A

  3 record(s) selected.
```

```
$ db2 "SELECT substr(sendq,1,20) as SENDQ, substr(recvq,1,20) as RECVQ,
STATE FROM PEER1.IBMQREP_SENDQUEUES"


SENDQ                RECVQ                STATE
-------------------- -------------------- -----
PEER1_TO_PEER2_Q     PEER1_TO_PEER2_Q     A

  1 record(s) selected.


$ db2 "SELECT substr(sendq,1,20) as SENDQ, substr(recvq,1,20) as RECVQ,
STATE FROM PEER1.IBMQREP_RECVQUEUES"


SENDQ                RECVQ                STATE
-------------------- -------------------- -----
PEER2_TO_PEER1_Q     PEER2_TO_PEER1_Q     A

  1 record(s) selected.


$ db2 CONNECT RESET
DB20000I  The SQL command completed successfully.
```

2. Investigate the Q apply logs on both servers:

   ```
   APPLY_PATH/db2inst1.STOCK_E.PEER1.QAPP.log
   ```

   Resolve any errors in the logs, and locate the first error that might have caused the problem.

3. Investigate the Q Capture logs on both servers:

   ```
   CAPTURE_PATH/db2inst1.STOCK_E.PEER1.QCAP.log
   ```

   Resolve any errors in the logs, and locate the first error that might have caused the problem.

4. Check the queue depth of the transmit queue and receive queue on both servers:

   ```
   $ runmqsc QM_PEER1

      dis ql(QM_PEER2) curdepth
        1 : dis ql(QM_PEER2) curdepth
      AMQ8409: Display Queue details.
      QUEUE(QM_PEER2)                           CURDEPTH(0)

      dis ql(PEER2_TO_PEER1_Q) curdepth
        2 : dis ql(PEER2_TO_PEER1_Q) curdepth
      AMQ8409: Display Queue details.
   ```

```
QUEUE(PEER2_TO_PEER1_Q)                 CURDEPTH(0)
```

If there are messages in the queue:

a. Make sure that the listeners are listening on both servers:

```
$ ps -ef | grep lsr
db2inst1 55090   1 0 Nov 15  -  0:00 runmqlsr -m QM_PEER1 -t tcp -p 1420
```

b. Make sure the channels are working on both servers:

```
      $ ps -ef | grep chl
      db2inst1 29598 35280 0 15:56:39 - 0:00 /usr/mqm/bin/runmqchl -c
PEER1_TO_PEER2 -m QM_PEER1
```

Or:

```
$ runmqsc QM_PEER1

         dis chstatus(PEER1_TO_PEER2)
           1 : dis chstatus(PEER1_TO_PEER2)
         AMQ8417: Display Channel Status details.
         CHANNEL(PEER1_TO_PEER2)                 XMITQ(QM_PEER2)
         CONNAME(9.1.39.89(1421))                CURRENT
         CHLTYPE(SDR)                            STATUS(RUNNING)
         RQMNAME(QM_PEER2)
```

Q Apply has detected a missing message

Q Apply has detected a missing message, as shown in Example D-1, and has stopped processing messages until the missing message arrives.

Example: D-1 Gap in message numbers

```
2004-11-02-13.44.34.920606 <browser::readNextDenseMsg> ASN7551E  "Q
Apply" : "WEST" : "BR00000" : The Q Apply program detected a gap in
message numbers on receive queue "EAST_TO_WEST_Q", replication queue
map "E_TO_W_MAP". It read message ID
"515245504187F4A6000000000000000000000000000002B", but expected to
find message ID "515245504187F4A60000000000000000000000000000001". The
Q Apply program cannot process any messages until if finds the expected
message.
```

The resolution to this problem is as follows:

1. Stop Q Capture on the (source) EAST server.

```
$ asnqccmd capture_server=STOCK_W capture_schema=WEST STOP
```

2. Stop the receive queue on the (target) WEST server (where the dense message numbering problem is occurring):

```
$ asnqacmd capture_server=STOCK_W capture_schema=WEST stopq=EAST_TO_WEST_Q
```

3. Manually change the state of *all* subscriptions on the queue to INACTIVE (in the IBMQREP_SUBS and IBMQREP_TARGETS tables on both servers EAST and WEST):

```
$ db2 connect to STOCK_W
   $ db2 "update IBMQREP.SUBS set state='I' where subname = 'ACCOUNT0002'";
$ db2 "update IBMQREP.SUBS set state='I' where subname = 'CUSTOMER0002'";
$ db2 "update IBMQREP.SUBS set state='I' where subname = 'STOCKS0002'";
   $ db2 "update IBMQREP.TARGETS set state='I' where subname =
'ACCOUNT0001'";
   $ db2 "update IBMQREP.TARGETS set state='I' where subname =
'CUSTOMER0001'";
   $ db2 "update IBMQREP.TARGETS set state='I' where subname =
'STOCKS0001'";
$ db2 connect reset

$ db2 connect to STOCK_E
   $ db2 "update IBMQREP.SUBS set state='I' where subname = 'ACCOUNT0001'";
$ db2 "update IBMQREP.SUBS set state='I' where subname = 'CUSTOMER0001'";
$ db2 "update IBMQREP.SUBS set state='I' where subname = 'STOCKS0001'";
   $ db2 "update IBMQREP.TARGETS set state='I' where subname =
'ACCOUNT0002'";
   $ db2 "update IBMQREP.TARGETS set state='I' where subname =
'CUSTOMER0002'";
   $ db2 "update IBMQREP.TARGETS set state='I' where subname =
'STOCKS0002'";
$ db2 connect reset
```

4. Empty the receive queue on the (target) WEST server:

```
$ asnqmfmt EAST_TO_WEST_Q QM_WEST -delmsg
```

5. Warm the start capture on (source) EAST server:

```
$ asnqcap capture_server=STOCK_E capture_schema=EAST startmode=WARMSI
```

6. Start the receive queue on (target) WEST server:

```
$ asnqacmd capture_server=STOCK_W capture_schema=WEST startq=EAST_TO_WEST_Q
```

7. Issue a CAPSTART signal on the subscriptions on the (source) EAST server:

```
$ db2 connect to STOCK_E
   $ db2 "insert into IBMQREP.SIGNAL (
SIGNAL_TIME,
SIGNAL_TYPE,
SIGNAL_SUBTYPE,
SIGNAL_INPUT_IN,
SIGNAL_STATE
```

```
) values (
CURRENT TIMESTAMP, 'CMD', 'CAPSTART',' ACCOUNT0001', 'P');
   $ db2 "insert into IBMQREP.SIGNAL (
SIGNAL_TIME,
SIGNAL_TYPE,
SIGNAL_SUBTYPE,
SIGNAL_INPUT_IN,
SIGNAL_STATE
) values (
CURRENT TIMESTAMP, 'CMD', 'CAPSTART',' CUSTOMER0001', 'P');
   $ db2 "insert into IBMQREP.SIGNAL (
SIGNAL_TIME,
SIGNAL_TYPE,
SIGNAL_SUBTYPE,
SIGNAL_INPUT_IN,
SIGNAL_STATE
) values (
CURRENT TIMESTAMP, 'CMD', 'CAPSTART','STOCKS0001', 'P');
$ db2 connect reset
```

8. If the subscription was created as an automatic load and the load method has not been changed, then the subscriptions will automatically reload. If the load was set to manual, it is the administrator's responsibility to perform the load and send LOADDONE messages to Q Capture.

> **Note:** This problem may be prevented by defining a dead letter queue and dead letter queue handler.

CAPSTOP signal issued on subscription state not I

When a subscription is not in an inactive (I) state but in an internal state (T or G) and a CAPSTOP signal is issued, it results in the error messages shown in Example D-2. A Q subscription hung in an internal state cannot be stopped (CAPSTOP). To correct the problem, the subscription must be restarted and go through a full refresh.

Example: D-2 CAPSTART signal on subscription not in I state

```
2004-11-02-13.59.50.652170 ASN7019I  "Q Capture" : "EAST" :
"WorkerThread" : "P2PSUBSTOPPING" signal was received and will be
processed.

2004-11-02-13.59.50.657769 ASN7094E  "Q Capture" : "EAST" :
"WorkerThread" : The "P2PSUBSTOPPING" signal failed because the XML
publication or Q subscription "BAL0001" has an invalid  subtype "B" or
an invalid state "I".
```

```
2004-11-02-13.59.50.680819 ASN7019I  "Q Capture" : "EAST" :
"WorkerThread" : "P2PSUBSTOPPING" signal was received and will be
processed.

2004-11-02-13.59.50.687768 ASN7094E  "Q Capture" : "EAST" :
"WorkerThread" : The "P2PSUBSTOPPING" signal failed because the XML
publication or Q subscription "OPTIONS0001" has an invalid  subtype "B"
or an invalid state "I".

2004-11-02-13.59.50.693540 ASN7019I  "Q Capture" : "EAST" :
"WorkerThread" : "P2PSUBSTOPPING" signal was received and will be
processed.
2004-11-02-13.59.50.699757 ASN7094E  "Q Capture" : "EAST" :
"WorkerThread" : The "P2PSUBSTOPPING" signal failed because the XML
publication or Q subscription "TRAN0001" has an invalid  subtype "B" or
an invalid state "I".
```

The resolution to this problem is to issue a CAPSTART signal to restart (and reload) the subscription, as follows:

```
$ db2 connect to STOCK_E
   $ db2 "insert into IBMQREP.SIGNAL (
SIGNAL_TIME,
SIGNAL_TYPE,
SIGNAL_SUBTYPE,
SIGNAL_INPUT_IN,
SIGNAL_STATE
) values (
CURRENT TIMESTAMP, 'CMD', 'CAPSTART',' ACCOUNT0001', 'P');
   $ db2 "insert into IBMQREP.SIGNAL (
SIGNAL_TIME,
SIGNAL_TYPE,
SIGNAL_SUBTYPE,
SIGNAL_INPUT_IN,
SIGNAL_STATE
) values (
CURRENT TIMESTAMP, 'CMD', 'CAPSTART',' CUSTOMER0001', 'P');
   $ db2 "insert into IBMQREP.SIGNAL (
SIGNAL_TIME,
SIGNAL_TYPE,
SIGNAL_SUBTYPE,
SIGNAL_INPUT_IN,
SIGNAL_STATE
) values (
CURRENT TIMESTAMP, 'CMD', 'CAPSTART','STOCKS0001', 'P');
$ db2 connect reset
```

Synchronizing tables

Q replication has two utilities (asntdiff and asntrep) for checking and repairing synchronization between target and source tables.

- Asntdiff determines any differences between the tables in a subscription and writes its results to a database table.
- Asntrep uses the information in the table created by anstdiff to resynchronize the data.

Subscription deactivation

An active subscription may be deactivated for a number of reasons, including the following:

- When an error occurs at one of the WebSphere MQ queues that is associated with a replication queue map, the Q subscriptions that use that queue are deactivated.
- When an invalid search condition is evaluated by the Q Capture program, the subscription is deactivated.
- When the Q Apply program encounters an unexpected condition in the target data, it only deactivates (or stops) the Q subscription where the unexpected condition occurred. However, the Q Apply program continues to apply changes for the other Q subscriptions.
- When the Q Apply program encounters an error, it only deactivates (or stops) the Q subscription where the error occurred. However, the Q Apply program continues to apply changes for the other Q subscriptions.
- If the Q Capture program can no longer write to the send queue, then the Q Capture program either deactivates all Q subscriptions that use that send queue or it shuts itself down, depending on the error option specified for the replication queue map.
- If a required load utility is not available during automatic load, the Q Apply program deactivates the Q subscription.

E

Startup and shutdown scripts for WebSphere MQ and Q replication

In this appendix we describe scripts for starting and shutting down WebSphere MQ and Q replication programs.

Startup and shutdown scripts

A number of scripts are provided here to start and stop WebSphere MQ and Q replication capture and apply programs. These scripts need to be executed on appropriate source and target servers.

Example E-1 shows the script that sets up the various environment variables and starts up the queue manager, the WebSphere MQ channels, DLQ handlers, listener port, and the Q Capture and Q Apply programs.

Example: E-1 Startup script for WebSphere MQ and Q replication

```
#!/usr/sbin/ksh
#
##############################################################################
# Information Integrator Q-Replication startup script
# This script assumes that control of the WebSphere MQ
# system is under control of the DB2 Replication DBA.
# If no, comment out the first four commands.
#
# Set up the environment
# Fill in the following script variables
#
##############################################################################
LOGFILE=/tmp/qrep_startup.log
DB2DIR=
MQ_DIR=
QMGR=
CHANNEL=
PORT=
SCHEMA=
CAPTURE_PATH=
APPLY_PATH=
# If using a DLQ manager
RULEFILE=
##############################################################################
#
# Start WebSphere MQ Queue Manager
#
$MQ_DIR/bin/strmqm $QMGR >> $LOGFILE 2>&1

#
# Start WebSphere MQ Channels
#
$MQ_DIR/bin/runmqsc $QMGR <<++END++ >> $LOGFILE 2>&1
start channel($CHANNEL)
end
++END++
```

```
#
# Start WebSphere MQ DLQ handlers
#
nohup $MQ_DIR/bin/runmqdlq $QMGR < $RULEFILE >> $LOGFILE 2>&1 &

#
# Start WebSphere MQ Listener
#
nohup $MQ_DIR/bin/runmqlsr -m $QMGR -t TCP -p $PORT >> $LOGFILE 2>&1 &

#
# Start the Capture Program
#
$DB2DIR/bin/asnqcap \
   CAPTURE_PATH=$CAPTURE_PATH
   CAPTURE_SCHEMA=$SCHEMA \
   CAPTURE_SERVER=$QMGR \
   LOGSTDOUT=Y \
   STARTMODE=WARMSI >> $LOGFILE 2>&1

#
# Start the Apply Program
#
$DB2DIR/bin/asnqapp \
   APPLY_PATH=$APPLY_PATH
   APPLY_SCHEMA=$SCHEMA \
   APPLY_SERVER=$QMGR \
   LOGSTDOUT=Y >> $LOGFILE 2>&1
```

Example E-2 shows the script that sets up the various environment variables and stops the Q Capture and Q Apply programs, the WebSphere MQ channels, the queue manager, and the listeners.

Example: E-2 Shutdown script for WebSphere MQ and Q replication

```
#!/usr/sbin/ksh
#
##############################################################################
# Information Integrator Q-Replication shutdown script
#
# This script assumes that control of the WebSphere MQ
# system is under control of the DB2 Replication DBA.
# If no, comment out the last three commands.
#
##############################################################################
#
# Set up the environment
# Fill in the following script variables
##############################################################################
```

```
LOGFILE=/tmp/qrep_shutdown.log
DB2DIR=
MQ_DIR=
QMGR=
CHANNEL=
SCHEMA=
CAPTURE_PATH=
APPLY_PATH=
###############################################################################

print `date`" - Q Replication Shutdown." >> $LOGFILE

#
# Stop the Capture Program
#
$DB2DIR/bin/asnqccmd \
   CAPTURE_SCHEMA=$SCHEMA \
   CAPTURE_SERVER=$QMGR \
   LOGSTDOUT=Y STOP >> $LOGFILE 2>&1

#
# Stop the Apply Program
#
$DB2DIR/bin/asnqacmd \
   APPLY_SCHEMA=$SCHEMA \
   APPLY_SERVER=$QMGR \
   LOGSTDOUT=Y STOP >> $LOGFILE 2>&1

#
# Stop WebSphere MQ Channels
#
$MQ_DIR/bin/runmqsc $QMGR <<++END++ >> $LOGFILE 2>&1
stop channel($CHANNEL)
end
++END++

#
# Stop WebSphere MQ Queue Managers
#
$MQ_DIR/bin/endmqm -i $QMGR >> $LOGFILE 2>&1

#
# Stop WebSphere MQ Listeners
#
$MQ_DIR/bin/endmqlsr -m $QMGR >> $LOGFILE 2>&1
```

F

Tables used in Q replication scenarios

In this appendix we describe the tables used in the Q replication bidirectional and peer-to-peer scenarios on the AIX and z/OS platforms.

Tables used in the Q replication scenarios on AIX

We created a set of simple test tables for the bidirectional and peer-to-peer scenarios to include referential integrity constraints and a LOB data type.

Example F-1 shows the relationship between the tables used in the bidirectional scenario—table BAL has a foreign key relationship to the TRAN table, while the OPTIONS table has a CLOB data type column. Example F-1 shows the DDL used in creating the three "financial" tables (TRAN, BAL, and OPTIONS).

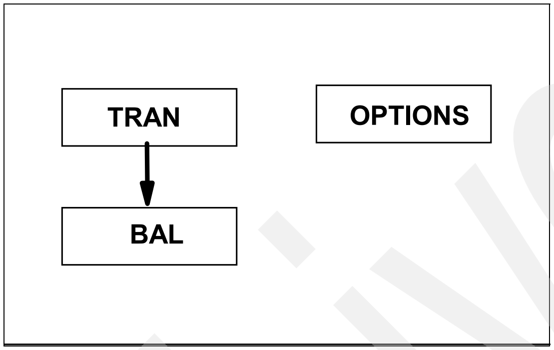


Figure F-1 Financial tables in bidirectional scenario for AIX

Example: F-1 Financial tables DDL in bidirectional scenario for AIX

```
CREATE TABLE DB2INST1.TRAN
(
 COL1 INTEGER,
 COL2 DATE,
 COL3 INTEGER NOT NULL,
 COL4 DATE,
 COL5 INTEGER NOT NULL,
 COL6 INTEGER NOT NULL,
 COL7 INTEGER,
 COL8 TIME NOT NULL,
 COL9 TIME,
 COL10 DOUBLE NOT NULL,
 COL11 CHARACTER(168),
 COL12 DOUBLE,
 COL13 DOUBLE NOT NULL,
 COL14 INTEGER NOT NULL,
 COL15 CHARACTER(52) NOT NULL,
 COL16 TIME NOT NULL,
 COL17 INTEGER NOT NULL,
 COL18 DOUBLE,
 COL19 DATE,
 COL20 DATE,
```

```
 COL21 CHARACTER(247) NOT NULL,
 COL22 DATE NOT NULL,
 COL23 DOUBLE,
 COL24 TIME,
 COL25 DATE,
 COL26 DOUBLE NOT NULL,
 COL27 CHARACTER(204) NOT NULL,
 COL28 TIME,
 COL29 DATE NOT NULL,
 COL30 CHARACTER(64),
 COL31 INTEGER,
 COL32 INTEGER NOT NULL,
 COL33 TIME,
 COL34 DOUBLE,
 COL35 DOUBLE NOT NULL,
 COL36 TIME NOT NULL,
 COL37 INTEGER,
 COL38 INTEGER,
 COL39 CHARACTER(235) NOT NULL,
 COL40 DATE,
 COL41 DATE NOT NULL,
 COL42 CHARACTER(198),
 COL43 TIME,
 COL44 DOUBLE,
 COL45 DOUBLE,
 COL46 DATE,
 COL47 DATE,
 CONSTRAINT CC1098193744412 PRIMARY KEY(COL3)
)
 DATA CAPTURE CHANGES
 IN USERSPACE1;
--------------------------------------------------------------------
CREATE TABLE DB2INST1.BAL
(
 COL1 CHARACTER(249) NOT NULL,
 COL2 CHARACTER(14) NOT NULL,
 COL3 INTEGER,
 COL4 CHARACTER(105) NOT NULL,
 COL5 DOUBLE,
 COL6 CHARACTER(46) NOT NULL,
 COL7 DOUBLE,
 COL8 DATE,
 COL9 CHARACTER(124) NOT NULL,
 COL10 DOUBLE,
 COL11 TIME,
 COL12 CHARACTER(52),
 COL13 DATE,
 COL14 DOUBLE NOT NULL,
 COL15 TIME NOT NULL,
```

```
 COL16 DATE,
 COL17 DOUBLE,
 COL18 TIME,
 COL19 DOUBLE NOT NULL,
 CONSTRAINT CC1098193766934 PRIMARY KEY(COL19),
 CONSTRAINT CC1098193801053 FOREIGN KEY(COL3) REFERENCES
 DB2INST1.TRAN(COL3) ON DELETE CASCADE ON UPDATE NO ACTION
)
 DATA CAPTURE CHANGES
 IN USERSPACE1;
-------------------------------------------------------------------
CREATE TABLE DB2INST1.OPTIONS
(
 COL1 CLOB(4765) NOT LOGGED NOT COMPACT NOT NULL,
 COL2 DATE NOT NULL,
 CONSTRAINT CC1098193900817 PRIMARY KEY(COL2)
)
 DATA CAPTURE CHANGES
 IN USERSPACE1;
```

Figure F-2 shows the relationship between the tables used in the peer-to-peer scenario—table CUSTOMER has a foreign key relationship to the ACCOUNT table, while the STOCKS table has no relationship with the other tables. Example F-2 shows the DDL used in creating the three financial tables ACCOUNT, CUSTOMER, and STOCKS.

Attention: These three tables show the two additional columns (data type of TIMESTAMP and SMALLINT) and the associated triggers that are added in support of peer-to-peer replication. These are highlighted in bold in Example F-2.

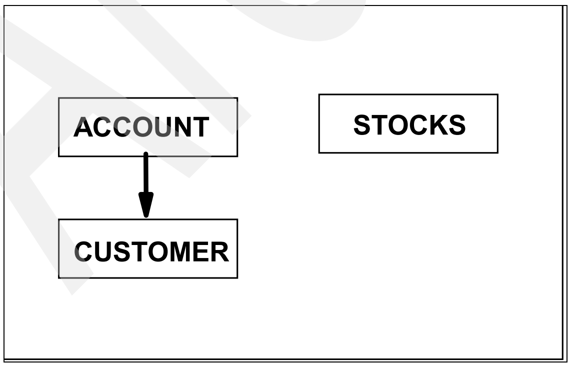


Figure F-2 Financial tables in the peer-to-peer scenario for AIX

Example: F-2 Financial tables DDL in the peer-to-peer scenario for AIX

```
CREATE TABLE DB2INST1.ACCOUNT
(
 COL1 INTEGER,
 COL2 DATE,
 COL3 INTEGER NOT NULL,
 COL4 DATE,
 COL5 INTEGER NOT NULL,
 COL6 INTEGER NOT NULL,
 COL7 INTEGER,
 COL8 TIME NOT NULL,
 COL9 TIME,
 COL10 DOUBLE NOT NULL,
 COL11 CHARACTER(168),
 COL12 DOUBLE,
 COL13 DOUBLE NOT NULL,
 COL14 INTEGER NOT NULL,
 COL15 CHARACTER(52) NOT NULL,
 COL16 TIME NOT NULL,
 COL17 INTEGER NOT NULL,
 COL18 DOUBLE,
 COL19 DATE,
 COL20 DATE,
 COL21 CHARACTER(247) NOT NULL,
 COL22 DATE NOT NULL,
 COL23 DOUBLE,
 COL24 TIME,
 COL25 DATE,
 COL26 DOUBLE NOT NULL,
 COL27 CHARACTER(204) NOT NULL,
 COL28 TIME,
 COL29 DATE NOT NULL,
 COL30 CHARACTER(64),
 COL31 INTEGER,
 COL32 INTEGER NOT NULL,
 COL33 TIME,
 COL34 DOUBLE,
 COL35 DOUBLE NOT NULL,
 COL36 TIME NOT NULL,
 COL37 INTEGER,
 COL38 INTEGER,
 COL39 CHARACTER(235) NOT NULL,
 COL40 DATE,
 COL41 DATE NOT NULL,
 COL42 CHARACTER(198),
 COL43 TIME,
 COL44 DOUBLE,
 COL45 DOUBLE,
```

```
 COL46 DATE,
 COL47 DATE,
 "ibmqrepVERTIME" TIMESTAMP NOT NULL WITH DEFAULT
 '0001-01-01-00.00.00',
 "ibmqrepVERNODE" SMALLINT NOT NULL WITH DEFAULT 0,
 PRIMARY KEY(COL3)
)
 DATA CAPTURE CHANGES
 IN USERSPACE1#
CREATE TRIGGER DB2INST1."1ACCOUNTUNT3" NO CASCADE BEFORE INSERT ON
 DB2INST1.ACCOUNT
 REFERENCING NEW AS new FOR EACH ROW MODE DB2SQL
 WHEN (new."ibmqrepVERNODE" = 0)
BEGIN ATOMIC
SET
 new."ibmqrepVERTIME" = (CURRENT TIMESTAMP - CURRENT TIMEZONE),

new."ibmqrepVERNODE" = 1*4;
END#
CREATE TRIGGER DB2INST1."2ACCOUNTUNT3" NO CASCADE BEFORE UPDATE ON
 DB2INST1.ACCOUNT
 REFERENCING NEW AS new OLD AS old FOR EACH ROW MODE DB2SQL
 WHEN ((new."ibmqrepVERTIME" = old."ibmqrepVERTIME")
AND ((
(new."ibmqrepVERNODE")/4) = ((old."ibmqrepVERNODE")/4)))
BEGIN ATOMIC

SET new."ibmqrepVERTIME" =
CASE
WHEN (CURRENT TIMESTAMP - CURRENT
 TIMEZONE) < old."ibmqrepVERTIME"
THEN old."ibmqrepVERTIME" +
 00000000000000.000001
ELSE (CURRENT TIMESTAMP - CURRENT TIMEZONE)

END,
new."ibmqrepVERNODE" = 1*4;
END#
----------------------------------------------------------------------

CREATE TABLE DB2INST1.CUSTOMER
(
 COL1 CHARACTER(249) NOT NULL,
 COL2 CHARACTER(14) NOT NULL,
 COL3 INTEGER,
 COL4 CHARACTER(105) NOT NULL,
 COL5 DOUBLE,
 COL6 CHARACTER(46) NOT NULL,
 COL7 DOUBLE,
```

```
 COL8 DATE,
 COL9 CHARACTER(124) NOT NULL,
 COL10 DOUBLE,
 COL11 TIME,
 COL12 CHARACTER(52),
 COL13 DATE,
 COL14 DOUBLE NOT NULL,
 COL15 TIME NOT NULL,
 COL16 DATE,
 COL17 DOUBLE,
 COL18 TIME,
 COL19 DOUBLE NOT NULL,
 "ibmqrepVERTIME" TIMESTAMP NOT NULL WITH DEFAULT
 '0001-01-01-00.00.00',
 "ibmqrepVERNODE" SMALLINT NOT NULL WITH DEFAULT 0,
 PRIMARY KEY(COL19),
 FOREIGN KEY(COL3) REFERENCES DB2INST1.ACCOUNT(COL3) ON DELETE
 CASCADE ON UPDATE NO ACTION
)
 DATA CAPTURE CHANGES
 IN USERSPACE1#
CREATE TRIGGER DB2INST1."1CUSTOMERMER3" NO CASCADE BEFORE INSERT ON
 DB2INST1.CUSTOMER
 REFERENCING NEW AS new FOR EACH ROW MODE DB2SQL
 WHEN (new."ibmqrepVERNODE" = 0)
BEGIN ATOMIC
SET
 new."ibmqrepVERTIME" = (CURRENT TIMESTAMP - CURRENT TIMEZONE),

new."ibmqrepVERNODE" = 1*4;
END#
CREATE TRIGGER DB2INST1."2CUSTOMERMER3" NO CASCADE BEFORE UPDATE ON
 DB2INST1.CUSTOMER
 REFERENCING NEW AS new OLD AS old FOR EACH ROW MODE DB2SQL
 WHEN ((new."ibmqrepVERTIME" = old."ibmqrepVERTIME")
AND ((
(new."ibmqrepVERNODE")/4) = ((old."ibmqrepVERNODE")/4)))
BEGIN ATOMIC

SET new."ibmqrepVERTIME" =
CASE
WHEN (CURRENT TIMESTAMP - CURRENT
 TIMEZONE) < old."ibmqrepVERTIME"
THEN old."ibmqrepVERTIME" +
 00000000000000.000001
ELSE (CURRENT TIMESTAMP - CURRENT TIMEZONE)

END,
new."ibmqrepVERNODE" = 1*4;
```

```
END#
----------------------------------------------------------------------
CREATE TABLE DB2INST1.STOCKS
(
 COL1 CHARACTER(189),
 COL2 CHARACTER(2) NOT NULL,
 COL3 DATE,
 COL4 TIME NOT NULL,
 "ibmqrepVERTIME" TIMESTAMP NOT NULL WITH DEFAULT
 '0001-01-01-00.00.00',
 "ibmqrepVERNODE" SMALLINT NOT NULL WITH DEFAULT 0,
 CONSTRAINT CC1098459773803 PRIMARY KEY(COL2)
)
 DATA CAPTURE CHANGES
 IN USERSPACE1#
CREATE TRIGGER DB2INST1."1STOCKSCKS3" NO CASCADE BEFORE INSERT ON
 DB2INST1.STOCKS
 REFERENCING NEW AS new FOR EACH ROW MODE DB2SQL
 WHEN (new."ibmqrepVERNODE" = 0)
BEGIN ATOMIC
SET
 new."ibmqrepVERTIME" = (CURRENT TIMESTAMP - CURRENT TIMEZONE),

new."ibmqrepVERNODE" = 1*4;
END#
CREATE TRIGGER DB2INST1."2STOCKSCKS3" NO CASCADE BEFORE UPDATE ON
 DB2INST1.STOCKS
 REFERENCING NEW AS new OLD AS old FOR EACH ROW MODE DB2SQL
 WHEN ((new."ibmqrepVERTIME" = old."ibmqrepVERTIME")
AND ((
(new."ibmqrepVERNODE")/4) = ((old."ibmqrepVERNODE")/4)))
BEGIN ATOMIC

SET new."ibmqrepVERTIME" =
CASE
WHEN (CURRENT TIMESTAMP - CURRENT
 TIMEZONE) < old."ibmqrepVERTIME"
THEN old."ibmqrepVERTIME" +
 00000000000000.000001
ELSE (CURRENT TIMESTAMP - CURRENT TIMEZONE)

END,
new."ibmqrepVERNODE" = 1*4;
END#
```

Tables used in the Q replication scenarios on z/OS

We created a set of 23 test tables for the bidirectional and peer-to-peer scenarios to include referential integrity constraints between some of the tables. These tables did not include a LOB data type.

Example F-3 shows the DDL used in creating the 23 financial tables that were used in both the bidirectional and peer-to-peer replication scenarios.

Example: F-3 Financial tables DDL in bidirectional & peer-to-peer scenarios for z/OS

```
SET CURRENT SQLID = 'QREPADM';

CREATE STOGROUP ITSOASTG
     VOLUMES ("TOTDDM")
     VCAT DB8GU ;

COMMIT WORK;

CREATE STOGROUP ITSIXSG1
     VOLUMES ("TOTDDM")
     VCAT DB8GU ;

COMMIT WORK;

  CREATE DATABASE ITSBADB1
     STOGROUP ITSOASTG
     BUFFERPOOL BP0;

  COMMIT WORK;

  CREATE DATABASE ITSBADB2
     STOGROUP ITSOASTG
     BUFFERPOOL BP0;

  COMMIT WORK;

  CREATE DATABASE ITSTHDB2
     STOGROUP ITSOASTG
     BUFFERPOOL BP0;

  COMMIT WORK;

  CREATE DATABASE ITSTRDB1
     STOGROUP ITSOASTG
     BUFFERPOOL BP0;

  COMMIT WORK;
```

```
  CREATE DATABASE ITSTRDB2
     STOGROUP ITSOASTG
     BUFFERPOOL BP0;

  COMMIT WORK;

  CREATE DATABASE ITSTRDB8
     STOGROUP ITSOASTG
     BUFFERPOOL BP0;

  COMMIT WORK;

  CREATE DATABASE ITSCODB1
     STOGROUP ITSOASTG
     BUFFERPOOL BP0;

  COMMIT WORK;

  CREATE DATABASE ITSITDB1
     STOGROUP ITSOASTG
     BUFFERPOOL BP0;

  COMMIT WORK;

---====================== TABLESPACE DEFINITIONS =====================

CREATE  TABLESPACE ITTSBLN IN ITSBADB1
  NUMPARTS   9
  ( PART  1 USING STOGROUP ITSOASTG
                PRIQTY 1296
                SECQTY 648
          FREEPAGE   25
          PCTFREE    20
          COMPRESS YES
  , PART  2 USING STOGROUP ITSOASTG
                PRIQTY 842
                SECQTY 432
          FREEPAGE   25
          PCTFREE    20
          COMPRESS YES
  , PART  3 USING STOGROUP ITSOASTG
                PRIQTY 741
                SECQTY 396
          FREEPAGE   25
          PCTFREE    20
          COMPRESS YES
  , PART  4 USING STOGROUP ITSOASTG
                PRIQTY 792
                SECQTY 396
```

```
              FREEPAGE   25
              PCTFREE    20
              COMPRESS YES
    , PART  5 USING STOGROUP ITSOASTG
                    PRIQTY 806
                    SECQTY 432
              FREEPAGE   25
              PCTFREE    20
              COMPRESS YES
    , PART  6 USING STOGROUP ITSOASTG
                    PRIQTY 633
                    SECQTY 324
              FREEPAGE   25
              PCTFREE    20
              COMPRESS YES
    , PART  7 USING STOGROUP ITSOASTG
                    PRIQTY 813
                    SECQTY 432
              FREEPAGE   25
              PCTFREE    20
              COMPRESS YES
    , PART  8 USING STOGROUP ITSOASTG
                    PRIQTY 799
                    SECQTY 432
              FREEPAGE   25
              PCTFREE    20
              COMPRESS YES
    , PART  9 USING STOGROUP ITSOASTG
                    PRIQTY 453
                    SECQTY 252
              FREEPAGE   25
              PCTFREE    20
              COMPRESS YES
    )
    BUFFERPOOL BP1
    LOCKSIZE   PAGE
    CLOSE      NO
    MAXROWS    255
    CCSID      EBCDIC
                                                              ;
  SET CURRENT SQLID = 'QREPADM';
  CREATE  TABLESPACE ITTSBPT IN ITSBADB1
    SEGSIZE    64
    USING STOGROUP ITSOASTG
          PRIQTY 14
          SECQTY 36
    PCTFREE    0
    BUFFERPOOL BP1
    LOCKSIZE   PAGE
```

```
  CLOSE      NO
  MAXROWS    255
  CCSID      EBCDIC
                                                                          ;
SET CURRENT SQLID = 'QREPADM';
CREATE  TABLESPACE ITTSOB1 IN ITSBADB1
  NUMPARTS   18
  ( PART  1 USING STOGROUP ITSOASTG
                  PRIQTY 1152
                  SECQTY 576
            COMPRESS YES
  , PART  2 USING STOGROUP ITSOASTG
                  PRIQTY 756
                  SECQTY 396
            COMPRESS YES
  , PART  3 USING STOGROUP ITSOASTG
                  PRIQTY 907
                  SECQTY 468
            COMPRESS YES
  , PART  4 USING STOGROUP ITSOASTG
                  PRIQTY 763
                  SECQTY 396
            COMPRESS YES
  , PART  5 USING STOGROUP ITSOASTG
                  PRIQTY 828
                  SECQTY 432
            COMPRESS YES
  , PART  6 USING STOGROUP ITSOASTG
                  PRIQTY 597
                  SECQTY 324
            COMPRESS YES
  , PART  7 USING STOGROUP ITSOASTG
                  PRIQTY 669
                  SECQTY 360
            COMPRESS YES
  , PART  8 USING STOGROUP ITSOASTG
                  PRIQTY 583
                  SECQTY 324
            COMPRESS YES
  , PART  9 USING STOGROUP ITSOASTG
                  PRIQTY 669
                  SECQTY 360
            COMPRESS YES
  , PART 10 USING STOGROUP ITSOASTG
                  PRIQTY 561
                  SECQTY 288
            COMPRESS YES
  , PART 11 USING STOGROUP ITSOASTG
                  PRIQTY 655
```

```
                SECQTY 360
          COMPRESS YES
  , PART 12 USING STOGROUP ITSOASTG
                PRIQTY 604
                SECQTY 324
          COMPRESS YES
  , PART 13 USING STOGROUP ITSOASTG
                PRIQTY 669
                SECQTY 360
          COMPRESS YES
  , PART 14 USING STOGROUP ITSOASTG
                PRIQTY 532
                SECQTY 288
          COMPRESS YES
  , PART 15 USING STOGROUP ITSOASTG
                PRIQTY 640
                SECQTY 324
          COMPRESS YES
  , PART 16 USING STOGROUP ITSOASTG
                PRIQTY 561
                SECQTY 288
          COMPRESS YES
  , PART 17 USING STOGROUP ITSOASTG
                PRIQTY 648
                SECQTY 324
          COMPRESS YES
  , PART 18 USING STOGROUP ITSOASTG
                PRIQTY 288
                SECQTY 144
          COMPRESS YES
  )
  BUFFERPOOL BP1
  LOCKSIZE   PAGE
  CLOSE      NO
  MAXROWS    255
  CCSID      EBCDIC
                                                            ;
SET CURRENT SQLID = 'QREPADM';
CREATE  TABLESPACE ITTSPOS IN ITSBADB1
  NUMPARTS   9
  ( PART  1 USING STOGROUP ITSOASTG
                PRIQTY 2664
                SECQTY 1310
          FREEPAGE   25
          PCTFREE    20
          COMPRESS YES
  , PART  2 USING STOGROUP ITSOASTG
                PRIQTY 2023
                SECQTY 1044
```

```
          FREEPAGE   25
          PCTFREE    20
          COMPRESS YES
, PART  3 USING STOGROUP ITSOASTG
                PRIQTY 1792
                SECQTY 900
          FREEPAGE   25
          PCTFREE    20
          COMPRESS YES
, PART  4 USING STOGROUP ITSOASTG
                PRIQTY 1591
                SECQTY 828
          FREEPAGE   25
          PCTFREE    20
          COMPRESS YES
, PART  5 USING STOGROUP ITSOASTG
                PRIQTY 1576
                SECQTY 792
          FREEPAGE   25
          PCTFREE    20
          COMPRESS YES
, PART  6 USING STOGROUP ITSOASTG
                PRIQTY 1598
                SECQTY 828
          FREEPAGE   25
          PCTFREE    20
          COMPRESS YES
, PART  7 USING STOGROUP ITSOASTG
                PRIQTY 1562
                SECQTY 792
          FREEPAGE   25
          PCTFREE    20
          COMPRESS YES
, PART  8 USING STOGROUP ITSOASTG
                PRIQTY 1562
                SECQTY 792
          FREEPAGE   25
          PCTFREE    20
          COMPRESS YES
, PART  9 USING STOGROUP ITSOASTG
                PRIQTY 1209
                SECQTY 612
          FREEPAGE   25
          PCTFREE    20
          COMPRESS YES
)
BUFFERPOOL BP1
LOCKSIZE   PAGE
CLOSE      NO
```

```
  MAXROWS    255
  CCSID      EBCDIC
                                                                   ;
SET CURRENT SQLID = 'QREPADM';
CREATE  TABLESPACE ITTSOB1 IN ITSBADB2
  NUMPARTS   18
  ( PART  1 USING STOGROUP ITSOASTG
                  PRIQTY 1152
                  SECQTY 576
            COMPRESS YES
  , PART  2 USING STOGROUP ITSOASTG
                  PRIQTY 763
                  SECQTY 396
            COMPRESS YES
  , PART  3 USING STOGROUP ITSOASTG
                  PRIQTY 907
                  SECQTY 468
            COMPRESS YES
  , PART  4 USING STOGROUP ITSOASTG
                  PRIQTY 756
                  SECQTY 396
            COMPRESS YES
  , PART  5 USING STOGROUP ITSOASTG
                  PRIQTY 828
                  SECQTY 432
            COMPRESS YES
  , PART  6 USING STOGROUP ITSOASTG
                  PRIQTY 597
                  SECQTY 324
            COMPRESS YES
  , PART  7 USING STOGROUP ITSOASTG
                  PRIQTY 648
                  SECQTY 324
            COMPRESS YES
  , PART  8 USING STOGROUP ITSOASTG
                  PRIQTY 583
                  SECQTY 324
            COMPRESS YES
  , PART  9 USING STOGROUP ITSOASTG
                  PRIQTY 669
                  SECQTY 360
            COMPRESS YES
  , PART 10 USING STOGROUP ITSOASTG
                  PRIQTY 561
                  SECQTY 288
            COMPRESS YES
  , PART 11 USING STOGROUP ITSOASTG
                  PRIQTY 640
                  SECQTY 324
```

```
        COMPRESS YES
, PART 12 USING STOGROUP ITSOASTG
              PRIQTY 597
              SECQTY 324
        COMPRESS YES
, PART 13 USING STOGROUP ITSOASTG
              PRIQTY 669
              SECQTY 360
        COMPRESS YES
, PART 14 USING STOGROUP ITSOASTG
              PRIQTY 525
              SECQTY 288
        COMPRESS YES
, PART 15 USING STOGROUP ITSOASTG
              PRIQTY 626
              SECQTY 324
        COMPRESS YES
, PART 16 USING STOGROUP ITSOASTG
              PRIQTY 561
              SECQTY 288
        COMPRESS YES
, PART 17 USING STOGROUP ITSOASTG
              PRIQTY 633
              SECQTY 324
        COMPRESS YES
, PART 18 USING STOGROUP ITSOASTG
              PRIQTY 288
              SECQTY 144
        COMPRESS YES
)
BUFFERPOOL BP1
LOCKSIZE   PAGE
CLOSE      NO
MAXROWS    255
CCSID      EBCDIC
                                                       ;
SET CURRENT SQLID = 'QREPADM';
CREATE  TABLESPACE ITTSUIR IN ITSTHDB2
  NUMPARTS   9
  ( PART  1 USING STOGROUP ITSOASTG
              PRIQTY 532
              SECQTY 288
        PCTFREE    15
        COMPRESS YES
  , PART  2 USING STOGROUP ITSOASTG
              PRIQTY 374
              SECQTY 216
        PCTFREE    15
        COMPRESS YES
```

```
  , PART  3 USING STOGROUP ITSOASTG
                  PRIQTY 324
                  SECQTY 180
            PCTFREE    15
            COMPRESS YES
  , PART  4 USING STOGROUP ITSOASTG
                  PRIQTY 280
                  SECQTY 144
            PCTFREE    15
            COMPRESS YES
  , PART  5 USING STOGROUP ITSOASTG
                  PRIQTY 273
                  SECQTY 144
            PCTFREE    15
            COMPRESS YES
  , PART  6 USING STOGROUP ITSOASTG
                  PRIQTY 273
                  SECQTY 144
            PCTFREE    15
            COMPRESS YES
  , PART  7 USING STOGROUP ITSOASTG
                  PRIQTY 273
                  SECQTY 144
            PCTFREE    15
            COMPRESS YES
  , PART  8 USING STOGROUP ITSOASTG
                  PRIQTY 273
                  SECQTY 144
            PCTFREE    15
            COMPRESS YES
  , PART  9 USING STOGROUP ITSOASTG
                  PRIQTY 208
                  SECQTY 108
            PCTFREE    15
            COMPRESS YES
  )
  BUFFERPOOL BP1
  LOCKSIZE   PAGE
  CLOSE      NO
  MAXROWS    255
  CCSID      EBCDIC
                                                                     ;
SET CURRENT SQLID = 'QREPADM';
CREATE  TABLESPACE ITTSLFC IN ITSTRDB1
  NUMPARTS   9
  ( PART  1 USING STOGROUP ITSOASTG
                  PRIQTY 446
                  SECQTY 252
            PCTFREE    10
```

```
            COMPRESS YES
  , PART  2 USING STOGROUP ITSOASTG
                  PRIQTY 230
                  SECQTY 144
            PCTFREE    10
            COMPRESS YES
  , PART  3 USING STOGROUP ITSOASTG
                  PRIQTY 201
                  SECQTY 108
            PCTFREE    10
            COMPRESS YES
  , PART  4 USING STOGROUP ITSOASTG
                  PRIQTY 172
                  SECQTY 108
            PCTFREE    10
            COMPRESS YES
  , PART  5 USING STOGROUP ITSOASTG
                  PRIQTY 172
                  SECQTY 108
            PCTFREE    10
            COMPRESS YES
  , PART  6 USING STOGROUP ITSOASTG
                  PRIQTY 172
                  SECQTY 108
            PCTFREE    10
            COMPRESS YES
  , PART  7 USING STOGROUP ITSOASTG
                  PRIQTY 165
                  SECQTY 108
            PCTFREE    10
            COMPRESS YES
  , PART  8 USING STOGROUP ITSOASTG
                  PRIQTY 187
                  SECQTY 108
            PCTFREE    10
            COMPRESS YES
  , PART  9 USING STOGROUP ITSOASTG
                  PRIQTY 122
                  SECQTY 72
            PCTFREE    10
            COMPRESS YES
  )
  BUFFERPOOL BP1
  LOCKSIZE   PAGE
  CLOSE      NO
  MAXROWS    255
  CCSID      EBCDIC
                                                                       ;
SET CURRENT SQLID = 'QREPADM';
```

```
CREATE  TABLESPACE ITTSLIC IN ITSTRDB1
  NUMPARTS   3
  ( PART  1 USING STOGROUP ITSOASTG
                  PRIQTY 93
                  SECQTY 72
            PCTFREE    10
            COMPRESS YES
  , PART  2 USING STOGROUP ITSOASTG
                  PRIQTY 50
                  SECQTY 36
            PCTFREE    10
            COMPRESS YES
  , PART  3 USING STOGROUP ITSOASTG
                  PRIQTY 50
                  SECQTY 36
            PCTFREE    10
            COMPRESS YES
  )
  BUFFERPOOL BP1
  LOCKSIZE   PAGE
  CLOSE      NO
  MAXROWS    255
  CCSID      EBCDIC
                                                                     ;
SET CURRENT SQLID = 'QREPADM';
CREATE  TABLESPACE ITTSLSD IN ITSTRDB1
  NUMPARTS   9
  ( PART  1 USING STOGROUP ITSOASTG
                  PRIQTY 93
                  SECQTY 72
            PCTFREE    10
            COMPRESS YES
  , PART  2 USING STOGROUP ITSOASTG
                  PRIQTY 72
                  SECQTY 36
            PCTFREE    10
            COMPRESS YES
  , PART  3 USING STOGROUP ITSOASTG
                  PRIQTY 72
                  SECQTY 36
            PCTFREE    10
            COMPRESS YES
  , PART  4 USING STOGROUP ITSOASTG
                  PRIQTY 64
                  SECQTY 36
            PCTFREE    10
            COMPRESS YES
  , PART  5 USING STOGROUP ITSOASTG
                  PRIQTY 64
```

```
                    SECQTY 36
              PCTFREE    10
              COMPRESS YES
      , PART  6 USING STOGROUP ITSOASTG
                    PRIQTY 64
                    SECQTY 36
              PCTFREE    10
              COMPRESS YES
      , PART  7 USING STOGROUP ITSOASTG
                    PRIQTY 64
                    SECQTY 36
              PCTFREE    10
              COMPRESS YES
      , PART  8 USING STOGROUP ITSOASTG
                    PRIQTY 64
                    SECQTY 36
              PCTFREE    10
              COMPRESS YES
      , PART  9 USING STOGROUP ITSOASTG
                    PRIQTY 43
                    SECQTY 36
              PCTFREE    10
              COMPRESS YES
      )
      BUFFERPOOL BP1
      LOCKSIZE   PAGE
      LOCKMAX    SYSTEM
      CLOSE      NO
      MAXROWS    255
      CCSID      EBCDIC
                                                                      ;
    SET CURRENT SQLID = 'QREPADM';
    CREATE  TABLESPACE ITTSLTR IN ITSTRDB1
      NUMPARTS   9
      ( PART  1 USING STOGROUP ITSOASTG
                    PRIQTY 432
                    SECQTY 216
              PCTFREE    10
              COMPRESS YES
      , PART  2 USING STOGROUP ITSOASTG
                    PRIQTY 273
                    SECQTY 144
              PCTFREE    10
              COMPRESS YES
      , PART  3 USING STOGROUP ITSOASTG
                    PRIQTY 230
                    SECQTY 144
              PCTFREE    10
              COMPRESS YES
```

```
  , PART  4 USING STOGROUP ITSOASTG
                  PRIQTY 201
                  SECQTY 108
            PCTFREE    10
            COMPRESS YES
  , PART  5 USING STOGROUP ITSOASTG
                  PRIQTY 201
                  SECQTY 108
            PCTFREE    10
            COMPRESS YES
  , PART  6 USING STOGROUP ITSOASTG
                  PRIQTY 194
                  SECQTY 108
            PCTFREE    10
            COMPRESS YES
  , PART  7 USING STOGROUP ITSOASTG
                  PRIQTY 201
                  SECQTY 108
            PCTFREE    10
            COMPRESS YES
  , PART  8 USING STOGROUP ITSOASTG
                  PRIQTY 201
                  SECQTY 108
            PCTFREE    10
            COMPRESS YES
  , PART  9 USING STOGROUP ITSOASTG
                  PRIQTY 151
                  SECQTY 108
            PCTFREE    10
            COMPRESS YES
  )
  BUFFERPOOL BP1
  LOCKSIZE   PAGE
  CLOSE      NO
  MAXROWS    255
  CCSID      EBCDIC
                                                            ;
SET CURRENT SQLID = 'QREPADM';
CREATE  TABLESPACE ITTSMTL IN ITSTRDB1
  SEGSIZE    64
  USING STOGROUP ITSOASTG
        PRIQTY 5241
        SECQTY 1310
  PCTFREE    10
  BUFFERPOOL BP1
  LOCKSIZE   PAGE
  CLOSE      NO
  MAXROWS    255
  CCSID      EBCDIC
```

;

```
SET CURRENT SQLID = 'QREPADM';
CREATE  TABLESPACE ITTSSTR IN ITSTRDB1
  NUMPARTS   9
  ( PART  1 USING STOGROUP ITSOASTG
                  PRIQTY 201
                  SECQTY 108
            PCTFREE    20
            COMPRESS YES
  , PART  2 USING STOGROUP ITSOASTG
                  PRIQTY 7912
                  SECQTY 432
            PCTFREE    20
            COMPRESS YES
  , PART  3 USING STOGROUP ITSOASTG
                  PRIQTY 122
                  SECQTY 72
            PCTFREE    20
            COMPRESS YES
  , PART  4 USING STOGROUP ITSOASTG
                  PRIQTY 6076
                  SECQTY 432
            PCTFREE    20
            COMPRESS YES
  , PART  5 USING STOGROUP ITSOASTG
                  PRIQTY 6955
                  SECQTY 432
            PCTFREE    20
            COMPRESS YES
  , PART  6 USING STOGROUP ITSOASTG
                  PRIQTY 7833
                  SECQTY 432
            PCTFREE    20
            COMPRESS YES
  , PART  7 USING STOGROUP ITSOASTG
                  PRIQTY 144
                  SECQTY 72
            PCTFREE    20
            COMPRESS YES
  , PART  8 USING STOGROUP ITSOASTG
                  PRIQTY 8193
                  SECQTY 288
            PCTFREE    20
            COMPRESS YES
  , PART  9 USING STOGROUP ITSOASTG
                  PRIQTY 72
                  SECQTY 36
            PCTFREE    20
            COMPRESS YES
```

```
  )
  BUFFERPOOL BP1
  LOCKSIZE   PAGE
  CLOSE      NO
  MAXROWS    255
  CCSID      EBCDIC
                                                                      ;
SET CURRENT SQLID = 'QREPADM';
CREATE  TABLESPACE ITTSTEP IN ITSTRDB1
  NUMPARTS   9
  ( PART  1 USING STOGROUP ITSOASTG
                  PRIQTY 165
                  SECQTY 108
            PCTFREE    20
  , PART  2 USING STOGROUP ITSOASTG
                  PRIQTY 158
                  SECQTY 108
            PCTFREE    20
  , PART  3 USING STOGROUP ITSOASTG
                  PRIQTY 108
                  SECQTY 72
            PCTFREE    20
  , PART  4 USING STOGROUP ITSOASTG
                  PRIQTY 93
                  SECQTY 72
            PCTFREE    20
  , PART  5 USING STOGROUP ITSOASTG
                  PRIQTY 144
                  SECQTY 72
            PCTFREE    20
  , PART  6 USING STOGROUP ITSOASTG
                  PRIQTY 115
                  SECQTY 72
            PCTFREE    20
  , PART  7 USING STOGROUP ITSOASTG
                  PRIQTY 93
                  SECQTY 72
            PCTFREE    20
  , PART  8 USING STOGROUP ITSOASTG
                  PRIQTY 79
                  SECQTY 72
            PCTFREE    20
  , PART  9 USING STOGROUP ITSOASTG
                  PRIQTY 86
                  SECQTY 72
            PCTFREE    20
  )
  BUFFERPOOL BP1
  LOCKSIZE   PAGE
```

```
  CLOSE      NO
  MAXROWS    255
  CCSID      EBCDIC
                                                                   ;
SET CURRENT SQLID = 'QREPADM';
CREATE  TABLESPACE ITTSTMS IN ITSTRDB1
  SEGSIZE    64
  USING STOGROUP ITSOASTG
        PRIQTY 72
        SECQTY 36
  PCTFREE    10
  BUFFERPOOL BP1
  LOCKSIZE   PAGE
  CLOSE      NO
  MAXROWS    255
  CCSID      EBCDIC
                                                                   ;
SET CURRENT SQLID = 'QREPADM';
CREATE  TABLESPACE ITTSTRN IN ITSTRDB1
  NUMPARTS   63
  ( PART  1 USING STOGROUP ITSOASTG
                  PRIQTY 525
                  SECQTY 288
            PCTFREE    20
            COMPRESS YES
  , PART  2 USING STOGROUP ITSOASTG
                  PRIQTY 288
                  SECQTY 144
            PCTFREE    20
            COMPRESS YES
  , PART  3 USING STOGROUP ITSOASTG
                  PRIQTY 417
                  SECQTY 216
            PCTFREE    20
            COMPRESS YES
  , PART  4 USING STOGROUP ITSOASTG
                  PRIQTY 309
                  SECQTY 180
            PCTFREE    20
            COMPRESS YES
  , PART  5 USING STOGROUP ITSOASTG
                  PRIQTY 288
                  SECQTY 144
            PCTFREE    20
            COMPRESS YES
  , PART  6 USING STOGROUP ITSOASTG
                  PRIQTY 266
                  SECQTY 144
            PCTFREE    20
```

```
          COMPRESS YES
, PART  7 USING STOGROUP ITSOASTG
                PRIQTY 266
                SECQTY 144
          PCTFREE    20
          COMPRESS YES
, PART  8 USING STOGROUP ITSOASTG
                PRIQTY 266
                SECQTY 144
          PCTFREE    20
          COMPRESS YES
, PART  9 USING STOGROUP ITSOASTG
                PRIQTY 216
                SECQTY 108
          PCTFREE    20
          COMPRESS YES
, PART 10 USING STOGROUP ITSOASTG
                PRIQTY 432
                SECQTY 216
          PCTFREE    20
          COMPRESS YES
, PART 11 USING STOGROUP ITSOASTG
                PRIQTY 273
                SECQTY 144
          PCTFREE    20
          COMPRESS YES
, PART 12 USING STOGROUP ITSOASTG
                PRIQTY 252
                SECQTY 144
          PCTFREE    20
          COMPRESS YES
, PART 13 USING STOGROUP ITSOASTG
                PRIQTY 266
                SECQTY 144
          PCTFREE    20
          COMPRESS YES
, PART 14 USING STOGROUP ITSOASTG
                PRIQTY 252
                SECQTY 144
          PCTFREE    20
          COMPRESS YES
, PART 15 USING STOGROUP ITSOASTG
                PRIQTY 252
                SECQTY 144
          PCTFREE    20
          COMPRESS YES
, PART 16 USING STOGROUP ITSOASTG
                PRIQTY 273
                SECQTY 144
```

```
          PCTFREE    20
          COMPRESS YES
, PART 17 USING STOGROUP ITSOASTG
                PRIQTY 367
                SECQTY 216
          PCTFREE    20
          COMPRESS YES
, PART 18 USING STOGROUP ITSOASTG
                PRIQTY 288
                SECQTY 144
          PCTFREE    20
          COMPRESS YES
, PART 19 USING STOGROUP ITSOASTG
                PRIQTY 244
                SECQTY 144
          PCTFREE    20
          COMPRESS YES
, PART 20 USING STOGROUP ITSOASTG
                PRIQTY 208
                SECQTY 108
          PCTFREE    20
          COMPRESS YES
, PART 21 USING STOGROUP ITSOASTG
                PRIQTY 201
                SECQTY 108
          PCTFREE    20
          COMPRESS YES
, PART 22 USING STOGROUP ITSOASTG
                PRIQTY 187
                SECQTY 108
          PCTFREE    20
          COMPRESS YES
, PART 23 USING STOGROUP ITSOASTG
                PRIQTY 201
                SECQTY 108
          PCTFREE    20
          COMPRESS YES
, PART 24 USING STOGROUP ITSOASTG
                PRIQTY 360
                SECQTY 180
          PCTFREE    20
          COMPRESS YES
, PART 25 USING STOGROUP ITSOASTG
                PRIQTY 223
                SECQTY 144
          PCTFREE    20
          COMPRESS YES
, PART 26 USING STOGROUP ITSOASTG
                PRIQTY 172
```

```
                SECQTY 108
          PCTFREE    20
          COMPRESS YES
, PART 27 USING STOGROUP ITSOASTG
                PRIQTY 172
                SECQTY 108
          PCTFREE    20
          COMPRESS YES
, PART 28 USING STOGROUP ITSOASTG
                PRIQTY 180
                SECQTY 108
          PCTFREE    20
          COMPRESS YES
, PART 29 USING STOGROUP ITSOASTG
                PRIQTY 187
                SECQTY 108
          PCTFREE    20
          COMPRESS YES
, PART 30 USING STOGROUP ITSOASTG
                PRIQTY 187
                SECQTY 108
          PCTFREE    20
          COMPRESS YES
, PART 31 USING STOGROUP ITSOASTG
                PRIQTY 324
                SECQTY 180
          PCTFREE    20
          COMPRESS YES
, PART 32 USING STOGROUP ITSOASTG
                PRIQTY 266
                SECQTY 144
          PCTFREE    20
          COMPRESS YES
, PART 33 USING STOGROUP ITSOASTG
                PRIQTY 187
                SECQTY 108
          PCTFREE    20
          COMPRESS YES
, PART 34 USING STOGROUP ITSOASTG
                PRIQTY 187
                SECQTY 108
          PCTFREE    20
          COMPRESS YES
, PART 35 USING STOGROUP ITSOASTG
                PRIQTY 180
                SECQTY 108
          PCTFREE    20
          COMPRESS YES
, PART 36 USING STOGROUP ITSOASTG
```

```
                PRIQTY 187
                SECQTY 108
          PCTFREE    20
          COMPRESS YES
, PART 37 USING STOGROUP ITSOASTG
                PRIQTY 158
                SECQTY 108
          PCTFREE    20
          COMPRESS YES
, PART 38 USING STOGROUP ITSOASTG
                PRIQTY 352
                SECQTY 180
          PCTFREE    20
          COMPRESS YES
, PART 39 USING STOGROUP ITSOASTG
                PRIQTY 201
                SECQTY 108
          PCTFREE    20
          COMPRESS YES
, PART 40 USING STOGROUP ITSOASTG
                PRIQTY 180
                SECQTY 108
          PCTFREE    20
          COMPRESS YES
, PART 41 USING STOGROUP ITSOASTG
                PRIQTY 180
                SECQTY 108
          PCTFREE    20
          COMPRESS YES
, PART 42 USING STOGROUP ITSOASTG
                PRIQTY 180
                SECQTY 108
          PCTFREE    20
          COMPRESS YES
, PART 43 USING STOGROUP ITSOASTG
                PRIQTY 194
                SECQTY 108
          PCTFREE    20
          COMPRESS YES
, PART 44 USING STOGROUP ITSOASTG
                PRIQTY 180
                SECQTY 108
          PCTFREE    20
          COMPRESS YES
, PART 45 USING STOGROUP ITSOASTG
                PRIQTY 324
                SECQTY 180
          PCTFREE    20
          COMPRESS YES
```

```
, PART 46 USING STOGROUP ITSOASTG
              PRIQTY 230
              SECQTY 144
        PCTFREE    20
        COMPRESS YES
, PART 47 USING STOGROUP ITSOASTG
              PRIQTY 187
              SECQTY 108
        PCTFREE    20
        COMPRESS YES
, PART 48 USING STOGROUP ITSOASTG
              PRIQTY 180
              SECQTY 108
        PCTFREE    20
        COMPRESS YES
, PART 49 USING STOGROUP ITSOASTG
              PRIQTY 180
              SECQTY 108
        PCTFREE    20
        COMPRESS YES
, PART 50 USING STOGROUP ITSOASTG
              PRIQTY 165
              SECQTY 108
        PCTFREE    20
        COMPRESS YES
, PART 51 USING STOGROUP ITSOASTG
              PRIQTY 165
              SECQTY 108
        PCTFREE    20
        COMPRESS YES
, PART 52 USING STOGROUP ITSOASTG
              PRIQTY 316
              SECQTY 180
        PCTFREE    20
        COMPRESS YES
, PART 53 USING STOGROUP ITSOASTG
              PRIQTY 223
              SECQTY 144
        PCTFREE    20
        COMPRESS YES
, PART 54 USING STOGROUP ITSOASTG
              PRIQTY 180
              SECQTY 108
        PCTFREE    20
        COMPRESS YES
, PART 55 USING STOGROUP ITSOASTG
              PRIQTY 201
              SECQTY 108
        PCTFREE    20
```

```
          COMPRESS YES
, PART 56 USING STOGROUP ITSOASTG
                PRIQTY 187
                SECQTY 108
          PCTFREE    20
          COMPRESS YES
, PART 57 USING STOGROUP ITSOASTG
                PRIQTY 208
                SECQTY 108
          PCTFREE    20
          COMPRESS YES
, PART 58 USING STOGROUP ITSOASTG
                PRIQTY 187
                SECQTY 108
          PCTFREE    20
          COMPRESS YES
, PART 59 USING STOGROUP ITSOASTG
                PRIQTY 316
                SECQTY 180
          PCTFREE    20
          COMPRESS YES
, PART 60 USING STOGROUP ITSOASTG
                PRIQTY 237
                SECQTY 144
          PCTFREE    20
          COMPRESS YES
, PART 61 USING STOGROUP ITSOASTG
                PRIQTY 187
                SECQTY 108
          PCTFREE    20
          COMPRESS YES
, PART 62 USING STOGROUP ITSOASTG
                PRIQTY 187
                SECQTY 108
          PCTFREE    20
          COMPRESS YES
, PART 63 USING STOGROUP ITSOASTG
                PRIQTY 172
                SECQTY 108
          PCTFREE    20
          COMPRESS YES
)
BUFFERPOOL BP1
LOCKSIZE   PAGE
CLOSE      NO
MAXROWS    255
CCSID      EBCDIC
                                                          ;
SET CURRENT SQLID = 'QREPADM';
```

```
CREATE  TABLESPACE ITTSTTA IN ITSTRDB1
  NUMPARTS   3
  ( PART  1 USING STOGROUP ITSOASTG
                  PRIQTY 475
                  SECQTY 252
            PCTFREE    25
            COMPRESS YES
  , PART  2 USING STOGROUP ITSOASTG
                  PRIQTY 367
                  SECQTY 216
            PCTFREE    25
            COMPRESS YES
  , PART  3 USING STOGROUP ITSOASTG
                  PRIQTY 324
                  SECQTY 180
            PCTFREE    25
            COMPRESS YES
  )
  BUFFERPOOL BP1
  LOCKSIZE   PAGE
  CLOSE      NO
  MAXROWS    255
  CCSID      EBCDIC
                                                                   ;
SET CURRENT SQLID = 'QREPADM';
CREATE  TABLESPACE ITTSIMT IN ITSTRDB2
  NUMPARTS   18
  ( PART  1 USING STOGROUP ITSOASTG
                  PRIQTY 302
                  SECQTY 180
            PCTFREE    20
            COMPRESS YES
  , PART  2 USING STOGROUP ITSOASTG
                  PRIQTY 453
                  SECQTY 252
            PCTFREE    20
            COMPRESS YES
  , PART  3 USING STOGROUP ITSOASTG
                  PRIQTY 396
                  SECQTY 216
            PCTFREE    20
            COMPRESS YES
  , PART  4 USING STOGROUP ITSOASTG
                  PRIQTY 324
                  SECQTY 180
            PCTFREE    20
            COMPRESS YES
  , PART  5 USING STOGROUP ITSOASTG
                  PRIQTY 360
```

```
                SECQTY 180
          PCTFREE    20
          COMPRESS YES
, PART  6 USING STOGROUP ITSOASTG
                PRIQTY 252
                SECQTY 144
          PCTFREE    20
          COMPRESS YES
, PART  7 USING STOGROUP ITSOASTG
                PRIQTY 295
                SECQTY 180
          PCTFREE    20
          COMPRESS YES
, PART  8 USING STOGROUP ITSOASTG
                PRIQTY 273
                SECQTY 144
          PCTFREE    20
          COMPRESS YES
, PART  9 USING STOGROUP ITSOASTG
                PRIQTY 338
                SECQTY 180
          PCTFREE    20
          COMPRESS YES
, PART 10 USING STOGROUP ITSOASTG
                PRIQTY 259
                SECQTY 144
          PCTFREE    20
          COMPRESS YES
, PART 11 USING STOGROUP ITSOASTG
                PRIQTY 280
                SECQTY 144
          PCTFREE    20
          COMPRESS YES
, PART 12 USING STOGROUP ITSOASTG
                PRIQTY 280
                SECQTY 144
          PCTFREE    20
          COMPRESS YES
, PART 13 USING STOGROUP ITSOASTG
                PRIQTY 302
                SECQTY 180
          PCTFREE    20
          COMPRESS YES
, PART 14 USING STOGROUP ITSOASTG
                PRIQTY 223
                SECQTY 144
          PCTFREE    20
          COMPRESS YES
, PART 15 USING STOGROUP ITSOASTG
```

```
                PRIQTY 302
                SECQTY 180
          PCTFREE    20
          COMPRESS YES
  , PART 16 USING STOGROUP ITSOASTG
                PRIQTY 252
                SECQTY 144
          PCTFREE    20
          COMPRESS YES
  , PART 17 USING STOGROUP ITSOASTG
                PRIQTY 309
                SECQTY 180
          PCTFREE    20
          COMPRESS YES
  , PART 18 USING STOGROUP ITSOASTG
                PRIQTY 136
                SECQTY 72
          PCTFREE    20
          COMPRESS YES
  )
  BUFFERPOOL BP1
  LOCKSIZE   PAGE
  CLOSE      NO
  MAXROWS    255
  CCSID      EBCDIC
                                                          ;
SET CURRENT SQLID = 'QREPADM';
CREATE  TABLESPACE ITTSSJE IN ITSTRDB8
  NUMPARTS   18
  ( PART  1 USING STOGROUP ITSOASTG
                PRIQTY 712
                SECQTY 360
          PCTFREE    15
          COMPRESS YES
  , PART  2 USING STOGROUP ITSOASTG
                PRIQTY 1000
                SECQTY 504
          PCTFREE    15
          COMPRESS YES
  , PART  3 USING STOGROUP ITSOASTG
                PRIQTY 900
                SECQTY 468
          PCTFREE    15
          COMPRESS YES
  , PART  4 USING STOGROUP ITSOASTG
                PRIQTY 655
                SECQTY 360
          PCTFREE    15
          COMPRESS YES
```

```
, PART  5 USING STOGROUP ITSOASTG
              PRIQTY 820
              SECQTY 432
          PCTFREE    15
          COMPRESS YES
, PART  6 USING STOGROUP ITSOASTG
              PRIQTY 525
              SECQTY 288
          PCTFREE    15
          COMPRESS YES
, PART  7 USING STOGROUP ITSOASTG
              PRIQTY 676
              SECQTY 360
          PCTFREE    15
          COMPRESS YES
, PART  8 USING STOGROUP ITSOASTG
              PRIQTY 597
              SECQTY 324
          PCTFREE    15
          COMPRESS YES
, PART  9 USING STOGROUP ITSOASTG
              PRIQTY 763
              SECQTY 396
          PCTFREE    15
          COMPRESS YES
, PART 10 USING STOGROUP ITSOASTG
              PRIQTY 561
              SECQTY 288
          PCTFREE    15
          COMPRESS YES
, PART 11 USING STOGROUP ITSOASTG
              PRIQTY 583
              SECQTY 324
          PCTFREE    15
          COMPRESS YES
, PART 12 USING STOGROUP ITSOASTG
              PRIQTY 590
              SECQTY 324
          PCTFREE    15
          COMPRESS YES
, PART 13 USING STOGROUP ITSOASTG
              PRIQTY 619
              SECQTY 324
          PCTFREE    15
          COMPRESS YES
, PART 14 USING STOGROUP ITSOASTG
              PRIQTY 460
              SECQTY 252
          PCTFREE    15
```

```
          COMPRESS YES
  , PART 15 USING STOGROUP ITSOASTG
                PRIQTY 662
                SECQTY 360
          PCTFREE    15
          COMPRESS YES
  , PART 16 USING STOGROUP ITSOASTG
                PRIQTY 554
                SECQTY 288
          PCTFREE    15
          COMPRESS YES
  , PART 17 USING STOGROUP ITSOASTG
                PRIQTY 626
                SECQTY 324
          PCTFREE    15
          COMPRESS YES
  , PART 18 USING STOGROUP ITSOASTG
                PRIQTY 324
                SECQTY 180
          PCTFREE    15
          COMPRESS YES
  )
  BUFFERPOOL BP1
  LOCKSIZE   PAGE
  CLOSE      NO
  MAXROWS    255
  CCSID      EBCDIC
                                                              ;
SET CURRENT SQLID = 'QREPADM';
CREATE LARGE TABLESPACE ITTSSJL IN ITSTRDB8
  NUMPARTS   18
  ( PART  1 USING STOGROUP ITSOASTG
                PRIQTY 2599
                SECQTY 1310
          PCTFREE    15
          COMPRESS YES
  , PART  2 USING STOGROUP ITSOASTG
                PRIQTY 4039
                SECQTY 1310
          PCTFREE    15
          COMPRESS YES
  , PART  3 USING STOGROUP ITSOASTG
                PRIQTY 3348
                SECQTY 1310
          PCTFREE    15
          COMPRESS YES
  , PART  4 USING STOGROUP ITSOASTG
                PRIQTY 2750
                SECQTY 1310
```

```
            PCTFREE    15
            COMPRESS YES
, PART  5 USING STOGROUP ITSOASTG
                  PRIQTY 3096
                  SECQTY 1310
            PCTFREE    15
            COMPRESS YES
, PART  6 USING STOGROUP ITSOASTG
                  PRIQTY 2203
                  SECQTY 1116
            PCTFREE    15
            COMPRESS YES
, PART  7 USING STOGROUP ITSOASTG
                  PRIQTY 2570
                  SECQTY 1296
            PCTFREE    15
            COMPRESS YES
, PART  8 USING STOGROUP ITSOASTG
                  PRIQTY 2469
                  SECQTY 1260
            PCTFREE    15
            COMPRESS YES
, PART  9 USING STOGROUP ITSOASTG
                  PRIQTY 2714
                  SECQTY 1310
            PCTFREE    15
            COMPRESS YES
, PART 10 USING STOGROUP ITSOASTG
                  PRIQTY 2304
                  SECQTY 1152
            PCTFREE    15
            COMPRESS YES
, PART 11 USING STOGROUP ITSOASTG
                  PRIQTY 2455
                  SECQTY 1260
            PCTFREE    15
            COMPRESS YES
, PART 12 USING STOGROUP ITSOASTG
                  PRIQTY 2390
                  SECQTY 1224
            PCTFREE    15
            COMPRESS YES
, PART 13 USING STOGROUP ITSOASTG
                  PRIQTY 2534
                  SECQTY 1296
            PCTFREE    15
            COMPRESS YES
, PART 14 USING STOGROUP ITSOASTG
                  PRIQTY 2030
```

```
                 SECQTY 1044
           PCTFREE    15
           COMPRESS YES
  , PART 15 USING STOGROUP ITSOASTG
                 PRIQTY 2606
                 SECQTY 1310
           PCTFREE    15
           COMPRESS YES
  , PART 16 USING STOGROUP ITSOASTG
                 PRIQTY 2368
                 SECQTY 1188
           PCTFREE    15
           COMPRESS YES
  , PART 17 USING STOGROUP ITSOASTG
                 PRIQTY 2541
                 SECQTY 1296
           PCTFREE    15
           COMPRESS YES
  , PART 18 USING STOGROUP ITSOASTG
                 PRIQTY 1209
                 SECQTY 612
           PCTFREE    15
           COMPRESS YES
  )
  BUFFERPOOL BP1
  LOCKSIZE   PAGE
  CLOSE      NO
  MAXROWS    255
  CCSID      EBCDIC
                                                                      ;
SET CURRENT SQLID = 'QREPADM';
CREATE  TABLESPACE ITTSRQE IN ITSCODB1
  SEGSIZE    64
  USING STOGROUP ITSOASTG
        PRIQTY 468
        SECQTY 252
  PCTFREE    10
  BUFFERPOOL BP1
  CLOSE      NO
  MAXROWS    255
  CCSID      EBCDIC
                                                                      ;
SET CURRENT SQLID = 'QREPADM';
CREATE  TABLESPACE ITTSISC IN ITSITDB1
  SEGSIZE    64
  USING STOGROUP ITSOASTG
        PRIQTY 28
        SECQTY 14
  PCTFREE    10
```

```
  BUFFERPOOL BP1
  LOCKSIZE   PAGE
  CLOSE      NO
  MAXROWS    255
  CCSID      EBCDIC
                                                                     ;

SET CURRENT SQLID = 'QREPADM';
CREATE  TABLESPACE ITTSISS IN ITSITDB1
  NUMPARTS   10
  ( PART  1 USING STOGROUP ITSOASTG
                  PRIQTY 914
                  SECQTY 468
            PCTFREE    15
            COMPRESS YES
  , PART  2 USING STOGROUP ITSOASTG
                  PRIQTY 568
                  SECQTY 288
            PCTFREE    15
            COMPRESS YES
  , PART  3 USING STOGROUP ITSOASTG
                  PRIQTY 576
                  SECQTY 288
            PCTFREE    15
            COMPRESS YES
  , PART  4 USING STOGROUP ITSOASTG
                  PRIQTY 568
                  SECQTY 288
            PCTFREE    15
            COMPRESS YES
  , PART  5 USING STOGROUP ITSOASTG
                  PRIQTY 576
                  SECQTY 288
            PCTFREE    15
            COMPRESS YES
  , PART  6 USING STOGROUP ITSOASTG
                  PRIQTY 576
                  SECQTY 288
            PCTFREE    15
            COMPRESS YES
  , PART  7 USING STOGROUP ITSOASTG
                  PRIQTY 576
                  SECQTY 288
            PCTFREE    15
            COMPRESS YES
  , PART  8 USING STOGROUP ITSOASTG
                  PRIQTY 633
                  SECQTY 324
            PCTFREE    15
            COMPRESS YES
```

```
  , PART  9 USING STOGROUP ITSOASTG
                PRIQTY 633
                SECQTY 324
          PCTFREE    15
          COMPRESS YES
  , PART 10 USING STOGROUP ITSOASTG
                PRIQTY 640
                SECQTY 324
          PCTFREE    15
          COMPRESS YES
  )
  BUFFERPOOL BP1
  LOCKSIZE   PAGE
  LOCKMAX    SYSTEM
  CLOSE      NO
  MAXROWS    255
  CCSID      EBCDIC
                                                                      ;
SET CURRENT SQLID = 'QREPADM';
CREATE  TABLESPACE ITTSMRR IN ITSITDB1
  SEGSIZE    64
  USING STOGROUP ITSOASTG
        PRIQTY 28
        SECQTY 4
  PCTFREE    10
  BUFFERPOOL BP1
  LOCKSIZE   PAGE
  CLOSE      NO
  MAXROWS    255
  CCSID      EBCDIC
                                                                      ;
SET CURRENT SQLID = 'QREPADM';
CREATE  TABLESPACE ITTSOPT IN ITSITDB1
  SEGSIZE    64
  USING STOGROUP ITSOASTG
        PRIQTY 763
        SECQTY 396
  PCTFREE    10
  COMPRESS YES
  BUFFERPOOL BP1
  LOCKSIZE   PAGE
  CLOSE      NO
  MAXROWS    255
  CCSID      EBCDIC
                                                                      ;
COMMIT WORK;

  SET CURRENT SQLID = 'QREPADM';
```

```
---======================== TABLE DEFINITIONS ========================

  CREATE TABLE ITSO.BAL
    ( ACCT_ID              INTEGER          NOT NULL
    , ACCTG_RULE_CD        CHAR(1)          NOT NULL
    , BAL_TYPE_CD          CHAR(2)          NOT NULL
    , BAL_EFF_DT           INTEGER          NOT NULL
    , BAL_AM               DECIMAL(15, 2)   NOT NULL WITH DEFAULT
    , AUDIT_INTG_CNTL_NR   SMALLINT         NOT NULL WITH DEFAULT
    , PRIMARY KEY ( ACCT_ID
                  , ACCTG_RULE_CD
                  , BAL_TYPE_CD
                  , BAL_EFF_DT
                  )
    )
    IN ITSBADB1.ITTSBLN
                                                                       ;
  CREATE TABLE ITSO.BPT_PARM_TABLE
    ( PARM_PGM_FUNC_NM     CHAR(8)          NOT NULL
    , PARAMETER_NM         CHAR(8)          NOT NULL
    , PARM_SUB_FUNC_NM     CHAR(8)          NOT NULL WITH DEFAULT
    , PARM_FIRST_EFF_TS    TIMESTAMP        NOT NULL WITH DEFAULT
    , ENVIRONMENT_ID       CHAR(8)          NOT NULL WITH DEFAULT
    , PARAMETER_LONG_NM    CHAR(30)         NOT NULL WITH DEFAULT
    , PARAMETER_VALUE_CD   CHAR(8)          NOT NULL
    , PARM_CHAR_VALUE_TX   CHAR(100)        NOT NULL WITH DEFAULT
    , PARM_DATE_VALUE_DT   DATE             NOT NULL WITH DEFAULT
                                           '0001-01-01'
    , PARM_INT_VALUE_QY    INTEGER          NOT NULL WITH DEFAULT
    , PARM_DEC_VALUE_QY    DECIMAL(15)      NOT NULL WITH DEFAULT
    , PARM_PRECISION_QY    SMALLINT         NOT NULL WITH DEFAULT
    , PARM_TS_VALUE_TS     TIMESTAMP        NOT NULL WITH DEFAULT
    '0001-01-01-00.00.00.000000'
    , PARM_CHAR_LGTH_QY    SMALLINT         NOT NULL WITH DEFAULT
    , AUDIT_USER_ID        CHAR(17)         NOT NULL WITH DEFAULT
    , PARM_LAST_EFF_TS     TIMESTAMP        NOT NULL WITH DEFAULT
    '9999-12-31-23.59.59.999999'
    , AUDIT_UPDT_TS        TIMESTAMP        NOT NULL WITH DEFAULT
    , PARM_COMMENT_TX      VARCHAR(1000)    NOT NULL WITH DEFAULT
    , PRIMARY KEY ( PARAMETER_NM
                  , PARM_PGM_FUNC_NM
                  , ENVIRONMENT_ID
                  , PARM_SUB_FUNC_NM
                  , PARM_FIRST_EFF_TS
                  )
    )
    IN ITSBADB1.ITTSBPT
                                                                       ;
  COMMENT ON TABLE ITSO.BPT_PARM_TABLE IS 'BPT V6.0 FILLER REMOVED 02/12
```

```
/99   RHC'
                                                                                    ;
  CREATE TABLE ITSO.CURR_TRADE_EXEC_PR
    ( ACCOUNT_ID            INTEGER           NOT NULL
    , ITEM_ISSUE_ID         INTEGER           NOT NULL
    , RULE_SET_SUFFIX_ID    SMALLINT          NOT NULL
    , TRAN_PROCESSING_DT    DATE              NOT NULL
    , TRAN_EXEC_PRICE_AM    DECIMAL(15, 5)    NOT NULL WITH DEFAULT
    , AUDIT_USER_ID         CHAR(17)          NOT NULL WITH DEFAULT
    , AUDIT_UPDT_TS         TIMESTAMP         NOT NULL WITH DEFAULT
    , PRIMARY KEY ( ACCOUNT_ID
                  , ITEM_ISSUE_ID
                  , RULE_SET_SUFFIX_ID
                  , TRAN_PROCESSING_DT
                  )
    )
    IN ITSTRDB1.ITTSTEP
                                                                                    ;
  CREATE TABLE ITSO.IME_TRANSACTION
    ( ACCOUNT_ID            INTEGER           NOT NULL
    , AUDIT_UPDT_TS         TIMESTAMP         NOT NULL WITH DEFAULT
    , TRAN_EFFECTIVE_DT     INTEGER           NOT NULL WITH DEFAULT
    , TRAN_SEQUENCE_ID      SMALLINT          NOT NULL WITH DEFAULT
    , BUSINESS_PROC_DT      DATE              NOT NULL WITH DEFAULT
                                             '0001-01-01'
    , ORIGINATING_ENV_CD    CHAR(2)           NOT NULL WITH DEFAULT
    , IME_OPEN_CLOSE_CD     CHAR(1)           NOT NULL WITH DEFAULT
    , IME_BUY_SELL_CD       CHAR(1)           NOT NULL WITH DEFAULT
    , IME_ACTION_CD         CHAR(8)           NOT NULL WITH DEFAULT
    , ISSUE_CLASS_CD        CHAR(8)           NOT NULL WITH DEFAULT
    , MJR_ISSUE_CLASS_CD    CHAR(8)           NOT NULL WITH DEFAULT
    , ACCOUNTING_RULE_CD    CHAR(1)           NOT NULL WITH DEFAULT
    , ITEM_ISSUE_ID         INTEGER           NOT NULL WITH DEFAULT
    , RULE_SET_SUFF_ID      SMALLINT          NOT NULL WITH DEFAULT
    , TRAN_CYCLE_ID         SMALLINT          NOT NULL WITH DEFAULT
    , TRAN_PROC_DT          INTEGER           NOT NULL WITH DEFAULT
    , TRAN_TYPE_CD          CHAR(2)           NOT NULL WITH DEFAULT
    , TRAN_SUBOR_TYPE_CD    CHAR(2)           NOT NULL WITH DEFAULT
    , TRAN_DR_CR_CD         CHAR(1)           NOT NULL WITH DEFAULT
    , TRAN_CANCEL_CD        CHAR(1)           NOT NULL WITH DEFAULT
    , TRAN_SHARE_QY         DECIMAL(15, 5)    NOT NULL WITH DEFAULT
    , TRANSACTION_AM        DECIMAL(15, 2)    NOT NULL WITH DEFAULT
    , NET_TRANSACTION_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
    , UNBOX_TRAN_SHAR_QY    DECIMAL(15, 5)    NOT NULL WITH DEFAULT
    , LIFO_UNUSED_SHR_QY    DECIMAL(15, 5)    NOT NULL WITH DEFAULT
    , TRAN_EXEC_PRICE_AM    DECIMAL(15, 5)    NOT NULL WITH DEFAULT
    , IME_CALL_EXCESS_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
    , MKT_VALUE_LONG_AM     DECIMAL(15, 2)    NOT NULL WITH DEFAULT
    , MKT_VALUE_SHORT_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
```

```
  , MARGIN_BALANCE_AM     DECIMAL(15, 2)    NOT NULL WITH DEFAULT
  , MARGIN_EQUITY_AM      DECIMAL(15, 2)    NOT NULL WITH DEFAULT
  , MKT_VALUE_SRVC_AM     DECIMAL(15, 2)    NOT NULL WITH DEFAULT
  , AUDIT_ADD_USER_ID     CHAR(17)          NOT NULL
  , AUDIT_ADD_TS          TIMESTAMP         NOT NULL WITH DEFAULT
  , AUDIT_USER_ID         CHAR(17)          NOT NULL
  , PRIMARY KEY ( ACCOUNT_ID
                , AUDIT_UPDT_TS
                )
  )
  IN ITSTRDB2.ITTSIMT
                                                                     ;
CREATE TABLE ITSO.ISSUE_CLASS
  ( ISSUE_CLASS_CD        CHAR(8)           NOT NULL
  , ISSUE_CLASS_DC        CHAR(15)          NOT NULL WITH DEFAULT
  , ISSUE_CLASS_DS        CHAR(60)          NOT NULL WITH DEFAULT
  , DFLT_SYM_TYPE_CD      CHAR(8)           NOT NULL WITH DEFAULT
  , PARNT_ISSU_CLAS_CD    CHAR(8)           NOT NULL WITH DEFAULT
  , DFLT_SETTL_DAY_CD     CHAR(8)           NOT NULL WITH DEFAULT
  , AUDIT_INTG_CNTL_NR    SMALLINT          NOT NULL WITH DEFAULT
  , SHR_FRAC_MTPLR_QY     DECIMAL(3, 2)     NOT NULL WITH DEFAULT
  , NYSE_EXT_EXEMP_IN     CHAR(1)           NOT NULL WITH DEFAULT
                                           'N'
  , CUST_FACE_CLASS_CD    CHAR(8)           NOT NULL WITH DEFAULT
  , PRIMARY KEY ( ISSUE_CLASS_CD
                )
  )
  IN ITSITDB1.ITTSISC
  DATA CAPTURE CHANGES
                                                                     ;
CREATE TABLE ITSO.ITEM_ISSUE
  ( ITEM_ISSUE_ID         INTEGER           NOT NULL
  , ITEM_ISSUE_NM         CHAR(60)          NOT NULL WITH DEFAULT
  , ISSUE_CUSIP_NR        CHAR(9)           NOT NULL WITH DEFAULT
  , ISSUE_SHRT_DESC_TX    CHAR(80)          NOT NULL WITH DEFAULT
  , WHEN_ISSUD_DESC_TX    CHAR(80)          NOT NULL WITH DEFAULT
  , SEDOL_ID              CHAR(6)           NOT NULL WITH DEFAULT
  , ISSUE_MRGN_CD         CHAR(8)           NOT NULL WITH DEFAULT
  , ISSUE_EFF_DT          INTEGER           NOT NULL WITH DEFAULT
  , BETA_FACTOR_RT        DECIMAL(15, 5)    NOT NULL WITH DEFAULT
  , ISSUE_CLASS_CD        CHAR(8)           NOT NULL WITH DEFAULT
  , ISSUER_ID             INTEGER           NOT NULL WITH DEFAULT
  , TAX_TYPE_CD           CHAR(8)           NOT NULL WITH DEFAULT
  , DFLT_ACCS_SYM_NM      CHAR(15)          NOT NULL WITH DEFAULT
  , ISSUE_VOL_LEVEL_CD    CHAR(2)           NOT NULL WITH DEFAULT
  , SOLCT_ORDER_CD        CHAR(1)           NOT NULL WITH DEFAULT
  , CURR_PROSP_DT         INTEGER           NOT NULL WITH DEFAULT
  , LAST_PROSP_MAIL_DT    INTEGER           NOT NULL WITH DEFAULT
  , ISSUE_PROD_CD         CHAR(4)           NOT NULL WITH DEFAULT
```

```
, DTC_ELIGIBILITY_IN    CHAR(1)           NOT NULL WITH DEFAULT
, DTC_STATUS_CD         CHAR(2)           NOT NULL WITH DEFAULT
, REORGANIZATION_CD     CHAR(1)           NOT NULL WITH DEFAULT
, MJR_ISSUE_CLASS_CD    CHAR(8)           NOT NULL WITH DEFAULT
, FINANCIAL_RPT_CD      CHAR(4)           NOT NULL WITH DEFAULT
, INTRN_ORD_ROUTE_ID    CHAR(8)           NOT NULL WITH DEFAULT
, SETTLEMENT_DAY_CD     CHAR(8)           NOT NULL WITH DEFAULT
, CNTRY_ISSUANCE_CD     CHAR(2)           NOT NULL WITH DEFAULT
, REORG_IMPACT_TX       CHAR(60)          NOT NULL WITH DEFAULT
, DTC_C_O_D_CHILL_IN    CHAR(1)           NOT NULL WITH DEFAULT
, DTC_DEP_CHILL_IN      CHAR(1)           NOT NULL WITH DEFAULT
, DTC_D_ORD_CHILL_IN    CHAR(1)           NOT NULL WITH DEFAULT
, DTC_PLDG_CHILL_IN     CHAR(1)           NOT NULL WITH DEFAULT
, DTC_SEG_CHILL_IN      CHAR(1)           NOT NULL WITH DEFAULT
, DTC_W_F_T_CHILL_IN    CHAR(1)           NOT NULL WITH DEFAULT
, DTC_DEP_CHILL_DT      INTEGER           NOT NULL WITH DEFAULT
, DIVIDEND_RNVST_IN     CHAR(1)           NOT NULL WITH DEFAULT
, WHEN_ISSUE_FRST_DT    INTEGER           NOT NULL WITH DEFAULT
, WHEN_ISSUE_LAST_DT    INTEGER           NOT NULL WITH DEFAULT
, UNDR_ITEM_ISSUE_ID    INTEGER           NOT NULL WITH DEFAULT
, TRADE_FLOOR_SYM_TX    CHAR(20)          NOT NULL WITH DEFAULT
, MANUAL_TRANSFER_IN    CHAR(1)           NOT NULL WITH DEFAULT
, ISSUE_CONV_RT         DECIMAL(10, 5)    NOT NULL WITH DEFAULT
, SPCL_REDEMPTION_IN    CHAR(1)           NOT NULL WITH DEFAULT
, SINKING_FUND_IN       CHAR(1)           NOT NULL WITH DEFAULT
, NSCC_ELIG_IN          CHAR(1)           NOT NULL WITH DEFAULT
, ROUND_LOT_QY          SMALLINT          NOT NULL WITH DEFAULT
, FAST_MARKET_IN        CHAR(1)           NOT NULL WITH DEFAULT
, GSCC_ELIG_IN          CHAR(1)           NOT NULL WITH DEFAULT
, IRS_TAX_RT            DECIMAL(5, 5)     NOT NULL WITH DEFAULT
, DTC_FED_SETTL_CD      CHAR(1)           NOT NULL WITH DEFAULT
, IRS_TAX_STATUS_CD     CHAR(1)           NOT NULL WITH DEFAULT
, ISSUE_EXTINCT_DT      INTEGER           NOT NULL WITH DEFAULT
, NONSTD_ASST_FEE_IN    CHAR(1)           NOT NULL WITH DEFAULT
, ORDR_ENTRY_METH_CD    CHAR(4)           NOT NULL WITH DEFAULT
, AUDIT_INTG_CNTL_NR    SMALLINT          NOT NULL WITH DEFAULT
, LAST_WS_POSTN_QY      INTEGER           NOT NULL WITH DEFAULT
, LAST_WS_POS_QY_DT     DATE              NOT NULL WITH DEFAULT
                                         '0001-01-01'
, AUTO_BOAT_REMND_IN    CHAR(1)           NOT NULL WITH DEFAULT
, SPCL_PACKAGING_CD     CHAR(1)           NOT NULL WITH DEFAULT
, DTC_DRS_ELIG_IN       CHAR(1)           NOT NULL WITH DEFAULT
, SEC_FEE_ELIG_IN       CHAR(1)           NOT NULL WITH DEFAULT
, AUDIT_UPDT_TS         TIMESTAMP         NOT NULL WITH DEFAULT
, AUDIT_USER_ID         CHAR(17)          NOT NULL WITH DEFAULT
, AUDIT_ADD_TS          TIMESTAMP         NOT NULL WITH DEFAULT
, AUDIT_ADD_USER_ID     CHAR(17)          NOT NULL WITH DEFAULT
, ISS_COMPOSITION_CD    CHAR(1)           NOT NULL WITH DEFAULT
, TEST_ISSUE_IN         CHAR(1)           NOT NULL WITH DEFAULT
```

```
  , BOOK_ENTRY_ONLY_IN    CHAR(1)          NOT NULL WITH DEFAULT
                                          'N'
  , FRGN_CUSTODY_IN       CHAR(1)          NOT NULL WITH DEFAULT
                                          'N'
  , ITEM_ISSUE_TYPE_CD    CHAR(8)          NOT NULL WITH DEFAULT
  , SPCL_EXECUTION_IN     CHAR(1)          NOT NULL WITH DEFAULT
                                          'N'
  , MARKET_CATEGORY_CD    CHAR(8)          NOT NULL WITH DEFAULT
  , MANUAL_PRICING_IN     CHAR(1)          NOT NULL WITH DEFAULT
                                          'N'
  , PROSP_URL_AVAL_IN     CHAR(1)          NOT NULL WITH DEFAULT
                                          'N'
  , MIN_PRICE_VARTN_AM    DECIMAL(12, 6)   NOT NULL WITH DEFAULT
  , AFTER_HOURS_MPV_AM    DECIMAL(12, 6)   NOT NULL WITH DEFAULT
  , PRICE_REP_TYPE_CD     CHAR(1)          NOT NULL WITH DEFAULT
  , PRIM_EXCH_QUOTE_ID    CHAR(1)          NOT NULL WITH DEFAULT
  , ITEM_FOOTNOTE_CD      CHAR(4)          NOT NULL WITH DEFAULT
  , CUST_FACE_CLASS_CD    CHAR(8)          NOT NULL WITH DEFAULT
  , ROUND_LOT_IN          CHAR(1)          NOT NULL WITH DEFAULT
  , SHR_FRAC_MTPLR_QY     DECIMAL(3, 2)    NOT NULL WITH DEFAULT
  , SEC_ACTIVE_IN         CHAR(1)          NOT NULL WITH DEFAULT
  , SPCL_MRGN_REQ_PC      DECIMAL(4, 2)    NOT NULL WITH DEFAULT
  , SWB_ASSET_CLASS_CD    CHAR(2)          NOT NULL WITH DEFAULT
  , PRIMARY KEY ( ITEM_ISSUE_ID
              )
  )
  IN ITSITDB1.ITTSISS
  DATA CAPTURE CHANGES
                                                                   ;
CREATE TABLE ITSO.LEGACY_FUND_TRAN
  ( ACCT_ID               INTEGER          NOT NULL
  , TRAN_CYCLE_ID         SMALLINT         NOT NULL
  , TRAN_EFF_DT           INTEGER          NOT NULL
  , TRAN_KEY_TM           SMALLINT         NOT NULL
  , M_SRCE_CD             CHAR(5)          NOT NULL WITH DEFAULT
  , LGCY_CXL_REAS_CD      CHAR(1)          NOT NULL WITH DEFAULT
  , M_SECURITY_NR         INTEGER          NOT NULL WITH DEFAULT
  , M_PAY_SEQ_NBR_ID      DECIMAL(5)       NOT NULL WITH DEFAULT
  , M_CNTRLNO             INTEGER          NOT NULL WITH DEFAULT
  , LGCY_MEMO_POST_IN     CHAR(1)          NOT NULL WITH DEFAULT
  , M_TRAN_DESC_1_TX      CHAR(24)         NOT NULL WITH DEFAULT
  , M_TRAN_DESC_2_TX      CHAR(24)         NOT NULL WITH DEFAULT
  , M_BNKNO               DECIMAL(5)       NOT NULL WITH DEFAULT
  , M_CWHOM               CHAR(5)          NOT NULL WITH DEFAULT
  , AUDIT_INTG_CNTL_NR    SMALLINT         NOT NULL WITH DEFAULT
  , PRIMARY KEY ( ACCT_ID
              , TRAN_CYCLE_ID
              , TRAN_EFF_DT
              , TRAN_KEY_TM
```

```
              )
  )
  IN ITSTRDB1.ITTSLFC
                                                                       ;
CREATE TABLE ITSO.LEGACY_ISSUE_TRAN
  ( ACCT_ID                INTEGER          NOT NULL
  , TRAN_CYCLE_ID          SMALLINT         NOT NULL
  , TRAN_EFF_DT            INTEGER          NOT NULL
  , TRAN_KEY_TM            SMALLINT         NOT NULL
  , M_SRCE_CD              CHAR(5)          NOT NULL WITH DEFAULT
  , LGCY_CXL_REAS_CD       CHAR(1)          NOT NULL WITH DEFAULT
  , M_SECURITY_NR          INTEGER          NOT NULL WITH DEFAULT
  , M_CNTRLNO              INTEGER          NOT NULL WITH DEFAULT
  , AUDIT_INTG_CNTL_NR     SMALLINT         NOT NULL WITH DEFAULT
  , PRIMARY KEY ( ACCT_ID
              , TRAN_CYCLE_ID
              , TRAN_EFF_DT
              , TRAN_KEY_TM
              )
  )
  IN ITSTRDB1.ITTSLIC
                                                                       ;
CREATE TABLE ITSO.LEGACY_TRADE_TRAN
  ( ACCT_ID                INTEGER          NOT NULL
  , TRAN_CYCLE_ID          SMALLINT         NOT NULL
  , TRAN_EFF_DT            INTEGER          NOT NULL
  , TRAN_KEY_TM            SMALLINT         NOT NULL
  , M_SRCE_CD              CHAR(5)          NOT NULL WITH DEFAULT
  , LGCY_CXL_REAS_CD       CHAR(1)          NOT NULL WITH DEFAULT
  , M_SECURITY_NR          INTEGER          NOT NULL WITH DEFAULT
  , M_SEQ_NBR              INTEGER          NOT NULL WITH DEFAULT
  , M_ORF_OR_ORDRNO        DECIMAL(8)       NOT NULL WITH DEFAULT
  , M_BRKR_CPCTY_CD        CHAR(1)          NOT NULL WITH DEFAULT
  , M_ORDER_TAKER_ID       CHAR(4)          NOT NULL WITH DEFAULT
  , M_EXCH_CLASS_CD        SMALLINT         NOT NULL WITH DEFAULT
  , LGCY_MEMO_POST_IN      CHAR(1)          NOT NULL WITH DEFAULT
  , M_SPCL_COMM_TBL        CHAR(4)          NOT NULL WITH DEFAULT
  , M_PROMO_CD             CHAR(5)          NOT NULL WITH DEFAULT
  , M_PROMO_COM_SAV_AM     DECIMAL(7, 2)    NOT NULL WITH DEFAULT
  , M_PROMO_SAV_MAX_IN     CHAR(1)          NOT NULL WITH DEFAULT
  , AUDIT_INTG_CNTL_NR     SMALLINT         NOT NULL WITH DEFAULT
  , COMM_CALC_CD           CHAR(2)          NOT NULL WITH DEFAULT
  , PRIMARY KEY ( ACCT_ID
              , TRAN_CYCLE_ID
              , TRAN_EFF_DT
              , TRAN_KEY_TM
              )
  )
  IN ITSTRDB1.ITTSLTR
```

```
                                                                               ;
CREATE TABLE ITSO.MEMO_TRAN_LIST
  ( ACCT_ID               INTEGER            NOT NULL
  , TRAN_CYCLE_ID         SMALLINT           NOT NULL
  , TRAN_EFF_DT           INTEGER            NOT NULL
  , TRAN_KEY_TM           SMALLINT           NOT NULL
  , TRAN_PROC_DT          INTEGER            NOT NULL WITH DEFAULT
  , AUDIT_INTG_CNTL_NR    SMALLINT           NOT NULL WITH DEFAULT
  , AUDIT_UPDT_USER_ID    CHAR(8)            NOT NULL WITH DEFAULT
  , TRAN_PUBLCN_TS        TIMESTAMP          NOT NULL WITH DEFAULT
  )
  IN ITSTRDB1.ITTSMTL
                                                                               ;
CREATE TABLE ITSO.MRGN_REQ_RULE
  ( ISSUE_CLASS_CD        CHAR(8)            NOT NULL
  , EQTY_REQ_TYPE_CD      CHAR(2)            NOT NULL
  , EQTY_ACCTG_TYPE_CD    CHAR(4)            NOT NULL
  , RANGE_BKPT_QY         DECIMAL(5, 2)      NOT NULL
  , RANGE_BKPT_UNIT_CD    CHAR(1)            NOT NULL WITH DEFAULT
  , BEGIN_EFF_DT          DATE               NOT NULL WITH DEFAULT
  , STOP_EFF_DT           DATE               NOT NULL WITH DEFAULT
  , EQTY_REQ_RT           DECIMAL(6, 2)      NOT NULL WITH DEFAULT
  , EQTY_RATE_UNIT_CD     CHAR(4)            NOT NULL WITH DEFAULT
  , EQTY_REQ_ADDTL_RT     DECIMAL(6, 2)      NOT NULL WITH DEFAULT
  , EQTY_ADDTL_UNIT_CD    CHAR(4)            NOT NULL WITH DEFAULT
  , EQTY_REQ_CMPR_CD      CHAR(2)            NOT NULL WITH DEFAULT
  , AUDIT_USER_ID         CHAR(17)           NOT NULL WITH DEFAULT
  , AUDIT_UPDT_TS         TIMESTAMP          NOT NULL WITH DEFAULT
  , SPCL_MAINT_IN         CHAR(1)            NOT NULL WITH DEFAULT
  , PRIMARY KEY ( ISSUE_CLASS_CD
              , EQTY_REQ_TYPE_CD
              , EQTY_ACCTG_TYPE_CD
              , RANGE_BKPT_QY
              )
  )
  IN ITSITDB1.ITTSMRR
  DATA CAPTURE CHANGES
                                                                               ;
CREATE TABLE ITSO.OPEN_DAILY_BAL_1
  ( ACCT_ID               INTEGER            NOT NULL
  , BAL_CLOSING_EFF_DT    DATE               NOT NULL
  , B001_ACCT_NT_WR_AM    DECIMAL(15, 2)     NOT NULL WITH DEFAULT
  , B007_MMF_HIER_AM      DECIMAL(15, 2)     NOT NULL WITH DEFAULT
  , B010_HOLD_FUNDS_IN    CHAR(1)            NOT NULL WITH DEFAULT
                                            'N'
  , B012_AVAL_T1_AM       DECIMAL(15, 2)     NOT NULL WITH DEFAULT
  , B013_STD_HSE_EX_AM    DECIMAL(15, 2)     NOT NULL WITH DEFAULT
  , B014_OPT_MIN_EX_AM    DECIMAL(15, 2)     NOT NULL WITH DEFAULT
  , B015_INIT_MN_EX_AM    DECIMAL(15, 2)     NOT NULL WITH DEFAULT
```

```
, B016_NON_NEG_MV_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B017_PENNY_STCK_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B018_NET_MARGIN_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B019_NT_MRGN_ST_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B020_SEC_DUE_IN       CHAR(1)           NOT NULL WITH DEFAULT
                                         'N'
, B021_SW_MNT_REQ_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B022_SWB_VA_FND_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B025_MRGN_ACCT_AM     DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B026_AVAL_CASH_AM     DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B028_CASH_ACCT_AM     DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B030_AVAL_MRGN_AM     DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B036_HOLD_AGNST_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B038_CASH_MMF_AM      DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B040_AVAL_2_BRW_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B042_TOT_AVL_PY_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B044_S2AVL_2BRW_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B045_S2AVL_2PAY_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B046_UNCLRD_FND_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B048_MAN_HOLD_AM      DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B050_UNCLRD_SLS_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B052_MARK_2_MKT_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B054_NO_TRD_MVS_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B056_CASH_HOLDT_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B058_CASH_DEBIT_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B060_CASH_DUE_AM      DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B064_TOT_FED_CL_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B066_HSE_MAINT_AM     DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B068_OPTMIN_MNT_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B070_INTMIN_CLL_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B072_MRG_CLL_DU_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B074_MONEY_TDUE_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B077_MF_AVL2TRD_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B078_MUTL_FUND_AM     DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B079_MF_NO_MRGN_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B080_FED_CL_YDY_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B087_MRG_AV2TRD_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B090_NEWINT_CLL_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B092_NEWINT_EOD_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B093_AV_2TRD_CS_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B095_EQ_GT_10SH_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B097_NON_MRGN_AM      DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B099_TREASURIES_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B101_GOVT_AGNCY_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B103_MUNI_BOND_AM     DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B105_NONCVT_CRP_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B107_CONV_CORP_AM     DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B109_LONG_OPTN_AM     DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B111_SHORT_OPTN_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
```

```
, B113_MRG_SEC_IN_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B116_MNYMKT_FND_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B118_SHRTBAL_ST_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B120_CASH_BAL_AM      DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B123_MARGIN_BAL_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B125_SHORT_BAL_AM     DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B128_TLACCT_BAL_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B131_MRGN_EQTY_AM     DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B133_EQUITY_PC        DECIMAL(3, 2)     NOT NULL WITH DEFAULT
, B135_EQ_INC_OPT_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B137_SMAFED_CLL_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B139_SWB_EXCMNT_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B141_NYSE_EXCLL_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B143_TOTACCT_ST_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B145_SW_OPT_MRG_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B147_TLACCT_EOD_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B148_NEW_FD_EOD_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B151_SHTBAL_EOD_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B152_MVL_CASH_AM      DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B154_OPT_MVS_CA_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B156_OPT_MVL_CA_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B158_MVS_CASH_AM      DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B160_MVL_MARGIN_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B162_MVS_MARGIN_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B164_OPT_MVL_MR_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B166_OPT_MVS_MR_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B168_TLMV_CA_MR_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B170_SWOPT_RQCH_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B171_PEG_OPT_MV_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B176_BAL_SUBJMI_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B182_AR_BALANCE_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B184_MO_END_P_O_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B187_VISA_AUTH_AM     DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B188_MRGMTD_INT_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B189_MRGDAY_INT_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B194_INT_MIN_CR_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B197_INMNY_CALL_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B199_SWOPT_MNRQ_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B215_CSHBAL_EOD_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B217_CSH_BAL_ST_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B218_MRGNBAL_ST_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B219_MRGBAL_EOD_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B220_MMF_STMT_AM      DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B221_MMF_EOD_ST_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B223_SW_MNT_REQ_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B225_NY_MNT_REQ_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B231_SPCL_AC_RQ_PC    DECIMAL(3, 2)     NOT NULL WITH DEFAULT
, B238_INMNY_PUTS_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, AUDIT_UPDT_TS         TIMESTAMP         NOT NULL WITH DEFAULT
```

```
    , AUDIT_USER_ID          CHAR(17)          NOT NULL WITH DEFAULT
    , B150_UNCSLS_EOD_AM     DECIMAL(15, 2)    NOT NULL WITH DEFAULT
    , B005_DAYTRD_BUYPWR     DECIMAL(15, 2)    NOT NULL WITH DEFAULT
    , B006_BROKER_BAL_AM     DECIMAL(15, 2)    NOT NULL WITH DEFAULT
    , B011_BRK_ST_EOD_AM     DECIMAL(15, 2)    NOT NULL WITH DEFAULT
    , B034_BANK_BAL_AM       DECIMAL(15, 2)    NOT NULL WITH DEFAULT
    , B062_BNK_ST_EOD_AM     DECIMAL(15, 2)    NOT NULL WITH DEFAULT
    , PRIMARY KEY ( ACCT_ID
                  , BAL_CLOSING_EFF_DT
                  )
    )
    IN ITSBADB1.ITTSOB1
                                                                     ;
  COMMENT ON TABLE ITSO.OPEN_DAILY_BAL_1 IS 'THE VALUATION OF AN ACCOUNT
 AT THE BEGINING OF A DAY BASED ON THE      PRIOR DAYS PRICES.'
                                                                     ;
  COMMENT ON ITSO.OPEN_DAILY_BAL_1
    ( ACCT_ID IS 'THE UNIQUE IDENTIFIER OF AN ACCOUNT.'
    , B001_ACCT_NT_WR_AM IS 'THE TOTAL VALUE OF ALL ASSETS ASSOCIATED WI
TH THE ACCOUNT'

    , AUDIT_USER_ID IS 'AUDIT USER IDENTIFIER -- ID OF THE ONLINE USER O
R THE BATCH JOB ID   AND STEP NAM E.  (DB2 FORMAT = CHAR 17)'
    )
                                                                     ;
  CREATE TABLE ITSO.OPEN_DAILY_BAL_2
    ( ACCT_ID                INTEGER           NOT NULL
    , BAL_CLOSING_EFF_DT     DATE              NOT NULL
    , B001_ACCT_NT_WR_AM     DECIMAL(15, 2)    NOT NULL WITH DEFAULT
    , B007_MMF_HIER_AM       DECIMAL(15, 2)    NOT NULL WITH DEFAULT
    , B010_HOLD_FUNDS_IN     CHAR(1)           NOT NULL WITH DEFAULT
                                              'N'
    , B012_AVAL_T1_AM        DECIMAL(15, 2)    NOT NULL WITH DEFAULT
    , B013_STD_HSE_EX_AM     DECIMAL(15, 2)    NOT NULL WITH DEFAULT
    , B014_OPT_MIN_EX_AM     DECIMAL(15, 2)    NOT NULL WITH DEFAULT
    , B015_INIT_MN_EX_AM     DECIMAL(15, 2)    NOT NULL WITH DEFAULT
    , B016_NON_NEG_MV_AM     DECIMAL(15, 2)    NOT NULL WITH DEFAULT
    , B017_PENNY_STCK_AM     DECIMAL(15, 2)    NOT NULL WITH DEFAULT
    , B018_NET_MARGIN_AM     DECIMAL(15, 2)    NOT NULL WITH DEFAULT
    , B019_NT_MRGN_ST_AM     DECIMAL(15, 2)    NOT NULL WITH DEFAULT
    , B020_SEC_DUE_IN        CHAR(1)           NOT NULL WITH DEFAULT
                                              'N'
    , B021_SW_MNT_REQ_AM     DECIMAL(15, 2)    NOT NULL WITH DEFAULT
    , B022_SWB_VA_FND_AM     DECIMAL(15, 2)    NOT NULL WITH DEFAULT
    , B025_MRGN_ACCT_AM      DECIMAL(15, 2)    NOT NULL WITH DEFAULT
    , B026_AVAL_CASH_AM      DECIMAL(15, 2)    NOT NULL WITH DEFAULT
    , B028_CASH_ACCT_AM      DECIMAL(15, 2)    NOT NULL WITH DEFAULT
    , B030_AVAL_MRGN_AM      DECIMAL(15, 2)    NOT NULL WITH DEFAULT
    , B036_HOLD_AGNST_AM     DECIMAL(15, 2)    NOT NULL WITH DEFAULT
```

```
, B038_CASH_MMF_AM      DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B040_AVAL_2_BRW_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B042_TOT_AVL_PY_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B044_S2AVL_2BRW_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B045_S2AVL_2PAY_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B046_UNCLRD_FND_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B048_MAN_HOLD_AM      DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B050_UNCLRD_SLS_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B052_MARK_2_MKT_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B054_NO_TRD_MVS_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B056_CASH_HOLDT_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B058_CASH_DEBIT_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B060_CASH_DUE_AM      DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B064_TOT_FED_CL_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B066_HSE_MAINT_AM     DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B068_OPTMIN_MNT_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B070_INTMIN_CLL_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B072_MRG_CLL_DU_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B074_MONEY_TDUE_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B077_MF_AVL2TRD_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B078_MUTL_FUND_AM     DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B079_MF_NO_MRGN_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B080_FED_CL_YDY_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B087_MRG_AV2TRD_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B090_NEWINT_CLL_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B092_NEWINT_EOD_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B093_AV_2TRD_CS_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B095_EQ_GT_10SH_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B097_NON_MRGN_AM      DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B099_TREASURIES_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B101_GOVT_AGNCY_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B103_MUNI_BOND_AM     DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B105_NONCVT_CRP_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B107_CONV_CORP_AM     DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B109_LONG_OPTN_AM     DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B111_SHORT_OPTN_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B113_MRG_SEC_IN_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B116_MNYMKT_FND_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B118_SHRTBAL_ST_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B120_CASH_BAL_AM      DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B123_MARGIN_BAL_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B125_SHORT_BAL_AM     DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B128_TLACCT_BAL_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B131_MRGN_EQTY_AM     DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B133_EQUITY_PC        DECIMAL(3, 2)     NOT NULL WITH DEFAULT
, B135_EQ_INC_OPT_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B137_SMAFED_CLL_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B139_SWB_EXCMNT_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B141_NYSE_EXCLL_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
```

```
, B143_TOTACCT_ST_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B145_SW_OPT_MRG_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B147_TLACCT_EOD_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B148_NEW_FD_EOD_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B151_SHTBAL_EOD_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B152_MVL_CASH_AM      DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B154_OPT_MVS_CA_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B156_OPT_MVL_CA_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B158_MVS_CASH_AM      DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B160_MVL_MARGIN_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B162_MVS_MARGIN_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B164_OPT_MVL_MR_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B166_OPT_MVS_MR_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B168_TLMV_CA_MR_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B170_SWOPT_RQCH_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B171_PEG_OPT_MV_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B176_BAL_SUBJMI_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B182_AR_BALANCE_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B184_MO_END_P_O_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B187_VISA_AUTH_AM     DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B188_MRGMTD_INT_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B189_MRGDAY_INT_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B194_INT_MIN_CR_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B197_INMNY_CALL_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B199_SWOPT_MNRQ_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B215_CSHBAL_EOD_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B217_CSH_BAL_ST_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B218_MRGNBAL_ST_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B219_MRGBAL_EOD_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B220_MMF_STMT_AM      DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B221_MMF_EOD_ST_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B223_SW_MNT_REQ_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B225_NY_MNT_REQ_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B231_SPCL_AC_RQ_PC    DECIMAL(3, 2)     NOT NULL WITH DEFAULT
, B238_INMNY_PUTS_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, AUDIT_UPDT_TS         TIMESTAMP         NOT NULL WITH DEFAULT
, AUDIT_USER_ID         CHAR(17)          NOT NULL WITH DEFAULT
, B150_UNCSLS_EOD_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B005_DAYTRD_BUYPWR    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B006_BROKER_BAL_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B011_BRK_ST_EOD_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B034_BANK_BAL_AM      DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, B062_BNK_ST_EOD_AM    DECIMAL(15, 2)    NOT NULL WITH DEFAULT
, PRIMARY KEY ( ACCT_ID
              , BAL_CLOSING_EFF_DT
              )
)
IN ITSBADB2.ITTSOB1
                                                                     ;
```

```
  COMMENT ON TABLE ITSO.OPEN_DAILY_BAL_2 IS 'THE VALUATION OF AN ACCOUNT
 AT THE BEGINING OF A DAY BASED ON THE      PRIOR DAYS PRICES.'
                                                                      ;
  COMMENT ON ITSO.OPEN_DAILY_BAL_2
    ( ACCT_ID IS 'THE UNIQUE IDENTIFIER OF AN ACCOUNT.'
    , B001_ACCT_NT_WR_AM IS 'THE TOTAL VALUE OF ALL ASSETS ASSOCIATED WI
TH THE ACCOUNT'

    , AUDIT_USER_ID IS 'AUDIT USER IDENTIFIER -- ID OF THE ONLINE USER O
R THE BATCH JOB ID   AND STEP NAM E.  (DB2 FORMAT = CHAR 17)'
    )
                                                                      ;
  CREATE TABLE ITSO.OPTIONS
    ( ITEM_ISSUE_ID         INTEGER          NOT NULL
    , USER_CUSIP_ID         CHAR(9)          NOT NULL WITH DEFAULT
    , STRIKE_PRICE_AM       DECIMAL(14, 5)   NOT NULL WITH DEFAULT
    , OPTION_EXPR_DT        INTEGER          NOT NULL WITH DEFAULT
    , UNDR_ITEM_ISSUE_ID    INTEGER          NOT NULL WITH DEFAULT
    , PRIC_USER_CUSIP_ID    CHAR(9)          NOT NULL WITH DEFAULT
    , UNIT_OF_TRADE_CD      CHAR(8)          NOT NULL WITH DEFAULT
    , STRK_PRICE_ADJ_RT     DECIMAL(8, 5)    NOT NULL WITH DEFAULT
    , OPTION_ACTIVATN_DT    INTEGER          NOT NULL WITH DEFAULT
    , OPTN_INACTIVATN_DT    INTEGER          NOT NULL WITH DEFAULT
    , OPTN_CALL_PUT_CD      CHAR(1)          NOT NULL WITH DEFAULT
    , ADJUSTED_OPTION_IN    CHAR(1)          NOT NULL WITH DEFAULT
    , UNIT_OF_TRADE_QY      SMALLINT         NOT NULL WITH DEFAULT
    , PRIOR_OCC_SYM_NM      CHAR(6)          NOT NULL WITH DEFAULT
    , ADJ_UNIT_TRADE_QY     SMALLINT         NOT NULL WITH DEFAULT
    , MARKET_INDEX_ID       SMALLINT         NOT NULL WITH DEFAULT
    , AUDIT_INTG_CNTL_NR    SMALLINT         NOT NULL WITH DEFAULT
    , MAX_CUST_POSTN_QY     DECIMAL(12)      NOT NULL WITH DEFAULT
    , OPTION_ADJ_TYPE_CD    CHAR(6)          NOT NULL WITH DEFAULT
    , LAST_CHG_BUSN_DT      DATE             NOT NULL WITH DEFAULT
    , OPTION_SERIES_ID      INTEGER          NOT NULL WITH DEFAULT
    , LAST_TRADE_DT         DATE             NOT NULL WITH DEFAULT
                                            '0001-01-01'
    , PRIMARY KEY ( ITEM_ISSUE_ID
                  )
    )
    IN ITSITDB1.ITTSOPT
    DATA CAPTURE CHANGES
                                                                      ;
  CREATE TABLE ITSO.POSTN
    ( ACCT_ID               INTEGER          NOT NULL
    , ACCTG_RULE_CD         CHAR(1)          NOT NULL
    , ITEM_ISSUE_ID         INTEGER          NOT NULL
    , RS_SUFF_ID            SMALLINT         NOT NULL
    , CONTROL_CD            CHAR(1)          NOT NULL
    , POSTN_TYPE_CD         CHAR(2)          NOT NULL
```

```
  , POSTN_EFF_DT          INTEGER           NOT NULL
  , POSTN_SHR_QY          DECIMAL(15, 5)    NOT NULL WITH DEFAULT
  , AUDIT_INTG_CNTL_NR    SMALLINT          NOT NULL WITH DEFAULT
  , PRIMARY KEY ( ACCT_ID
                , ACCTG_RULE_CD
                , ITEM_ISSUE_ID
                , RS_SUFF_ID
                , CONTROL_CD
                , POSTN_TYPE_CD
                , POSTN_EFF_DT
                )
  )
  IN ITSBADB1.ITTSPOS
                                                                      ;
CREATE TABLE ITSO.RULE_QLF_VALUE
  ( CMPNT_ID              INTEGER           NOT NULL
  , PRI_RULE_ID           INTEGER           NOT NULL
  , PRI_QLF_ID            INTEGER           NOT NULL
  , ATTRB_ID              INTEGER           NOT NULL
  , QLF_ROW_ID            INTEGER           NOT NULL
  , ATTRB_VALU_OPR_CD     CHAR(2)           NOT NULL WITH DEFAULT
  , VALUE_BEG_DT          DATE              NOT NULL WITH DEFAULT
  , VALUE_END_DT          DATE              NOT NULL WITH DEFAULT
  , ATTRB_DATA_TYPE_TX    CHAR(9)           NOT NULL WITH DEFAULT
  , SML_INTEGER_QY        SMALLINT          NOT NULL WITH DEFAULT
  , INTEGER_QY            INTEGER           NOT NULL WITH DEFAULT
  , DEC_QY                DECIMAL(17, 6)    NOT NULL WITH DEFAULT
  , QLF_TM                TIME              NOT NULL WITH DEFAULT
  , QLF_DT                DATE              NOT NULL WITH DEFAULT
  , CHAR_TX               CHAR(75)          NOT NULL WITH DEFAULT
  , COMM_CALC_IN          CHAR(1)           NOT NULL WITH DEFAULT
  , DESKTOP_IN            CHAR(1)           NOT NULL WITH DEFAULT
  , AUDIT_INTG_CNTL_NR    SMALLINT          NOT NULL WITH DEFAULT
  , AUDIT_PWS_ID          CHAR(16)          NOT NULL WITH DEFAULT
  , AUDIT_UPDT_TS         TIMESTAMP         NOT NULL WITH DEFAULT
  , AUDIT_USER_ID         CHAR(17)          NOT NULL WITH DEFAULT
  , PRIMARY KEY ( CMPNT_ID
                , PRI_RULE_ID
                , PRI_QLF_ID
                , ATTRB_ID
                , QLF_ROW_ID
                , COMM_CALC_IN
                , ATTRB_VALU_OPR_CD
                )
  )
  IN ITSCODB1.ITTSRQE
  DATA CAPTURE CHANGES
                                                                      ;
CREATE TABLE ITSO.SMA_BASE_JRNL
```

```
( ACCT_ID                INTEGER          NOT NULL
, AUDIT_ADD_TS           TIMESTAMP        NOT NULL
, SMA_ADJ_AM             DECIMAL(15, 2)   NOT NULL WITH DEFAULT
, SMA_BAL_AM             DECIMAL(15, 2)   NOT NULL WITH DEFAULT
, SMA_BAL_DT             DATE             NOT NULL WITH DEFAULT
, SMA_ADJ_PC             DECIMAL(3, 2)    NOT NULL WITH DEFAULT
, SMA_ACTN_CD            CHAR(7)          NOT NULL WITH DEFAULT
, BOX_ACTN_CD            CHAR(1)          NOT NULL WITH DEFAULT
, SMA_CALC_TRAN_AM       DECIMAL(15, 2)   NOT NULL WITH DEFAULT
, SMA_T2_SHARE_QY        DECIMAL(15, 5)   NOT NULL WITH DEFAULT
, SMA_T6_SHARE_QY        DECIMAL(15, 5)   NOT NULL WITH DEFAULT
, TRAN_CYCLE_ID          SMALLINT         NOT NULL WITH DEFAULT
, TRAN_EFF_DT            DATE             NOT NULL WITH DEFAULT
, TRAN_SEQ_ID            SMALLINT         NOT NULL WITH DEFAULT
, TRAN_TYPE_CD           CHAR(2)          NOT NULL WITH DEFAULT
, TRAN_SUBOR_TYPE_CD     CHAR(2)          NOT NULL WITH DEFAULT
, TRAN_DR_CR_CD          CHAR(1)          NOT NULL WITH DEFAULT
, TRAN_CXL_CD            CHAR(1)          NOT NULL WITH DEFAULT
, ACCTG_RULE_CD          CHAR(1)          NOT NULL WITH DEFAULT
, ITEM_ISSUE_ID          INTEGER          NOT NULL WITH DEFAULT
, TRAN_EXEC_DT           DATE             NOT NULL WITH DEFAULT
, TRAN_EXEC_TM           TIME             NOT NULL WITH DEFAULT
, TRAN_SHARE_QY          DECIMAL(15, 5)   NOT NULL WITH DEFAULT
, TRAN_EXEC_PR_AM        DECIMAL(15, 5)   NOT NULL WITH DEFAULT
, TRAN_AM                DECIMAL(15, 2)   NOT NULL WITH DEFAULT
, ORIG_SYS_CD            CHAR(1)          NOT NULL WITH DEFAULT
, TRAN_ADD_DT            DATE             NOT NULL WITH DEFAULT
, TRAN_ADD_TM            TIME             NOT NULL WITH DEFAULT
, TRAN_PROC_DT           DATE             NOT NULL WITH DEFAULT
, RS_SUFF_ID             SMALLINT         NOT NULL WITH DEFAULT
, TRAN_PUBLCN_TS         TIMESTAMP        NOT NULL WITH DEFAULT
, ISSUE_CLASS_CD         CHAR(8)          NOT NULL WITH DEFAULT
, MJR_ISSUE_CLASS_CD     CHAR(8)          NOT NULL WITH DEFAULT
, SMA_CLASS_CD           CHAR(8)          NOT NULL WITH DEFAULT
, ISSUE_PRICE_DT         DATE             NOT NULL WITH DEFAULT
, ISSUE_PRICE_AM         DECIMAL(15, 5)   NOT NULL WITH DEFAULT
, ISSUE_MRGN_CD          CHAR(8)          NOT NULL WITH DEFAULT
, UNDR_ITEM_ISSUE_ID     INTEGER          NOT NULL WITH DEFAULT
, CONVERTIBLE_IN         CHAR(1)          NOT NULL WITH DEFAULT
, ISSUE_CONV_RT          DECIMAL(10, 5)   NOT NULL WITH DEFAULT
, M_SOURCE_CD            CHAR(5)          NOT NULL WITH DEFAULT
, M_ORF_OR_ORDRNO        DECIMAL(8)       NOT NULL WITH DEFAULT
, M_SEQ_NBR              INTEGER          NOT NULL WITH DEFAULT
, ORIG_ENV_CD            CHAR(2)          NOT NULL WITH DEFAULT
, BUY_SELL_CD            CHAR(1)          NOT NULL WITH DEFAULT
, UNBOX_SHARE_QY         DECIMAL(15, 5)   NOT NULL WITH DEFAULT
, DTRD_ELIG_IN           CHAR(1)
, AUDIT_ADD_USER_ID      CHAR(10)         NOT NULL WITH DEFAULT
, AUDIT_UPDT_TS          TIMESTAMP        NOT NULL WITH DEFAULT
```

```
  , AUDIT_UPDT_USER_ID    CHAR(10)          NOT NULL WITH DEFAULT
  , PRIMARY KEY ( ACCT_ID
                , AUDIT_ADD_TS
                )
  )
  IN ITSTRDB8.ITTSSJL
                                                                    ;
CREATE TABLE ITSO.SMA_EXTD_JRNL
  ( ACCT_ID               INTEGER           NOT NULL
  , AUDIT_ADD_TS          TIMESTAMP         NOT NULL
  , EXTD_JRNL_ADD_TS      TIMESTAMP         NOT NULL
  , NET_TRADE_AM          DECIMAL(15, 2)    NOT NULL WITH DEFAULT
  , DFLT_ACCS_SYM_NM      CHAR(15)          NOT NULL WITH DEFAULT
  , M_SEC_NR              DECIMAL(7)        NOT NULL WITH DEFAULT
  , TRANS_DESC_TX         CHAR(60)          NOT NULL WITH DEFAULT
  , CURR_OPTN_REQ_AM      DECIMAL(15, 2)    NOT NULL WITH DEFAULT
  , PREV_OPTN_REQ_AM      DECIMAL(15, 2)    NOT NULL WITH DEFAULT
  , OPTN_PEG_CALL_AM      DECIMAL(15, 2)    NOT NULL WITH DEFAULT
  , OPTN_PEG_PUT_AM       DECIMAL(15, 2)    NOT NULL WITH DEFAULT
  , DTRD_USED_QY          DECIMAL(15, 5)    NOT NULL WITH DEFAULT
  , DTRD_FIFO_UNUSE_QY    DECIMAL(15, 5)    NOT NULL WITH DEFAULT
  , DTRD_BUY_CYCLE_ID     SMALLINT          NOT NULL WITH DEFAULT
  , DTRD_BUY_EFF_DT       DATE              NOT NULL WITH DEFAULT
  , DTRD_BUY_SEQ_ID       SMALLINT          NOT NULL WITH DEFAULT
  , DTRD_NET_BUY_PR_AM    DECIMAL(15, 5)    NOT NULL WITH DEFAULT
  , DTRD_NET_SEL_PR_AM    DECIMAL(15, 5)    NOT NULL WITH DEFAULT
  , AUDIT_UPDT_TS         TIMESTAMP         NOT NULL WITH DEFAULT
  , AUDIT_UPDT_USER_ID    CHAR(10)          NOT NULL WITH DEFAULT
  , PRIMARY KEY ( ACCT_ID
                , AUDIT_ADD_TS
                , EXTD_JRNL_ADD_TS
                )
  )
  IN ITSTRDB8.ITTSSJE
                                                                    ;
CREATE TABLE ITSO.STAT_TRAN
  ( ACCT_ID               INTEGER           NOT NULL
  , TRAN_EFF_DT           INTEGER           NOT NULL
  , TRAN_SEQ_ID           SMALLINT          NOT NULL
  , TRAN_TYPE_CD          CHAR(11)          NOT NULL WITH DEFAULT
  , TRAN_SUBOR_TYPE_CD    CHAR(17)          NOT NULL WITH DEFAULT
  , TRAN_DR_CR_CD         CHAR(1)           NOT NULL WITH DEFAULT
  , TRAN_CXL_CD           CHAR(1)           NOT NULL WITH DEFAULT
  , TRAN_PAIR_CXL_IN      CHAR(1)           NOT NULL WITH DEFAULT
  , ACCTG_RULE_CD         CHAR(1)           NOT NULL WITH DEFAULT
  , ITEM_ISSUE_ID         INTEGER           NOT NULL WITH DEFAULT
  , RS_SUFF_ID            SMALLINT          NOT NULL WITH DEFAULT
  , OFSET_ACCT_ID         INTEGER           NOT NULL WITH DEFAULT
  , TRAN_EXEC_DT          INTEGER           NOT NULL WITH DEFAULT
```

```
  , TRAN_SHARE_QY         DECIMAL(15, 5)    NOT NULL WITH DEFAULT
  , TRAN_AM               DECIMAL(15, 2)    NOT NULL WITH DEFAULT
  , TRAN_RQST_TYPE_CD     CHAR(8)           NOT NULL WITH DEFAULT
  , TRAN_RQST_ID          INTEGER           NOT NULL WITH DEFAULT
  , ORIG_SYS_CD           CHAR(1)           NOT NULL WITH DEFAULT
  , TRAN_ADD_DT           INTEGER           NOT NULL WITH DEFAULT
  , TRAN_ADD_TM           INTEGER           NOT NULL WITH DEFAULT
  , STATUS_EXPIR_DT       INTEGER           NOT NULL WITH DEFAULT
  , TRAN_PROC_DT          INTEGER           NOT NULL WITH DEFAULT
  , TRAN_BUSN_TYPE_CD     CHAR(2)           NOT NULL WITH DEFAULT
  , AUDIT_INTG_CNTL_NR    SMALLINT          NOT NULL WITH DEFAULT
  , AUDIT_UPDT_USER_ID    CHAR(8)           NOT NULL WITH DEFAULT
  , TRAN_PUBLCN_TS        TIMESTAMP         NOT NULL WITH DEFAULT
  , PRIMARY KEY ( ACCT_ID
              , TRAN_EFF_DT
              , TRAN_SEQ_ID
              )
  )
  IN ITSTRDB1.ITTSSTR
                                                                     ;
CREATE TABLE ITSO.STAT_TRAN_MESSAGE
  ( ACCT_ID               INTEGER           NOT NULL
  , TRAN_EFF_DT           INTEGER           NOT NULL
  , TRAN_SEQ_ID           SMALLINT          NOT NULL
  , STAT_TRANS_DESC_TX    CHAR(60)          NOT NULL
  , AUDIT_USER_ID         CHAR(17)          NOT NULL WITH DEFAULT
  , AUDIT_UPDT_TS         TIMESTAMP         NOT NULL WITH DEFAULT
  , PRIMARY KEY ( ACCT_ID
              , TRAN_EFF_DT
              , TRAN_SEQ_ID
              )
  )
  IN ITSTRDB1.ITTSTMS
                                                                     ;
CREATE TABLE ITSO.TRAN
  ( ACCT_ID               INTEGER           NOT NULL
  , TRAN_CYCLE_ID         SMALLINT          NOT NULL
  , TRAN_EFF_DT           INTEGER           NOT NULL
  , TRAN_KEY_TM           SMALLINT          NOT NULL
  , TRAN_TYPE_CD          CHAR(2)           NOT NULL WITH DEFAULT
  , TRAN_SUBOR_TYPE_CD    CHAR(2)           NOT NULL WITH DEFAULT
  , TRAN_DR_CR_CD         CHAR(1)           NOT NULL WITH DEFAULT
  , TRAN_CXL_CD           CHAR(1)           NOT NULL WITH DEFAULT
  , TRAN_PAIR_CXL_IN      CHAR(1)           NOT NULL WITH DEFAULT
  , ACCTG_RULE_CD         CHAR(1)           NOT NULL WITH DEFAULT
  , ITEM_ISSUE_ID         INTEGER           NOT NULL WITH DEFAULT
  , OFSET_ACCT_ID         INTEGER           NOT NULL WITH DEFAULT
  , TRAN_EXEC_DT          INTEGER           NOT NULL WITH DEFAULT
  , TRAN_EXEC_TM          INTEGER           NOT NULL WITH DEFAULT
```

```
  , TRAN_SETTL_DT         INTEGER           NOT NULL WITH DEFAULT
  , TRAN_SHARE_QY         DECIMAL(15, 5)    NOT NULL WITH DEFAULT
  , TRAN_EXEC_PR          DECIMAL(15, 5)    NOT NULL WITH DEFAULT
  , TRAN_AM               DECIMAL(15, 2)    NOT NULL WITH DEFAULT
  , TRAN_RQST_TYPE_CD     CHAR(8)           NOT NULL WITH DEFAULT
  , TRAN_RQST_ID          INTEGER           NOT NULL WITH DEFAULT
  , ORIG_SYS_CD           CHAR(1)           NOT NULL WITH DEFAULT
  , STMT_SUPRS_CD         CHAR(1)           NOT NULL WITH DEFAULT
  , TRAN_ADD_DT           INTEGER           NOT NULL WITH DEFAULT
  , TRAN_ADD_TM           INTEGER           NOT NULL WITH DEFAULT
  , TRAN_PROC_DT          INTEGER           NOT NULL WITH DEFAULT
  , TRAN_PROC_TM          INTEGER           NOT NULL WITH DEFAULT
  , TRAN_BUSN_TYPE_CD     CHAR(2)           NOT NULL WITH DEFAULT
  , TRAN_CATG_SLCT_CD     CHAR(2)           NOT NULL WITH DEFAULT
  , RS_SUFF_ID            SMALLINT          NOT NULL WITH DEFAULT
  , AUDIT_INTG_CNTL_NR    SMALLINT          NOT NULL WITH DEFAULT
  , TRAN_PROC_CD          CHAR(1)           NOT NULL WITH DEFAULT
  , AUDIT_UPDT_USER_ID    CHAR(8)           NOT NULL WITH DEFAULT
  , TRAN_PUBLCN_TS        TIMESTAMP         NOT NULL WITH DEFAULT
  , PRIMARY KEY ( ACCT_ID
              , TRAN_CYCLE_ID
              , TRAN_EFF_DT
              , TRAN_KEY_TM
              )
  )
  IN ITSTRDB1.ITTSTRN
                                                                    ;
CREATE TABLE ITSO.TRAN_ASSCN
  ( ACCT_ID               INTEGER           NOT NULL
  , TRAN_CYCLE_ID         SMALLINT          NOT NULL
  , TRAN_EFF_DT           INTEGER           NOT NULL
  , TRAN_KEY_TM           SMALLINT          NOT NULL
  , RELT_TRAN_KEY_TM      SMALLINT          NOT NULL
  , TRAN_ASSCN_TYPE_CD    CHAR(2)           NOT NULL WITH DEFAULT
  , AUDIT_INTG_CNTL_NR    SMALLINT          NOT NULL WITH DEFAULT
  , PRIMARY KEY ( ACCT_ID
              , TRAN_CYCLE_ID
              , TRAN_EFF_DT
              , TRAN_KEY_TM
              , RELT_TRAN_KEY_TM
              )
  )
  IN ITSTRDB1.ITTSTTA
                                                                    ;
CREATE TABLE ITSO.TRAN_LGCY_SEC_DESC
  ( ACCT_ID               INTEGER           NOT NULL
  , TRAN_CYCLE_ID         SMALLINT          NOT NULL
  , TRAN_EFF_DT           INTEGER           NOT NULL
  , TRAN_KEY_TM           SMALLINT          NOT NULL
```

```
  , M_SECURITY_NR         INTEGER          NOT NULL WITH DEFAULT
  , M_SEC_DESC_1          CHAR(24)         NOT NULL WITH DEFAULT
  , M_SEC_DESC_2          CHAR(24)         NOT NULL WITH DEFAULT
  , M_SEC_DESC_3          CHAR(24)         NOT NULL WITH DEFAULT
  , M_SEC_CUSIP_NBR       CHAR(9)          NOT NULL WITH DEFAULT
  , M_SEC_SYM             CHAR(5)          NOT NULL WITH DEFAULT
  , UNDER_M_SEC_NBR       INTEGER          NOT NULL WITH DEFAULT
  , PRIMARY KEY ( ACCT_ID
              , TRAN_CYCLE_ID
              , TRAN_EFF_DT
              , TRAN_KEY_TM
              )
  )
  IN ITSTRDB1.ITTSLSD
                                                                  ;
 CREATE TABLE ITSO.UNIQ_ID_RGSTR
  ( ACCT_ID               INTEGER          NOT NULL
  , RGSTR_CREATE_TS       TIMESTAMP        NOT NULL
  , ID_RGSTR_CT           SMALLINT         NOT NULL
  , AUDIT_UPDT_TS         TIMESTAMP        NOT NULL WITH DEFAULT
  , AUDIT_UPDT_USER_ID    CHAR(10)         NOT NULL WITH DEFAULT
  , PRIMARY KEY ( ACCT_ID
              , RGSTR_CREATE_TS
              )
  )
  IN ITSTHDB2.ITTSUIR
                                                                  ;

---======================== INDEX DEFINITIONS ========================

 CREATE TYPE 2 UNIQUE INDEX ITSO.IXBLN1
  ON ITSO.BAL
  (ACCT_ID ASC, ACCTG_RULE_CD ASC, BAL_TYPE_CD ASC, BAL_EFF_DT DESC)
  CLUSTER
  ( PART  1 VALUES(19999999)
          USING STOGROUP ITSIXSG1
                 PRIQTY 964
                 SECQTY 504
           PCTFREE    20
  , PART  2 VALUES(29999999)
          USING STOGROUP ITSIXSG1
                 PRIQTY 864
                 SECQTY 432
           PCTFREE    20
  , PART  3 VALUES(39999999)
          USING STOGROUP ITSIXSG1
                 PRIQTY 727
                 SECQTY 396
           PCTFREE    20
```

```
  , PART  4 VALUES(49999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 655
                SECQTY 360
          PCTFREE    20
  , PART  5 VALUES(59999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 662
                SECQTY 360
          PCTFREE    20
  , PART  6 VALUES(69999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 648
                SECQTY 324
          PCTFREE    20
  , PART  7 VALUES(79999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 626
                SECQTY 324
          PCTFREE    20
  , PART  8 VALUES(89999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 640
                SECQTY 324
          PCTFREE    20
  , PART  9 VALUES(99999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 496
                SECQTY 252
          PCTFREE    20
  )
  BUFFERPOOL BP2
  CLOSE      NO
                                                                  ;
CREATE TYPE 2 UNIQUE INDEX ITSO.IXBPT1
  ON ITSO.BPT_PARM_TABLE
  (PARAMETER_NM ASC, PARM_PGM_FUNC_NM ASC, ENVIRONMENT_ID ASC,
  PARM_SUB_FUNC_NM ASC, PARM_FIRST_EFF_TS DESC)
  CLUSTER
  USING STOGROUP ITSIXSG1
        PRIQTY 14
        SECQTY 36
  PCTFREE    0
  BUFFERPOOL BP2
  CLOSE      NO
                                                                  ;
CREATE TYPE 2 UNIQUE INDEX ITSO.IXIMT1
  ON ITSO.IME_TRANSACTION
  (ACCOUNT_ID ASC, AUDIT_UPDT_TS DESC)
```

```
CLUSTER
( PART  1 VALUES(12999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 86
                SECQTY 72
          PCTFREE    20
, PART  2 VALUES(19999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 129
                SECQTY 72
          PCTFREE    20
, PART  3 VALUES(24999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 122
                SECQTY 72
          PCTFREE    20
, PART  4 VALUES(29999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 93
                SECQTY 72
          PCTFREE    20
, PART  5 VALUES(34999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 115
                SECQTY 72
          PCTFREE    20
, PART  6 VALUES(39999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 79
                SECQTY 72
          PCTFREE    20
, PART  7 VALUES(44999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 86
                SECQTY 72
          PCTFREE    20
, PART  8 VALUES(49999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 72
                SECQTY 36
          PCTFREE    20
, PART  9 VALUES(54999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 108
                SECQTY 72
          PCTFREE    20
, PART 10 VALUES(59999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 79
```

```
                SECQTY 72
          PCTFREE    20
  , PART 11 VALUES(64999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 86
                SECQTY 72
          PCTFREE    20
  , PART 12 VALUES(69999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 72
                SECQTY 36
          PCTFREE    20
  , PART 13 VALUES(74999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 86
                SECQTY 72
          PCTFREE    20
  , PART 14 VALUES(79999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 64
                SECQTY 36
          PCTFREE    20
  , PART 15 VALUES(84999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 86
                SECQTY 72
          PCTFREE    20
  , PART 16 VALUES(89999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 79
                SECQTY 72
          PCTFREE    20
  , PART 17 VALUES(94999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 86
                SECQTY 72
          PCTFREE    20
  , PART 18 VALUES(99999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 36
                SECQTY 36
          PCTFREE    20
  )
  BUFFERPOOL BP2
  CLOSE      NO
                                                                  ;
CREATE TYPE 2 UNIQUE INDEX ITSO.IXISCC
  ON ITSO.ISSUE_CLASS
  (ISSUE_CLASS_CD ASC)
```

```
  CLUSTER
  USING STOGROUP ITSIXSG1
        PRIQTY 96
        SECQTY 48
  BUFFERPOOL BP2
  CLOSE      NO
                                                                   ;
CREATE TYPE 2 UNIQUE INDEX ITSO.IXISSC
  ON ITSO.ITEM_ISSUE
  (ITEM_ISSUE_ID ASC)
  CLUSTER
  ( PART  1 VALUES(195495394)
           USING STOGROUP ITSIXSG1
                  PRIQTY 43
                  SECQTY 36
  , PART  2 VALUES(396264601)
           USING STOGROUP ITSIXSG1
                  PRIQTY 43
                  SECQTY 36
  , PART  3 VALUES(596924608)
           USING STOGROUP ITSIXSG1
                  PRIQTY 43
                  SECQTY 36
  , PART  4 VALUES(797708399)
           USING STOGROUP ITSIXSG1
                  PRIQTY 43
                  SECQTY 36
  , PART  5 VALUES(998033881)
           USING STOGROUP ITSIXSG1
                  PRIQTY 43
                  SECQTY 36
  , PART  6 VALUES(1198846359)
           USING STOGROUP ITSIXSG1
                  PRIQTY 43
                  SECQTY 36
  , PART  7 VALUES(1400220419)
           USING STOGROUP ITSIXSG1
                  PRIQTY 43
                  SECQTY 36
  , PART  8 VALUES(1646595029)
           USING STOGROUP ITSIXSG1
                  PRIQTY 50
                  SECQTY 36
  , PART  9 VALUES(1896563703)
           USING STOGROUP ITSIXSG1
                  PRIQTY 50
                  SECQTY 36
  , PART 10 VALUES(2146849493)
           USING STOGROUP ITSIXSG1
```

```
                PRIQTY 50
                SECQTY 36
  )
  BUFFERPOOL BP2
  CLOSE      NO
                                                                    ;
CREATE TYPE 2  INDEX ITSO.IXISS2
  ON ITSO.ITEM_ISSUE
  (ITEM_ISSUE_NM ASC)
  USING STOGROUP ITSIXSG1
        PRIQTY 2829
        SECQTY 1310
  PCTFREE    20
  BUFFERPOOL BP2
  CLOSE      NO
                                                                    ;
CREATE TYPE 2  INDEX ITSO.IXISS3
  ON ITSO.ITEM_ISSUE
  (ISSUE_CUSIP_NR ASC)
  USING STOGROUP ITSIXSG1
        PRIQTY 352
        SECQTY 180
  PCTFREE    20
  BUFFERPOOL BP2
  CLOSE      NO
                                                                    ;
CREATE TYPE 2  INDEX ITSO.IXISS4
  ON ITSO.ITEM_ISSUE
  (DFLT_ACCS_SYM_NM ASC)
  USING STOGROUP ITSIXSG1
        PRIQTY 547
        SECQTY 288
  PCTFREE    20
  BUFFERPOOL BP2
  CLOSE      NO
                                                                    ;
CREATE TYPE 2 UNIQUE INDEX ITSO.IXISS6
  ON ITSO.ITEM_ISSUE
  (ISSUER_ID ASC, ITEM_ISSUE_ID ASC)
  USING STOGROUP ITSIXSG1
        PRIQTY 640
        SECQTY 324
  PCTFREE    20
  BUFFERPOOL BP2
  CLOSE      NO
                                                                    ;
CREATE TYPE 2  INDEX ITSO.IXISS7
  ON ITSO.ITEM_ISSUE
  (ISSUE_CLASS_CD ASC, ITEM_ISSUE_NM ASC)
```

```
  USING STOGROUP ITSIXSG1
        PRIQTY 3175
        SECQTY 1310
  PCTFREE    20
  BUFFERPOOL BP2
  CLOSE      NO
                                                                      ;
CREATE TYPE 2  INDEX ITSO.IXISS8
  ON ITSO.ITEM_ISSUE
  (MJR_ISSUE_CLASS_CD ASC, ITEM_ISSUE_NM ASC, DFLT_ACCS_SYM_NM ASC)
  USING STOGROUP ITSIXSG1
        PRIQTY 3801
        SECQTY 1310
  PCTFREE    20
  BUFFERPOOL BP2
  CLOSE      NO
                                                                      ;
CREATE TYPE 2  INDEX ITSO.IXISS9
  ON ITSO.ITEM_ISSUE
  (UNDR_ITEM_ISSUE_ID ASC)
  USING STOGROUP ITSIXSG1
        PRIQTY 216
        SECQTY 108
  BUFFERPOOL BP2
  CLOSE      NO
                                                                      ;
CREATE TYPE 2 UNIQUE INDEX ITSO.IXLFC1
  ON ITSO.LEGACY_FUND_TRAN
  (ACCT_ID ASC, TRAN_CYCLE_ID ASC, TRAN_EFF_DT DESC, TRAN_KEY_TM ASC)
  CLUSTER
  ( PART  1 VALUES(19999999)
           USING STOGROUP ITSIXSG1
                  PRIQTY 244
                  SECQTY 144
            PCTFREE    20
  , PART  2 VALUES(29999999)
           USING STOGROUP ITSIXSG1
                  PRIQTY 172
                  SECQTY 108
            PCTFREE    20
  , PART  3 VALUES(39999999)
           USING STOGROUP ITSIXSG1
                  PRIQTY 151
                  SECQTY 108
            PCTFREE    20
  , PART  4 VALUES(49999999)
           USING STOGROUP ITSIXSG1
                  PRIQTY 122
                  SECQTY 72
```

```
          PCTFREE    20
, PART  5 VALUES(59999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 129
                SECQTY 72
          PCTFREE    20
, PART  6 VALUES(69999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 129
                SECQTY 72
          PCTFREE    20
, PART  7 VALUES(79999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 129
                SECQTY 72
          PCTFREE    20
, PART  8 VALUES(89999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 122
                SECQTY 72
          PCTFREE    20
, PART  9 VALUES(99999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 93
                SECQTY 72
          PCTFREE    20
)
BUFFERPOOL BP2
CLOSE      NO
                                                                      ;
CREATE TYPE 2 UNIQUE INDEX ITSO.IXLIC1
  ON ITSO.LEGACY_ISSUE_TRAN
  (ACCT_ID ASC, TRAN_CYCLE_ID ASC, TRAN_EFF_DT DESC, TRAN_KEY_TM ASC)
  CLUSTER
  ( PART  1 VALUES(39999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 79
                SECQTY 72
          PCTFREE    20
  , PART  2 VALUES(69999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 57
                SECQTY 36
          PCTFREE    20
  , PART  3 VALUES(99999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 43
                SECQTY 36
          PCTFREE    20
```

```
  )
  BUFFERPOOL BP2
  CLOSE      NO
                                                                       ;
CREATE TYPE 2 UNIQUE INDEX ITSO.IXLSD1
  ON ITSO.TRAN_LGCY_SEC_DESC
  (ACCT_ID ASC, TRAN_CYCLE_ID ASC, TRAN_EFF_DT DESC, TRAN_KEY_TM ASC)
  CLUSTER
  ( PART  1 VALUES(19999999)
           USING STOGROUP ITSIXSG1
                  PRIQTY 50
                  SECQTY 36
            PCTFREE    20
  , PART  2 VALUES(29999999)
           USING STOGROUP ITSIXSG1
                  PRIQTY 43
                  SECQTY 36
            PCTFREE    20
  , PART  3 VALUES(39999999)
           USING STOGROUP ITSIXSG1
                  PRIQTY 43
                  SECQTY 36
            PCTFREE    20
  , PART  4 VALUES(49999999)
           USING STOGROUP ITSIXSG1
                  PRIQTY 57
                  SECQTY 36
            PCTFREE    20
  , PART  5 VALUES(59999999)
           USING STOGROUP ITSIXSG1
                  PRIQTY 57
                  SECQTY 36
            PCTFREE    20
  , PART  6 VALUES(69999999)
           USING STOGROUP ITSIXSG1
                  PRIQTY 57
                  SECQTY 36
            PCTFREE    20
  , PART  7 VALUES(79999999)
           USING STOGROUP ITSIXSG1
                  PRIQTY 57
                  SECQTY 36
            PCTFREE    20
  , PART  8 VALUES(89999999)
           USING STOGROUP ITSIXSG1
                  PRIQTY 57
                  SECQTY 36
            PCTFREE    20
  , PART  9 VALUES(99999999)
```

```
          USING STOGROUP ITSIXSG1
                 PRIQTY 36
                 SECQTY 36
           PCTFREE    20
  )
  BUFFERPOOL BP2
  CLOSE      NO
                                                                  ;
CREATE TYPE 2 UNIQUE INDEX ITSO.IXLTR1
  ON ITSO.LEGACY_TRADE_TRAN
  (ACCT_ID ASC, TRAN_CYCLE_ID ASC, TRAN_EFF_DT DESC, TRAN_KEY_TM ASC)
  CLUSTER
  ( PART  1 VALUES(19999999)
          USING STOGROUP ITSIXSG1
                 PRIQTY 266
                 SECQTY 144
           PCTFREE    20
  , PART  2 VALUES(29999999)
          USING STOGROUP ITSIXSG1
                 PRIQTY 187
                 SECQTY 108
           PCTFREE    20
  , PART  3 VALUES(39999999)
          USING STOGROUP ITSIXSG1
                 PRIQTY 151
                 SECQTY 108
           PCTFREE    20
  , PART  4 VALUES(49999999)
          USING STOGROUP ITSIXSG1
                 PRIQTY 136
                 SECQTY 72
           PCTFREE    20
  , PART  5 VALUES(59999999)
          USING STOGROUP ITSIXSG1
                 PRIQTY 136
                 SECQTY 72
           PCTFREE    20
  , PART  6 VALUES(69999999)
          USING STOGROUP ITSIXSG1
                 PRIQTY 136
                 SECQTY 72
           PCTFREE    20
  , PART  7 VALUES(79999999)
          USING STOGROUP ITSIXSG1
                 PRIQTY 136
                 SECQTY 72
           PCTFREE    20
  , PART  8 VALUES(89999999)
          USING STOGROUP ITSIXSG1
```

```
                  PRIQTY 144
                  SECQTY 72
            PCTFREE    20
  , PART  9 VALUES(99999999)
           USING STOGROUP ITSIXSG1
                  PRIQTY 115
                  SECQTY 72
            PCTFREE    20
  )
  BUFFERPOOL BP2
  CLOSE      NO
                                                                    ;
CREATE TYPE 2 UNIQUE INDEX ITSO.IXMRR1
  ON ITSO.MRGN_REQ_RULE
  (ISSUE_CLASS_CD ASC, EQTY_REQ_TYPE_CD ASC, EQTY_ACCTG_TYPE_CD ASC,
  RANGE_BKPT_QY DESC)
  CLUSTER
  USING STOGROUP ITSIXSG1
        PRIQTY 96
        SECQTY 48
  PCTFREE    20
  BUFFERPOOL BP2
  CLOSE      NO
                                                                    ;
CREATE TYPE 2  INDEX ITSO.IXMTL
  ON ITSO.MEMO_TRAN_LIST
  (ACCT_ID ASC, TRAN_PROC_DT DESC)
  CLUSTER
  USING STOGROUP ITSIXSG1
        PRIQTY 1476
        SECQTY 756
  PCTFREE    30
  BUFFERPOOL BP2
  CLOSE      NO
                                                                    ;
CREATE TYPE 2 UNIQUE INDEX ITSO.IXOB1
  ON ITSO.OPEN_DAILY_BAL_1
  (ACCT_ID ASC, BAL_CLOSING_EFF_DT DESC)
  CLUSTER
  ( PART  1 VALUES(14999999)
           USING STOGROUP ITSIXSG1
                  PRIQTY 108
                  SECQTY 72
            PCTFREE    5
  , PART  2 VALUES(19999999)
           USING STOGROUP ITSIXSG1
                  PRIQTY 79
                  SECQTY 72
            PCTFREE    5
```

```
, PART  3 VALUES(24999999)
        USING STOGROUP ITSIXSG1
               PRIQTY 93
               SECQTY 72
         PCTFREE    5
, PART  4 VALUES(29999999)
        USING STOGROUP ITSIXSG1
               PRIQTY 79
               SECQTY 72
         PCTFREE    5
, PART  5 VALUES(34999999)
        USING STOGROUP ITSIXSG1
               PRIQTY 86
               SECQTY 72
         PCTFREE    5
, PART  6 VALUES(39999999)
        USING STOGROUP ITSIXSG1
               PRIQTY 64
               SECQTY 36
         PCTFREE    5
, PART  7 VALUES(44999999)
        USING STOGROUP ITSIXSG1
               PRIQTY 64
               SECQTY 36
         PCTFREE    5
, PART  8 VALUES(49999999)
        USING STOGROUP ITSIXSG1
               PRIQTY 57
               SECQTY 36
         PCTFREE    5
, PART  9 VALUES(54999999)
        USING STOGROUP ITSIXSG1
               PRIQTY 64
               SECQTY 36
         PCTFREE    5
, PART 10 VALUES(59999999)
        USING STOGROUP ITSIXSG1
               PRIQTY 57
               SECQTY 36
         PCTFREE    5
, PART 11 VALUES(64999999)
        USING STOGROUP ITSIXSG1
               PRIQTY 64
               SECQTY 36
         PCTFREE    5
, PART 12 VALUES(69999999)
        USING STOGROUP ITSIXSG1
               PRIQTY 57
               SECQTY 36
```

```
            PCTFREE    5
  , PART 13 VALUES(74999999)
           USING STOGROUP ITSIXSG1
                  PRIQTY 64
                  SECQTY 36
            PCTFREE    5
  , PART 14 VALUES(79999999)
           USING STOGROUP ITSIXSG1
                  PRIQTY 57
                  SECQTY 36
            PCTFREE    5
  , PART 15 VALUES(84999999)
           USING STOGROUP ITSIXSG1
                  PRIQTY 64
                  SECQTY 36
            PCTFREE    5
  , PART 16 VALUES(89999999)
           USING STOGROUP ITSIXSG1
                  PRIQTY 57
                  SECQTY 36
            PCTFREE    5
  , PART 17 VALUES(94999999)
           USING STOGROUP ITSIXSG1
                  PRIQTY 64
                  SECQTY 36
            PCTFREE    5
  , PART 18 VALUES(99999999)
           USING STOGROUP ITSIXSG1
                  PRIQTY 28
                  SECQTY 36
            PCTFREE    5
  )
  BUFFERPOOL BP2
  CLOSE      NO
                                                                     ;
CREATE TYPE 2 UNIQUE INDEX ITSO.IXOB1A
  ON ITSO.OPEN_DAILY_BAL_2
  (ACCT_ID ASC, BAL_CLOSING_EFF_DT DESC)
  CLUSTER
  ( PART  1 VALUES(14999999)
           USING STOGROUP ITSIXSG1
                  PRIQTY 108
                  SECQTY 72
            PCTFREE    5
  , PART  2 VALUES(19999999)
           USING STOGROUP ITSIXSG1
                  PRIQTY 79
                  SECQTY 72
            PCTFREE    5
```

```
, PART  3 VALUES(24999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 93
                SECQTY 72
          PCTFREE    5
, PART  4 VALUES(29999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 79
                SECQTY 72
          PCTFREE    5
, PART  5 VALUES(34999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 86
                SECQTY 72
          PCTFREE    5
, PART  6 VALUES(39999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 64
                SECQTY 36
          PCTFREE    5
, PART  7 VALUES(44999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 64
                SECQTY 36
          PCTFREE    5
, PART  8 VALUES(49999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 57
                SECQTY 36
          PCTFREE    5
, PART  9 VALUES(54999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 64
                SECQTY 36
          PCTFREE    5
, PART 10 VALUES(59999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 57
                SECQTY 36
          PCTFREE    5
, PART 11 VALUES(64999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 64
                SECQTY 36
          PCTFREE    5
, PART 12 VALUES(69999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 64
                SECQTY 36
```

```
           PCTFREE    5
  , PART 13 VALUES(74999999)
          USING STOGROUP ITSIXSG1
                 PRIQTY 64
                 SECQTY 36
           PCTFREE    5
  , PART 14 VALUES(79999999)
          USING STOGROUP ITSIXSG1
                 PRIQTY 57
                 SECQTY 36
           PCTFREE    5
  , PART 15 VALUES(84999999)
          USING STOGROUP ITSIXSG1
                 PRIQTY 64
                 SECQTY 36
           PCTFREE    5
  , PART 16 VALUES(89999999)
          USING STOGROUP ITSIXSG1
                 PRIQTY 57
                 SECQTY 36
           PCTFREE    5
  , PART 17 VALUES(94999999)
          USING STOGROUP ITSIXSG1
                 PRIQTY 64
                 SECQTY 36
           PCTFREE    5
  , PART 18 VALUES(99999999)
          USING STOGROUP ITSIXSG1
                 PRIQTY 28
                 SECQTY 36
           PCTFREE    5
  )
  BUFFERPOOL BP2
  CLOSE      NO
                                                                   ;
CREATE TYPE 2 UNIQUE INDEX ITSO.IXOPT1
  ON ITSO.OPTIONS
  (ITEM_ISSUE_ID ASC)
  CLUSTER
  USING STOGROUP ITSIXSG1
        PRIQTY 331
        SECQTY 180
  PCTFREE    20
  BUFFERPOOL BP2
  CLOSE      NO
                                                                   ;
CREATE TYPE 2  INDEX ITSO.IXOPT2
  ON ITSO.OPTIONS
  (OPTION_EXPR_DT ASC, STRIKE_PRICE_AM ASC)
```

```
  USING STOGROUP ITSIXSG1
        PRIQTY 180
        SECQTY 108
  PCTFREE    20
  BUFFERPOOL BP2
  CLOSE      NO
                                                                     ;
CREATE TYPE 2 UNIQUE INDEX ITSO.IXPOS1
  ON ITSO.POSTN
  (ACCT_ID ASC, ACCTG_RULE_CD ASC, ITEM_ISSUE_ID ASC, RS_SUFF_ID ASC,
  CONTROL_CD ASC, POSTN_TYPE_CD ASC, POSTN_EFF_DT DESC)
  CLUSTER
  ( PART  1 VALUES(19999999)
           USING STOGROUP ITSIXSG1
                  PRIQTY 2210
                  SECQTY 1116
            PCTFREE    20
  , PART  2 VALUES(29999999)
           USING STOGROUP ITSIXSG1
                  PRIQTY 1951
                  SECQTY 1008
            PCTFREE    20
  , PART  3 VALUES(39999999)
           USING STOGROUP ITSIXSG1
                  PRIQTY 1713
                  SECQTY 864
            PCTFREE    20
  , PART  4 VALUES(49999999)
           USING STOGROUP ITSIXSG1
                  PRIQTY 1526
                  SECQTY 792
            PCTFREE    20
  , PART  5 VALUES(59999999)
           USING STOGROUP ITSIXSG1
                  PRIQTY 1519
                  SECQTY 792
            PCTFREE    20
  , PART  6 VALUES(69999999)
           USING STOGROUP ITSIXSG1
                  PRIQTY 1526
                  SECQTY 792
            PCTFREE    20
  , PART  7 VALUES(79999999)
           USING STOGROUP ITSIXSG1
                  PRIQTY 1490
                  SECQTY 756
            PCTFREE    20
  , PART  8 VALUES(89999999)
           USING STOGROUP ITSIXSG1
```

```
                PRIQTY 1504
                SECQTY 756
          PCTFREE    20
 , PART  9 VALUES(99999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 1152
                SECQTY 576
          PCTFREE    20
 )
 BUFFERPOOL BP2
 CLOSE      NO
                                                                     ;
CREATE TYPE 2 UNIQUE INDEX ITSO.IXRQE1
 ON ITSO.RULE_QLF_VALUE
 (CMPNT_ID ASC, PRI_RULE_ID ASC, PRI_QLF_ID ASC, ATTRB_ID ASC,
 QLF_ROW_ID ASC, COMM_CALC_IN ASC, ATTRB_VALU_OPR_CD ASC)
 CLUSTER
 USING STOGROUP ITSIXSG1
       PRIQTY 79
       SECQTY 72
 BUFFERPOOL BP2
 CLOSE      NO
                                                                     ;
CREATE TYPE 2  INDEX ITSO.IXRQE2
 ON ITSO.RULE_QLF_VALUE
 (PRI_QLF_ID ASC, ATTRB_ID ASC, QLF_ROW_ID ASC, COMM_CALC_IN ASC)
 USING STOGROUP ITSIXSG1
       PRIQTY 21
       SECQTY 36
 BUFFERPOOL BP2
 CLOSE      NO
                                                                     ;
CREATE TYPE 2 UNIQUE INDEX ITSO.IXRQE3
 ON ITSO.RULE_QLF_VALUE
 (CMPNT_ID ASC, PRI_RULE_ID ASC, PRI_QLF_ID ASC, ATTRB_ID ASC,
 QLF_ROW_ID ASC, ATTRB_VALU_OPR_CD ASC, SML_INTEGER_QY ASC,
 COMM_CALC_IN ASC)
 USING STOGROUP ITSIXSG1
       PRIQTY 100
       SECQTY 72
 BUFFERPOOL BP2
 CLOSE      NO
                                                                     ;
CREATE TYPE 2 UNIQUE INDEX ITSO.IXRQE4
 ON ITSO.RULE_QLF_VALUE
 (CMPNT_ID ASC, PRI_RULE_ID ASC, PRI_QLF_ID ASC, ATTRB_ID ASC,
 QLF_ROW_ID ASC, ATTRB_VALU_OPR_CD ASC, INTEGER_QY ASC, COMM_CALC_IN
 ASC)
 USING STOGROUP ITSIXSG1
```

```
       PRIQTY 86
       SECQTY 72
  BUFFERPOOL BP2
  CLOSE      NO
                                                                      ;
CREATE TYPE 2 UNIQUE INDEX ITSO.IXRQE5
  ON ITSO.RULE_QLF_VALUE
  (CMPNT_ID ASC, PRI_RULE_ID ASC, PRI_QLF_ID ASC, ATTRB_ID ASC,
  QLF_ROW_ID ASC, ATTRB_VALU_OPR_CD ASC, DEC_QY ASC, COMM_CALC_IN ASC)
  USING STOGROUP ITSIXSG1
       PRIQTY 100
       SECQTY 72
  BUFFERPOOL BP2
  CLOSE      NO
                                                                      ;
CREATE TYPE 2 UNIQUE INDEX ITSO.IXRQE6
  ON ITSO.RULE_QLF_VALUE
  (CMPNT_ID ASC, PRI_RULE_ID ASC, PRI_QLF_ID ASC, ATTRB_ID ASC,
  QLF_ROW_ID ASC, ATTRB_VALU_OPR_CD ASC, QLF_TM ASC, COMM_CALC_IN ASC)
  USING STOGROUP ITSIXSG1
       PRIQTY 86
       SECQTY 72
  BUFFERPOOL BP2
  CLOSE      NO
                                                                      ;
CREATE TYPE 2 UNIQUE INDEX ITSO.IXRQE7
  ON ITSO.RULE_QLF_VALUE
  (CMPNT_ID ASC, PRI_RULE_ID ASC, PRI_QLF_ID ASC, ATTRB_ID ASC,
  QLF_ROW_ID ASC, ATTRB_VALU_OPR_CD ASC, QLF_DT ASC, COMM_CALC_IN ASC)
  USING STOGROUP ITSIXSG1
       PRIQTY 100
       SECQTY 72
  BUFFERPOOL BP2
  CLOSE      NO
                                                                      ;
CREATE TYPE 2 UNIQUE INDEX ITSO.IXRQE8
  ON ITSO.RULE_QLF_VALUE
  (CMPNT_ID ASC, PRI_RULE_ID ASC, PRI_QLF_ID ASC, ATTRB_ID ASC,
  QLF_ROW_ID ASC, ATTRB_VALU_OPR_CD ASC, COMM_CALC_IN ASC, CHAR_TX
  ASC)
  USING STOGROUP ITSIXSG1
       PRIQTY 288
       SECQTY 144
  BUFFERPOOL BP2
  CLOSE      NO
                                                                      ;
CREATE TYPE 2 UNIQUE INDEX ITSO.IXSJE1
  ON ITSO.SMA_EXTD_JRNL
  (ACCT_ID ASC, AUDIT_ADD_TS ASC, EXTD_JRNL_ADD_TS ASC)
```

```
CLUSTER
( PART  1 VALUES(12999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 345
                SECQTY 180
          PCTFREE    20
, PART  2 VALUES(19999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 453
                SECQTY 252
          PCTFREE    20
, PART  3 VALUES(24999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 424
                SECQTY 216
          PCTFREE    20
, PART  4 VALUES(29999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 309
                SECQTY 180
          PCTFREE    20
, PART  5 VALUES(34999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 381
                SECQTY 216
          PCTFREE    20
, PART  6 VALUES(39999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 259
                SECQTY 144
          PCTFREE    20
, PART  7 VALUES(44999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 295
                SECQTY 180
          PCTFREE    20
, PART  8 VALUES(49999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 288
                SECQTY 144
          PCTFREE    20
, PART  9 VALUES(54999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 367
                SECQTY 216
          PCTFREE    20
, PART 10 VALUES(59999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 266
```

```
                SECQTY 144
           PCTFREE    20
 , PART 11 VALUES(64999999)
          USING STOGROUP ITSIXSG1
                PRIQTY 280
                SECQTY 144
           PCTFREE    20
 , PART 12 VALUES(69999999)
          USING STOGROUP ITSIXSG1
                PRIQTY 280
                SECQTY 144
           PCTFREE    20
 , PART 13 VALUES(74999999)
          USING STOGROUP ITSIXSG1
                PRIQTY 302
                SECQTY 180
           PCTFREE    20
 , PART 14 VALUES(79999999)
          USING STOGROUP ITSIXSG1
                PRIQTY 230
                SECQTY 144
           PCTFREE    20
 , PART 15 VALUES(84999999)
          USING STOGROUP ITSIXSG1
                PRIQTY 324
                SECQTY 180
           PCTFREE    20
 , PART 16 VALUES(89999999)
          USING STOGROUP ITSIXSG1
                PRIQTY 266
                SECQTY 144
           PCTFREE    20
 , PART 17 VALUES(94999999)
          USING STOGROUP ITSIXSG1
                PRIQTY 302
                SECQTY 180
           PCTFREE    20
 , PART 18 VALUES(99999999)
          USING STOGROUP ITSIXSG1
                PRIQTY 165
                SECQTY 108
           PCTFREE    20
 )
 BUFFERPOOL BP2
 CLOSE      NO
                                                                      ;
CREATE TYPE 2 UNIQUE INDEX ITSO.IXSJL1
 ON ITSO.SMA_BASE_JRNL
 (ACCT_ID ASC, AUDIT_ADD_TS ASC)
```

```
CLUSTER
( PART  1 VALUES(12999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 453
                SECQTY 252
          PCTFREE    20
, PART  2 VALUES(19999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 669
                SECQTY 360
          PCTFREE    20
, PART  3 VALUES(24999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 583
                SECQTY 324
          PCTFREE    20
, PART  4 VALUES(29999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 468
                SECQTY 252
          PCTFREE    20
, PART  5 VALUES(34999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 532
                SECQTY 288
          PCTFREE    20
, PART  6 VALUES(39999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 381
                SECQTY 216
          PCTFREE    20
, PART  7 VALUES(44999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 446
                SECQTY 252
          PCTFREE    20
, PART  8 VALUES(49999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 417
                SECQTY 216
          PCTFREE    20
, PART  9 VALUES(54999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 475
                SECQTY 252
          PCTFREE    20
, PART 10 VALUES(59999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 403
```

```
                SECQTY 216
          PCTFREE    20
  , PART 11 VALUES(64999999)
          USING STOGROUP ITSIXSG1
                PRIQTY 424
                SECQTY 216
          PCTFREE    20
  , PART 12 VALUES(69999999)
          USING STOGROUP ITSIXSG1
                PRIQTY 410
                SECQTY 216
          PCTFREE    20
  , PART 13 VALUES(74999999)
          USING STOGROUP ITSIXSG1
                PRIQTY 446
                SECQTY 252
          PCTFREE    20
  , PART 14 VALUES(79999999)
          USING STOGROUP ITSIXSG1
                PRIQTY 352
                SECQTY 180
          PCTFREE    20
  , PART 15 VALUES(84999999)
          USING STOGROUP ITSIXSG1
                PRIQTY 446
                SECQTY 252
          PCTFREE    20
  , PART 16 VALUES(89999999)
          USING STOGROUP ITSIXSG1
                PRIQTY 396
                SECQTY 216
          PCTFREE    20
  , PART 17 VALUES(94999999)
          USING STOGROUP ITSIXSG1
                PRIQTY 439
                SECQTY 252
          PCTFREE    20
  , PART 18 VALUES(99999999)
          USING STOGROUP ITSIXSG1
                PRIQTY 201
                SECQTY 108
          PCTFREE    20
  )
  BUFFERPOOL BP2
  CLOSE      NO
                                                              ;
CREATE TYPE 2 UNIQUE INDEX ITSO.IXSTR1
  ON ITSO.STAT_TRAN
  (ACCT_ID ASC, TRAN_EFF_DT DESC, TRAN_SEQ_ID ASC)
```

```
  USING STOGROUP ITSIXSG1
        PRIQTY 252
        SECQTY 144
  PCTFREE    20
  BUFFERPOOL BP2
  CLOSE      NO
                                                                     ;
CREATE TYPE 2 UNIQUE INDEX ITSO.IXSTR2
  ON ITSO.STAT_TRAN
  (ACCT_ID ASC, TRAN_EFF_DT DESC, TRAN_SEQ_ID ASC, TRAN_PROC_DT DESC)
  CLUSTER
  ( PART  1 VALUES(19999999)
           USING STOGROUP ITSIXSG1
                  PRIQTY 64
                  SECQTY 36
            PCTFREE    20
  , PART  2 VALUES(29999999)
           USING STOGROUP ITSIXSG1
                  PRIQTY 3297
                  SECQTY 180
            PCTFREE    20
  , PART  3 VALUES(39999999)
           USING STOGROUP ITSIXSG1
                  PRIQTY 57
                  SECQTY 36
            PCTFREE    20
  , PART  4 VALUES(49999999)
           USING STOGROUP ITSIXSG1
                  PRIQTY 2736
                  SECQTY 180
            PCTFREE    20
  , PART  5 VALUES(59999999)
           USING STOGROUP ITSIXSG1
                  PRIQTY 2923
                  SECQTY 180
            PCTFREE    20
  , PART  6 VALUES(69999999)
           USING STOGROUP ITSIXSG1
                  PRIQTY 2923
                  SECQTY 180
            PCTFREE    20
  , PART  7 VALUES(79999999)
           USING STOGROUP ITSIXSG1
                  PRIQTY 50
                  SECQTY 36
            PCTFREE    20
  , PART  8 VALUES(89999999)
           USING STOGROUP ITSIXSG1
                  PRIQTY 2923
```

```
                SECQTY 180
          PCTFREE    20
  , PART  9 VALUES(99999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 36
                SECQTY 36
          PCTFREE    20
  )
  BUFFERPOOL BP2
  CLOSE      NO
                                                                      ;
CREATE TYPE 2 UNIQUE INDEX ITSO.IXTEP1
  ON ITSO.CURR_TRADE_EXEC_PR
  (ACCOUNT_ID ASC, ITEM_ISSUE_ID ASC, RULE_SET_SUFFIX_ID ASC,
  TRAN_PROCESSING_DT DESC)
  CLUSTER
  ( PART  1 VALUES(19999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 93
                SECQTY 72
          PCTFREE    20
  , PART  2 VALUES(29999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 86
                SECQTY 72
          PCTFREE    20
  , PART  3 VALUES(39999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 50
                SECQTY 36
          PCTFREE    20
  , PART  4 VALUES(49999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 43
                SECQTY 36
          PCTFREE    20
  , PART  5 VALUES(59999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 64
                SECQTY 36
          PCTFREE    20
  , PART  6 VALUES(69999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 64
                SECQTY 36
          PCTFREE    20
  , PART  7 VALUES(79999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 43
```

```
                SECQTY 36
          PCTFREE    20
 , PART  8 VALUES(89999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 64
                SECQTY 36
          PCTFREE    20
 , PART  9 VALUES(99999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 43
                SECQTY 36
          PCTFREE    20
 )
 BUFFERPOOL BP2
 CLOSE      NO
                                                                             ;
CREATE TYPE 2 UNIQUE INDEX ITSO.IXTMS1
 ON ITSO.STAT_TRAN_MESSAGE
 (ACCT_ID ASC, TRAN_EFF_DT DESC, TRAN_SEQ_ID ASC)
 CLUSTER
 USING STOGROUP ITSIXSG1
       PRIQTY 72
       SECQTY 36
 PCTFREE    20
 BUFFERPOOL BP2
 CLOSE      NO
                                                                             ;
CREATE TYPE 2 UNIQUE INDEX ITSO.IXTRN1
 ON ITSO.TRAN
 (ACCT_ID ASC, TRAN_CYCLE_ID ASC, TRAN_EFF_DT DESC, TRAN_KEY_TM ASC)
 CLUSTER
 ( PART  1 VALUES(10019999)
         USING STOGROUP ITSIXSG1
                PRIQTY 165
                SECQTY 108
          PCTFREE    20
 , PART  2 VALUES(10959999)
         USING STOGROUP ITSIXSG1
                PRIQTY 79
                SECQTY 72
          PCTFREE    20
 , PART  3 VALUES(12009999)
         USING STOGROUP ITSIXSG1
                PRIQTY 115
                SECQTY 72
          PCTFREE    20
 , PART  4 VALUES(13129999)
         USING STOGROUP ITSIXSG1
                PRIQTY 79
```

```
              SECQTY 72
        PCTFREE    20
, PART  5 VALUES(14419999)
       USING STOGROUP ITSIXSG1
              PRIQTY 93
              SECQTY 72
        PCTFREE    20
, PART  6 VALUES(15809999)
       USING STOGROUP ITSIXSG1
              PRIQTY 86
              SECQTY 72
        PCTFREE    20
, PART  7 VALUES(17289999)
       USING STOGROUP ITSIXSG1
              PRIQTY 72
              SECQTY 36
        PCTFREE    20
, PART  8 VALUES(18739999)
       USING STOGROUP ITSIXSG1
              PRIQTY 86
              SECQTY 72
        PCTFREE    20
, PART  9 VALUES(19999999)
       USING STOGROUP ITSIXSG1
              PRIQTY 64
              SECQTY 36
        PCTFREE    20
, PART 10 VALUES(21599999)
       USING STOGROUP ITSIXSG1
              PRIQTY 122
              SECQTY 72
        PCTFREE    20
, PART 11 VALUES(22759999)
       USING STOGROUP ITSIXSG1
              PRIQTY 72
              SECQTY 36
        PCTFREE    20
, PART 12 VALUES(24149999)
       USING STOGROUP ITSIXSG1
              PRIQTY 86
              SECQTY 72
        PCTFREE    20
, PART 13 VALUES(25719999)
       USING STOGROUP ITSIXSG1
              PRIQTY 86
              SECQTY 72
        PCTFREE    20
, PART 14 VALUES(27159999)
       USING STOGROUP ITSIXSG1
```

```
              PRIQTY 72
              SECQTY 36
        PCTFREE    20
, PART 15 VALUES(28509999)
         USING STOGROUP ITSIXSG1
              PRIQTY 72
              SECQTY 36
        PCTFREE    20
, PART 16 VALUES(29999999)
         USING STOGROUP ITSIXSG1
              PRIQTY 86
              SECQTY 72
        PCTFREE    20
, PART 17 VALUES(31359999)
         USING STOGROUP ITSIXSG1
              PRIQTY 108
              SECQTY 72
        PCTFREE    20
, PART 18 VALUES(32559999)
         USING STOGROUP ITSIXSG1
              PRIQTY 93
              SECQTY 72
        PCTFREE    20
, PART 19 VALUES(34049999)
         USING STOGROUP ITSIXSG1
              PRIQTY 72
              SECQTY 36
        PCTFREE    20
, PART 20 VALUES(35439999)
         USING STOGROUP ITSIXSG1
              PRIQTY 64
              SECQTY 36
        PCTFREE    20
, PART 21 VALUES(36809999)
         USING STOGROUP ITSIXSG1
              PRIQTY 64
              SECQTY 36
        PCTFREE    20
, PART 22 VALUES(38349999)
         USING STOGROUP ITSIXSG1
              PRIQTY 57
              SECQTY 36
        PCTFREE    20
, PART 23 VALUES(39999999)
         USING STOGROUP ITSIXSG1
              PRIQTY 64
              SECQTY 36
        PCTFREE    20
, PART 24 VALUES(41649999)
```

```
        USING STOGROUP ITSIXSG1
               PRIQTY 108
               SECQTY 72
         PCTFREE    20
, PART 25 VALUES(43139999)
        USING STOGROUP ITSIXSG1
               PRIQTY 64
               SECQTY 36
         PCTFREE    20
, PART 26 VALUES(44659999)
        USING STOGROUP ITSIXSG1
               PRIQTY 57
               SECQTY 36
         PCTFREE    20
, PART 27 VALUES(46179999)
        USING STOGROUP ITSIXSG1
               PRIQTY 57
               SECQTY 36
         PCTFREE    20
, PART 28 VALUES(47269999)
        USING STOGROUP ITSIXSG1
               PRIQTY 57
               SECQTY 36
         PCTFREE    20
, PART 29 VALUES(48679999)
        USING STOGROUP ITSIXSG1
               PRIQTY 57
               SECQTY 36
         PCTFREE    20
, PART 30 VALUES(49999999)
        USING STOGROUP ITSIXSG1
               PRIQTY 57
               SECQTY 36
         PCTFREE    20
, PART 31 VALUES(51429999)
        USING STOGROUP ITSIXSG1
               PRIQTY 100
               SECQTY 72
         PCTFREE    20
, PART 32 VALUES(53009999)
        USING STOGROUP ITSIXSG1
               PRIQTY 86
               SECQTY 72
         PCTFREE    20
, PART 33 VALUES(54329999)
        USING STOGROUP ITSIXSG1
               PRIQTY 57
               SECQTY 36
         PCTFREE    20
```

```
, PART 34 VALUES(55859999)
        USING STOGROUP ITSIXSG1
               PRIQTY 57
               SECQTY 36
         PCTFREE    20
, PART 35 VALUES(57299999)
        USING STOGROUP ITSIXSG1
               PRIQTY 57
               SECQTY 36
         PCTFREE    20
, PART 36 VALUES(58809999)
        USING STOGROUP ITSIXSG1
               PRIQTY 57
               SECQTY 36
         PCTFREE    20
, PART 37 VALUES(59999999)
        USING STOGROUP ITSIXSG1
               PRIQTY 43
               SECQTY 36
         PCTFREE    20
, PART 38 VALUES(61759999)
        USING STOGROUP ITSIXSG1
               PRIQTY 100
               SECQTY 72
         PCTFREE    20
, PART 39 VALUES(62999999)
        USING STOGROUP ITSIXSG1
               PRIQTY 64
               SECQTY 36
         PCTFREE    20
, PART 40 VALUES(64609999)
        USING STOGROUP ITSIXSG1
               PRIQTY 57
               SECQTY 36
         PCTFREE    20
, PART 41 VALUES(65979999)
        USING STOGROUP ITSIXSG1
               PRIQTY 57
               SECQTY 36
         PCTFREE    20
, PART 42 VALUES(67279999)
        USING STOGROUP ITSIXSG1
               PRIQTY 57
               SECQTY 36
         PCTFREE    20
, PART 43 VALUES(68739999)
        USING STOGROUP ITSIXSG1
               PRIQTY 72
               SECQTY 36
```

```
         PCTFREE    20
, PART 44 VALUES(69999999)
        USING STOGROUP ITSIXSG1
               PRIQTY 57
               SECQTY 36
         PCTFREE    20
, PART 45 VALUES(71409999)
        USING STOGROUP ITSIXSG1
               PRIQTY 86
               SECQTY 72
         PCTFREE    20
, PART 46 VALUES(72839999)
        USING STOGROUP ITSIXSG1
               PRIQTY 64
               SECQTY 36
         PCTFREE    20
, PART 47 VALUES(74309999)
        USING STOGROUP ITSIXSG1
               PRIQTY 57
               SECQTY 36
         PCTFREE    20
, PART 48 VALUES(75669999)
        USING STOGROUP ITSIXSG1
               PRIQTY 57
               SECQTY 36
         PCTFREE    20
, PART 49 VALUES(77249999)
        USING STOGROUP ITSIXSG1
               PRIQTY 57
               SECQTY 36
         PCTFREE    20
, PART 50 VALUES(78439999)
        USING STOGROUP ITSIXSG1
               PRIQTY 43
               SECQTY 36
         PCTFREE    20
, PART 51 VALUES(79999999)
        USING STOGROUP ITSIXSG1
               PRIQTY 43
               SECQTY 36
         PCTFREE    20
, PART 52 VALUES(81399999)
        USING STOGROUP ITSIXSG1
               PRIQTY 93
               SECQTY 72
         PCTFREE    20
, PART 53 VALUES(82879999)
        USING STOGROUP ITSIXSG1
               PRIQTY 64
```

```
                SECQTY 36
          PCTFREE    20
, PART 54 VALUES(84369999)
         USING STOGROUP ITSIXSG1
                PRIQTY 57
                SECQTY 36
          PCTFREE    20
, PART 55 VALUES(86119999)
         USING STOGROUP ITSIXSG1
                PRIQTY 64
                SECQTY 36
          PCTFREE    20
, PART 56 VALUES(87539999)
         USING STOGROUP ITSIXSG1
                PRIQTY 57
                SECQTY 36
          PCTFREE    20
, PART 57 VALUES(88639999)
         USING STOGROUP ITSIXSG1
                PRIQTY 64
                SECQTY 36
          PCTFREE    20
, PART 58 VALUES(89999999)
         USING STOGROUP ITSIXSG1
                PRIQTY 57
                SECQTY 36
          PCTFREE    20
, PART 59 VALUES(91389999)
         USING STOGROUP ITSIXSG1
                PRIQTY 86
                SECQTY 72
          PCTFREE    20
, PART 60 VALUES(92989999)
         USING STOGROUP ITSIXSG1
                PRIQTY 72
                SECQTY 36
          PCTFREE    20
, PART 61 VALUES(94569999)
         USING STOGROUP ITSIXSG1
                PRIQTY 57
                SECQTY 36
          PCTFREE    20
, PART 62 VALUES(96499999)
         USING STOGROUP ITSIXSG1
                PRIQTY 57
                SECQTY 36
          PCTFREE    20
, PART 63 VALUES(99999999)
         USING STOGROUP ITSIXSG1
```

```
                  PRIQTY 57
                  SECQTY 36
            PCTFREE    20
  )
  BUFFERPOOL BP2
  CLOSE      NO
                                                                       ;
CREATE TYPE 2 UNIQUE INDEX ITSO.IXTTA1
  ON ITSO.TRAN_ASSCN
  (ACCT_ID ASC, TRAN_CYCLE_ID ASC, TRAN_EFF_DT DESC, TRAN_KEY_TM ASC,
  RELT_TRAN_KEY_TM ASC)
  CLUSTER
  ( PART  1 VALUES(39999999)
           USING STOGROUP ITSIXSG1
                  PRIQTY 568
                  SECQTY 288
            PCTFREE    30
  , PART  2 VALUES(69999999)
           USING STOGROUP ITSIXSG1
                  PRIQTY 460
                  SECQTY 252
            PCTFREE    30
  , PART  3 VALUES(99999999)
           USING STOGROUP ITSIXSG1
                  PRIQTY 403
                  SECQTY 216
            PCTFREE    30
  )
  BUFFERPOOL BP2
  CLOSE      NO
                                                                       ;
CREATE TYPE 2  INDEX ITSO.IXTTA2
  ON ITSO.TRAN_ASSCN
  (ACCT_ID ASC, TRAN_CYCLE_ID ASC, TRAN_EFF_DT DESC, RELT_TRAN_KEY_TM
  ASC)
  USING STOGROUP ITSIXSG1
        PRIQTY 1310
        SECQTY 684
  PCTFREE    30
  BUFFERPOOL BP2
  CLOSE      NO
                                                                       ;
CREATE TYPE 2 UNIQUE INDEX ITSO.IXUIR1
  ON ITSO.UNIQ_ID_RGSTR
  (ACCT_ID ASC, RGSTR_CREATE_TS ASC)
  CLUSTER
  ( PART  1 VALUES(19999999)
           USING STOGROUP ITSIXSG1
                  PRIQTY 295
```

```
                  SECQTY 180
            PCTFREE    15
  , PART  2 VALUES(29999999)
           USING STOGROUP ITSIXSG1
                  PRIQTY 216
                  SECQTY 108
            PCTFREE    15
  , PART  3 VALUES(39999999)
           USING STOGROUP ITSIXSG1
                  PRIQTY 187
                  SECQTY 108
            PCTFREE    15
  , PART  4 VALUES(49999999)
           USING STOGROUP ITSIXSG1
                  PRIQTY 165
                  SECQTY 108
            PCTFREE    15
  , PART  5 VALUES(59999999)
           USING STOGROUP ITSIXSG1
                  PRIQTY 158
                  SECQTY 108
            PCTFREE    15
  , PART  6 VALUES(69999999)
           USING STOGROUP ITSIXSG1
                  PRIQTY 165
                  SECQTY 108
            PCTFREE    15
  , PART  7 VALUES(79999999)
           USING STOGROUP ITSIXSG1
                  PRIQTY 158
                  SECQTY 108
            PCTFREE    15
  , PART  8 VALUES(89999999)
           USING STOGROUP ITSIXSG1
                  PRIQTY 165
                  SECQTY 108
            PCTFREE    15
  , PART  9 VALUES(99999999)
           USING STOGROUP ITSIXSG1
                  PRIQTY 129
                  SECQTY 72
            PCTFREE    15
  )
  BUFFERPOOL BP2
  CLOSE      NO
                                                                     ;
---====================== FOREIGN KEY DEFINITIONS =====================

  ALTER  TABLE ITSO.LEGACY_FUND_TRAN  ADD FOREIGN KEY FTRNLFC1
```

```
    (ACCT_ID, TRAN_CYCLE_ID, TRAN_EFF_DT, TRAN_KEY_TM)
    REFERENCES ITSO.TRAN
    ON DELETE CASCADE
                                                       ;
  ALTER  TABLE ITSO.LEGACY_ISSUE_TRAN  ADD FOREIGN KEY FTRNLIC1
    (ACCT_ID, TRAN_CYCLE_ID, TRAN_EFF_DT, TRAN_KEY_TM)
    REFERENCES ITSO.TRAN
    ON DELETE CASCADE
                                                       ;
  ALTER  TABLE ITSO.LEGACY_TRADE_TRAN  ADD FOREIGN KEY FTRNLTR1
    (ACCT_ID, TRAN_CYCLE_ID, TRAN_EFF_DT, TRAN_KEY_TM)
    REFERENCES ITSO.TRAN
    ON DELETE CASCADE
                                                       ;
  ALTER  TABLE ITSO.SMA_EXTD_JRNL  ADD FOREIGN KEY FSJLSJE1
    (ACCT_ID, AUDIT_ADD_TS)
    REFERENCES ITSO.SMA_BASE_JRNL
                                                       ;
  ALTER  TABLE ITSO.TRAN_ASSCN  ADD FOREIGN KEY FTRNTTA1
    (ACCT_ID, TRAN_CYCLE_ID, TRAN_EFF_DT, TRAN_KEY_TM)
    REFERENCES ITSO.TRAN
    ON DELETE CASCADE
                                                       ;
  ALTER  TABLE ITSO.TRAN_ASSCN  ADD FOREIGN KEY FTRNTTA2
    (ACCT_ID, TRAN_CYCLE_ID, TRAN_EFF_DT, RELT_TRAN_KEY_TM)
    REFERENCES ITSO.TRAN
    ON DELETE CASCADE
                                                       ;
  ALTER  TABLE ITSO.TRAN_LGCY_SEC_DESC  ADD FOREIGN KEY FTRNLSD1
    (ACCT_ID, TRAN_CYCLE_ID, TRAN_EFF_DT, TRAN_KEY_TM)
    REFERENCES ITSO.TRAN
    ON DELETE CASCADE
                                                       ;
COMMIT WORK;
```

G

Communicating with Q replication programs in z/OS

In this appendix we provide an overview of some of the methods available for communicating with Q replication programs in a z/OS environment.

Introduction

WebSphere Information Integrator provides a number of mechanisms for changing Q replication parameters, refreshing configuration and parameters in executing programs, and stopping and starting Q replication processes. This section provides an overview of the main mechanisms, as follows:

- Replication Center script generation and execution
- Replication Center command processing
- Unix System Services commands via OMVS
- USS commands executed via batch JCL
- Z/OS system commands

> **Note:** Not every mechanism provides the capability to perform all the desired functions to configure and manage a Q replication environment on the z/OS platform.

Replication Center script generation and execution

Any Replication Center option that generates an SQL script can be executed from the Replication Center, even if Data Administration Server (DAS) is not installed on the workstation. This includes all the configuration scripts produced in this book.

In addition, scripts may be used to perform management operations such as modify parameters and introduce signals that trigger operational functions. A number of signals are available including CAPSTART, CAPSTOP, LOADDONE, STOP, and REINIT_SUB[1]. These signals are inserted into the xxxx.IBMQREP_SIGNAL table via SQL INSERT statements, which get written to the DB2 log. The Q Capture program encounters these log records when scanning the DB2 log and triggers the appropriate action based on the signal.

Replication Center command processing

Q replication commands include the ability to start or modify a Q replication program such as Q Capture and Q Apply.

Q replication commands to execute programs include:

- **`asnqcap`**: Capture program
- **`asnqapp`**: Apply program
- **`asnmon`**: Monitor program

[1] To re-initialize a subscription after changing parameter values.

- **asntdiff**: Table compare utility
- **asntrep**: Table repair utility
- **asntrc**: Trace facility

The Q replication commands that communicate with executing Q replication programs in order to modify parameters or behavior include:

- **asnqccmd**: Communicate with the Capture program
- **asnqacmd**: Communicate with the Apply program
- **asnmcmd**: Communicate with the Monitor program

Note: Q replication commands can only be executed from the Replication Center if DAS is installed. If not, these commands must be executed from USS via OMVS, batch JCL, or MVS started task, as described in the following sections.

Unix System Services commands via OMVS

asnqccmd, **asnqacmd**, and **asnmcmd** are all USS commands that can be executed from the OMVS command prompt.

To get to OMVS from an ISPF TSO session:

1. Go to the command processor (=6).
2. Type `OMVS`, which should give you the Unix OSHELL processor.

One can edit and save the Q replication commands from the Replication Center as files in this Unix System Services shell, or execute them directly from the OSHELL command prompt.

The OMVS SubCmd QUIT will return the session to TSO.

Example G-1 shows the sample output of the **asnqccmd** command issued at the OMVS prompt.

Example: G-1 Sample asnqccmd command output

```
asnqccmd CAPTURE_SERVER=D8G1 CAPTURE_SCHEMA=\"ITSO\" status

   2004-12-13-11.45.29.916038 ASN0520I  "AsnQCcmd" : "ITSO" : "Initial" : The STATU
   S command response: "HoldLThread" thread is in the "is waiting" state.
   2004-12-13-11.45.29.943592 ASN0520I  "AsnQCcmd" : "ITSO" : "Initial" : The STATU
   S command response: "AdminThread" thread is in the "is resting" state.
   2004-12-13-11.45.29.945592 ASN0520I  "AsnQCcmd" : "ITSO" : "Initial" : The STATU
   S command response: "PruneThread" thread is in the "is resting" state.
   2004-12-13-11.45.29.947921 ASN0520I  "AsnQCcmd" : "ITSO" : "Initial" : The STATU
```

```
S command response: "WorkerThread" thread is in the "is resting" state.
```

USS commands via batch JCL

The batch program BPXBATCH can be used to execute USS commands via JCL. The STDOUT (standard output file) must be a USS file. A second step can be added to the JCL to print the contents of the USS output file, resulting in a job that resembles an MVS job and its output. Some Q replication users may be more comfortable with this “strictly” MVS approach to command processing.

Example G-2 shows example batch JCL for the **asnqccmd** command, while Example G-3 shows its corresponding output.

Example: G-2 Sample asnqccmd using batch JCL

```
//ASNQCCMD JOB NOTIFY=&SYSUID,
//         MSGCLASS=H,MSGLEVEL=(1,1),
//         REGION=0M,TIME=NOLIMIT
/*JOBPARM S=SC53
//*****************************************************************/
//*                                                               */
//*    IBM DB2 Information Integrator Replication Version 8.2    */
//*    for z/OS (5655-I60)                                        */
//*                                                               */
//*    Sample JCL communicate with Qcapture using                 */
//*    asnqccmd via (BPXBATCH)                                    */
//*    capture_server is the subsystem name                       */
//*    logstsout will go /u/repluser/stdout                       */
//*    User running this must have a USS profile                  */
//*    and repluser directory with write permission               */
//*****************************************************************/
//STOPC1   EXEC PGM=BPXBATCH,
//      PARM='sh cd; . .profile;asnqccmd capture_server=D8G1
//             capture_schema=itso status '
//SYSPRINT DD  SYSOUT=*
//SYSTSPRT DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//STDOUT   DD PATH='/tmp/asnqccmd.stdout',
//         PATHDISP=(KEEP,KEEP),
//         PATHOPTS=(OWRONLY,OCREAT),
//         PATHMODE=(SIRWXU,SIRGRP,SIROTH)
//*
//SND2SYS  EXEC PGM=IKJEFT01,DYNAMNBR=20,REGION=64M
//*
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
```

```
//*
//INPUT    DD  PATH='/tmp/asnqccmd.stdout',PATHOPTS=(ORDONLY)
//OUTPUT   DD  SYSOUT=*,DCB=(LRECL=255,RECFM=F)
//*
//SYSTSIN  DD  *
 OCOPY  INDD(INPUT) OUTDD(OUTPUT) TEXT PATHOPTS(OVERRIDE)
//
```

Example: G-3   asnqccmd output of batch JCL shown in Example G-2

```
1                   J E S 2  J O B  L O G  --  S Y S T E M  S C 5 3  --  N O D E  W T S C P L X
1
0
 20.24.24 JOB04583 ---- FRIDAY,    19 NOV 2004 ----
 20.24.24 JOB04583  IRR010I  USERID QREPADM IS ASSIGNED TO THIS JOB.
 20.24.24 JOB04583  ICH70001I QREPADM  LAST ACCESS AT 20:23:02 ON FRIDAY, NOVEMBER 19, 2004
 20.24.24 JOB04583  $HASP373 ASNQCCMD STARTED - INIT A    - CLASS A - SYS SC53
 20.24.24 JOB04583  IEF403I ASNQCCMD - STARTED - TIME=20.24.24 - ASID=0039 - SC53
 20.24.24 JOB04583  -                                        --TIMINGS (MINS.)--
----PAGING COUNTS---
 20.24.24 JOB04583  -JOBNAME  STEPNAME PROCSTEP    RC   EXCP    CPU    SRB  CLOCK   SERV  PG
PAGE   SWAP    VIO SWAPS
 20.24.24 JOB04583  -ASNQCCMD         STOPC1      00     61    .00    .00     .0   2258   0
0      0      0     0
 20.24.24 JOB04583  -                                        --TIMINGS (MINS.)--
----PAGING COUNTS---
 20.24.24 JOB04583  -JOBNAME  STEPNAME PROCSTEP    RC   EXCP    CPU    SRB  CLOCK   SERV  PG
PAGE   SWAP    VIO SWAPS
 20.24.24 JOB04583  -ASNQCCMD         *OMVSEX     00     25    .00    .00     .0   2647   0
0      0      0     0
 20.24.36 JOB04583  -                                        --TIMINGS (MINS.)--
----PAGING COUNTS---
 20.24.36 JOB04583  -JOBNAME  STEPNAME PROCSTEP    RC   EXCP    CPU    SRB  CLOCK   SERV  PG
PAGE   SWAP    VIO SWAPS
 20.24.36 JOB04583  -ASNQCCMD         *OMVSEX     00    502    .00    .00     .1  15245   0
0      0      0     0
 20.24.36 JOB04583  -ASNQCCMD         SND2SYS     00     65    .00    .00     .0  10336   0
0      0      0     0
 20.24.36 JOB04583  IEF404I ASNQCCMD - ENDED - TIME=20.24.36 - ASID=0039 - SC53
 20.24.36 JOB04583  -ASNQCCMD ENDED.  NAME-                    TOTAL CPU TIME=   .00  TOTAL
ELAPSED TIME=    .2
 20.24.36 JOB04583  $HASP395 ASNQCCMD ENDED
0------ JES2 JOB STATISTICS ------
-  19 NOV 2004 JOB EXECUTION DATE
-           40 CARDS READ
-          123 SYSOUT PRINT RECORDS
-            0 SYSOUT PUNCH RECORDS
-            8 SYSOUT SPOOL KBYTES
-         0.20 MINUTES EXECUTION TIME
```

```
        1 //ASNQCCMD JOB NOTIFY=&SYSUID,                                          JOB04583
          //          MSGCLASS=H,MSGLEVEL=(1,1),
          //          REGION=0M,TIME=NOLIMIT
          /*JOBPARM S=SC53
          //*****************************************************************/
          //*                                                               */
          //*    IBM DB2 Information Integrator Replication Version 8.2    */
          //*    for z/OS (5655-I60)                                        */
          //*                                                               */
          //*    Sample JCL communicate with Qcapture using                 */
          //*    asnqccmd via (BPXBATCH)                                    */
          //*    capture_server is the subsystem name                       */
          //*    logstsout will go /u/repluser/stdout                       */
          //*    User running this must have a USS profile                  */
          //*    and repluser directory with write permission               */
          //*****************************************************************/
          IEFC653I SUBSTITUTION JCL -
NOTIFY=QREPADM,MSGCLASS=H,MSGLEVEL=(1,1),REGION=0M,TIME=NOLIMIT
        2 //STOPC1   EXEC PGM=BPXBATCH,
          //      PARM='sh cd; . .profile;asnqccmd capture_server=D8G1
          //             capture_schema=itso status '
        3 //SYSPRINT DD  SYSOUT=*
        4 //SYSTSPRT DD  SYSOUT=*
        5 //SYSUDUMP DD  SYSOUT=*
          //*STDOUT   DD  SYSOUT=*
          //*STDOUT   DD  DSN=QREPADM.ASNQCCMD.OUT,DISP=(MOD),
          //*             DCB=(RECFM=VB,LRECL=1028,BLKSIZE=6144)
        6 //STDOUT   DD PATH='/tmp/asnqccmd.stdout',
          //          PATHDISP=(KEEP,KEEP),
          //          PATHOPTS=(OWRONLY,OCREAT),
          //          PATHMODE=(SIRWXU,SIRGRP,SIROTH)
          //*
        7 //SND2SYS  EXEC PGM=IKJEFT01,DYNAMNBR=20,REGION=64M
          //*
        8 //SYSTSPRT DD SYSOUT=*
        9 //SYSPRINT DD SYSOUT=*
          //*
       10 //INPUT    DD  PATH='/tmp/asnqccmd.stdout',PATHOPTS=(ORDONLY)
       11 //OUTPUT   DD  SYSOUT=*,DCB=(LRECL=255,RECFM=F)
          //*
       12 //SYSTSIN  DD  *
 ICH70001I QREPADM  LAST ACCESS AT 20:23:02 ON FRIDAY, NOVEMBER 19, 2004
 IEF236I ALLOC. FOR ASNQCCMD STOPC1
 IEF237I JES2 ALLOCATED TO SYSPRINT
 IEF237I JES2 ALLOCATED TO SYSTSPRT
 IEF237I JES2 ALLOCATED TO SYSUDUMP
 IGD103I SMS ALLOCATED TO DDNAME STDOUT
 IEF142I ASNQCCMD STOPC1 - STEP WAS EXECUTED - COND CODE 0000
 IEF285I   QREPADM.ASNQCCMD.JOB04583.D0000102.?         SYSOUT
```

```
 IEF285I   QREPADM.ASNQCCMD.JOB04583.D0000103.?         SYSOUT
 IEF285I   QREPADM.ASNQCCMD.JOB04583.D0000104.?         SYSOUT
 IGD104I HFS FILE WAS RETAINED, DDNAME IS (STDOUT  )
 FILENAME IS (/tmp/asnqccmd.stdout)
 IEF373I STEP/STOPC1  /START 2004324.2024
 IEF374I STEP/STOPC1  /STOP  2004324.2024 CPU    0MIN 00.01SEC SRB    0MIN 00.00SEC VIRT    40K
SYS   260K EXT      12K SYS   10600K
 IEF142I ASNQCCMD *OMVSEX - STEP WAS EXECUTED - COND CODE 0000
 IEF373I STEP/*OMVSEX /START 2004324.2024
 IEF374I STEP/*OMVSEX /STOP  2004324.2024 CPU    0MIN 00.01SEC SRB    0MIN 00.00SEC VIRT     8K
SYS   264K EXT     436K SYS   10584K
 IEF142I ASNQCCMD *OMVSEX - STEP WAS EXECUTED - COND CODE 0000
 IEF373I STEP/*OMVSEX /START 2004324.2024
 IEF374I STEP/*OMVSEX /STOP  2004324.2024 CPU    0MIN 00.06SEC SRB    0MIN 00.00SEC VIRT     8K
SYS   260K EXT     464K SYS   10588K
 IEF236I ALLOC. FOR ASNQCCMD SND2SYS
 IEF237I JES2 ALLOCATED TO SYSTSPRT
 IEF237I JES2 ALLOCATED TO SYSPRINT
 IGD103I SMS ALLOCATED TO DDNAME INPUT
 IEF237I JES2 ALLOCATED TO OUTPUT
 IEF237I JES2 ALLOCATED TO SYSTSIN
 IEF142I ASNQCCMD SND2SYS - STEP WAS EXECUTED - COND CODE 0000
 IEF285I   QREPADM.ASNQCCMD.JOB04583.D0000105.?         SYSOUT
 IEF285I   QREPADM.ASNQCCMD.JOB04583.D0000106.?         SYSOUT
 IGD104I HFS FILE WAS RETAINED, DDNAME IS (INPUT   )
 FILENAME IS (/tmp/asnqccmd.stdout)
 IEF285I   QREPADM.ASNQCCMD.JOB04583.D0000107.?         SYSOUT
 IEF285I   QREPADM.ASNQCCMD.JOB04583.D0000101.?         SYSIN
 IEF373I STEP/SND2SYS /START 2004324.2024
 IEF374I STEP/SND2SYS /STOP  2004324.2024 CPU    0MIN 00.05SEC SRB    0MIN 00.00SEC VIRT   128K
SYS   352K EXT     156K SYS   10880K
 IEF375I  JOB/ASNQCCMD/START 2004324.2024
 IEF376I  JOB/ASNQCCMD/STOP  2004324.2024 CPU    0MIN 00.13SEC SRB    0MIN 00.00SEC
1READY
  OCOPY  INDD(INPUT) OUTDD(OUTPUT) TEXT PATHOPTS(OVERRIDE)
 READY
 END
 --> PATH set for JAVA JDK-/usr/lpp/java/J1.4/bin:/bin:.
 --> JAVA_HOME set for JAVA Servlet support-/usr/lpp/java/J1.4
 Path set to: /usr/lpp/java/J1.4/bin:/bin:.
 New Classpath set to: :.:/SC53/imsv9/imsjava91/imsjava.jar:/u/jouko2
 New LIBPATH set to:
/SC53/imsv9/imsjava91:/usr/lpp/java/J1.4/bin/classic:/usr/lpp/java/J1.4/bin:
 PATH is /usr/lpp/java/J1.4/bin:/bin:.:/u/qrepadm:
 STEPLIB is ASN.V8R2M0.SASNLOAD:DB8G8.SDSNLOAD
 PATH is /usr/lpp/java/J1.4/bin:/bin:.:/u/qrepadm::/u/qrepadm:
 STEPLIB is ASN.V8R2M0.SASNLOAD:DB8G8.SDSNLOAD
 2004-11-19-20.24.28.430215 ASN8999D  "N/A" : "N/A" : "Initial" : thread name = Initial, tid =
0x101331d000000000, tcb = 006E3298
```

```
 2004-11-19-20.24.36.453335 ASN0506E  "AsnQCcmd" : "ITSO" : "Initial" : The command was not
processed. The "Q Capture" program is presumed down.
 state.
 2004-11-19-16.38.44.414805 ASN0520I  "AsnQCcmd" : "ITSO" : "Initial" : The STATUS command
response: "AdminThread" thread is in the "is resting" state.
 2004-11-19-16.38.44.418158 ASN0520I  "AsnQCcmd" : "ITSO" : "Initial" : The STATUS command
response: "PruneThread" thread is in the "is resting" state.
 2004-11-19-16.38.44.421808 ASN0520I  "AsnQCcmd" : "ITSO" : "Initial" : The STATUS command
response: "WorkerThread" thread is in the "is resting" state.
```

Z/OS system commands

If the Q replication programs are set up in MVS as started tasks, and the JCL for them placed in the SYS1.PROCLIB, then these Q replication programs can be started and stopped via the system commands shown in Example G-4 and Example G-5.

Example: G-4 Starting the Q Capture program

```
/s ITSOCAP
```

Example: G-5 Stopping the Q Capture programs

```
/f ITSOCAP,stop
```

H

Cataloging remote database servers

In this appendix we describe the process for cataloging remote database servers in the Replication Center (RC).

Cataloging remote database servers in the RC

The RC is typically used to configure and manage an SQL or Q Replication environment because of its ease-of-use GUI interface. The RC is also most likely to be installed on a Windows machine that is separate from the database servers that are to be part of the replication environment. In order for the RC to be aware of the remote database servers of interest, they need to be cataloged in the RC, in addition to them having to be cataloged in the Control Center.

Note: There is no command equivalent for cataloging the remote database servers in the Replication Center.

The following screens describe the process for cataloging a remote database in the RC. Figure H-1 shows the main screen of the RC.

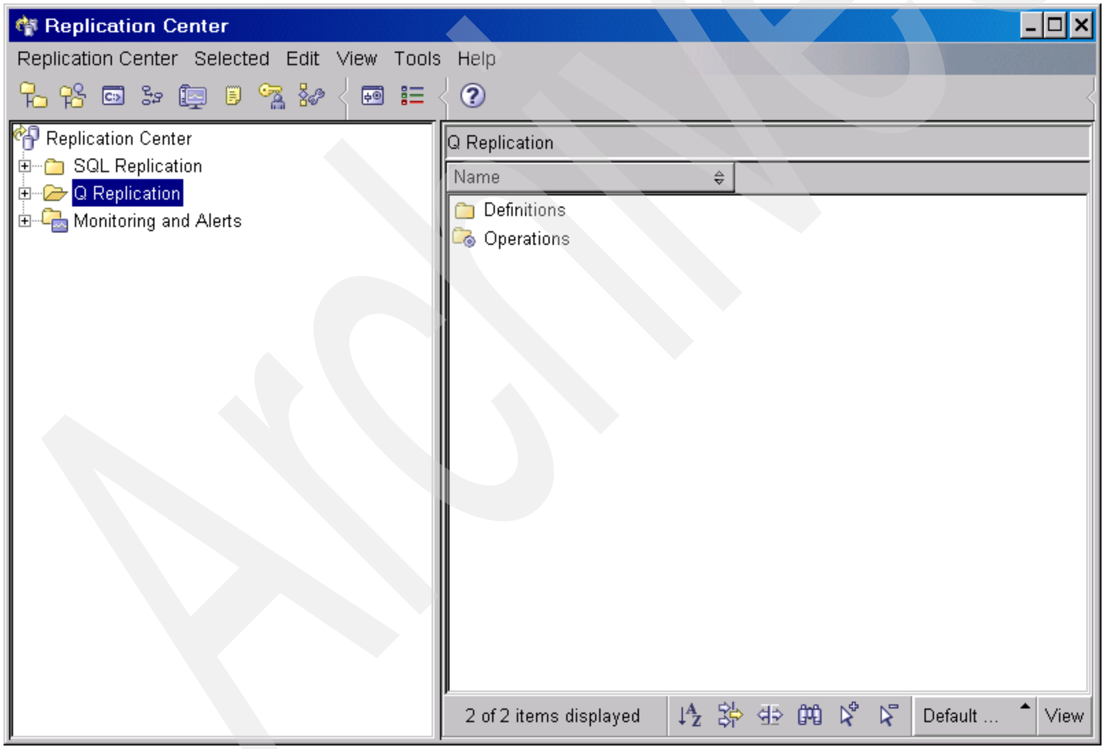


Figure H-1 Replication Center main screen

Click the **Replication Center** tab and select **Manage Passwords and Connectivity...** (as shown in Figure H-2) to proceed to Figure H-3 to add the servers.

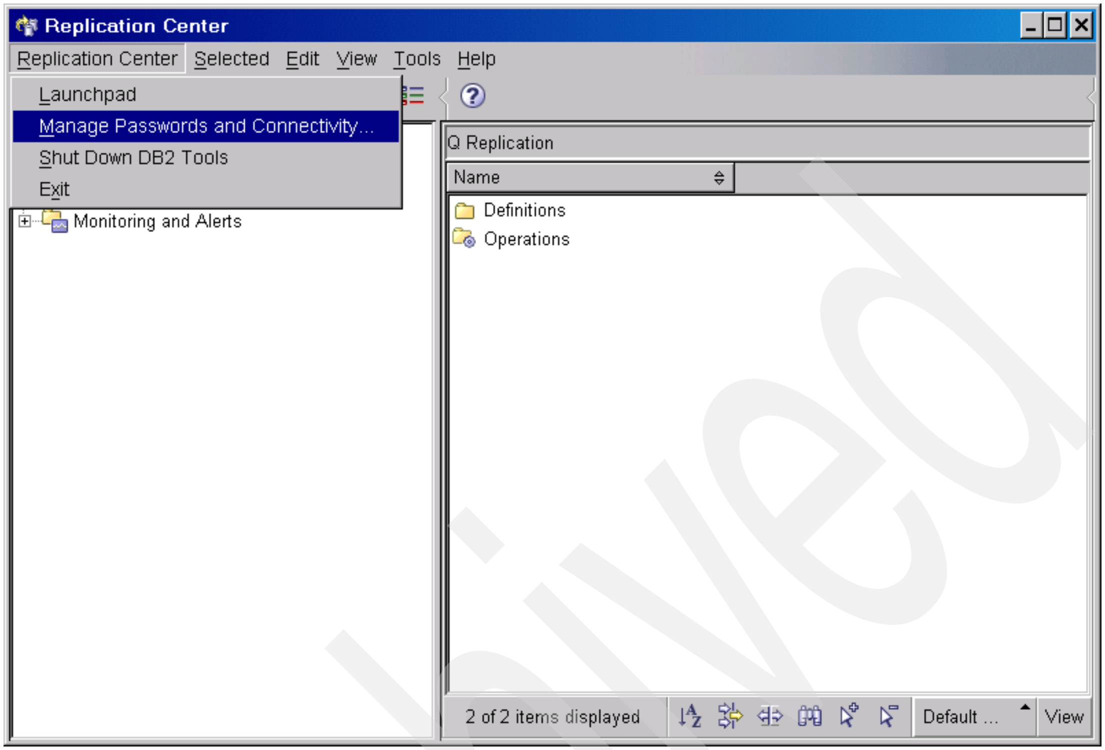


Figure H-2 Select Manage Passwords and Connectivity

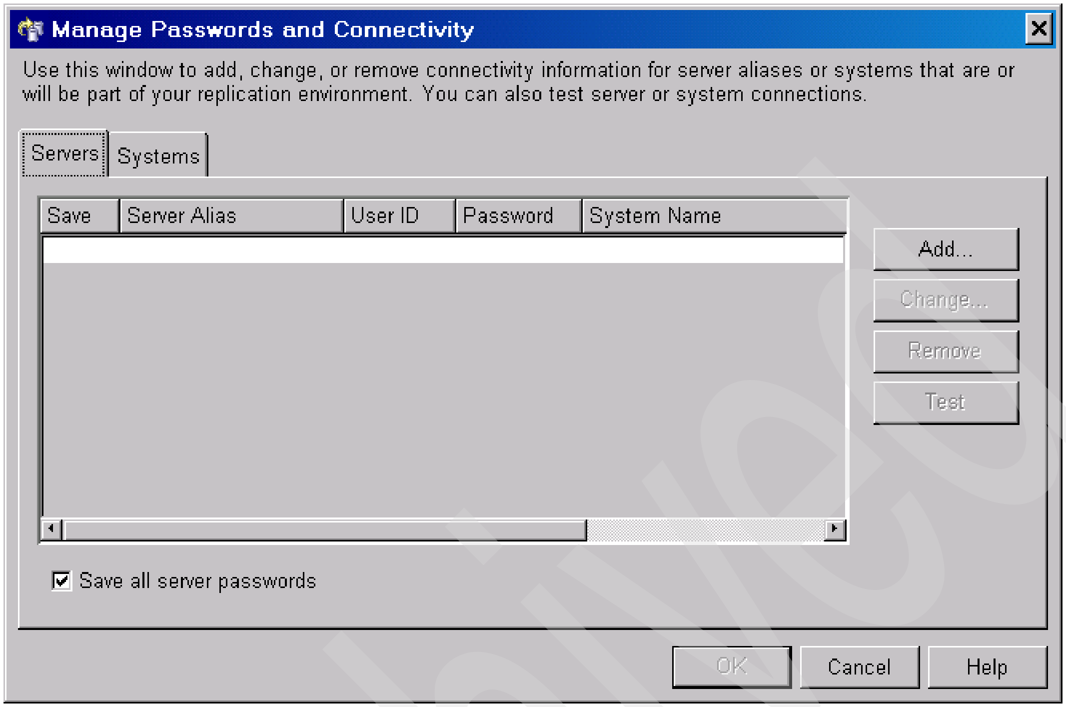


Figure H-3 Manage Passwords and Connectivity

Figure H-3 shows an empty list of servers under the Servers tab. Click **Add...** to proceed to Figure H-4 to add server information.

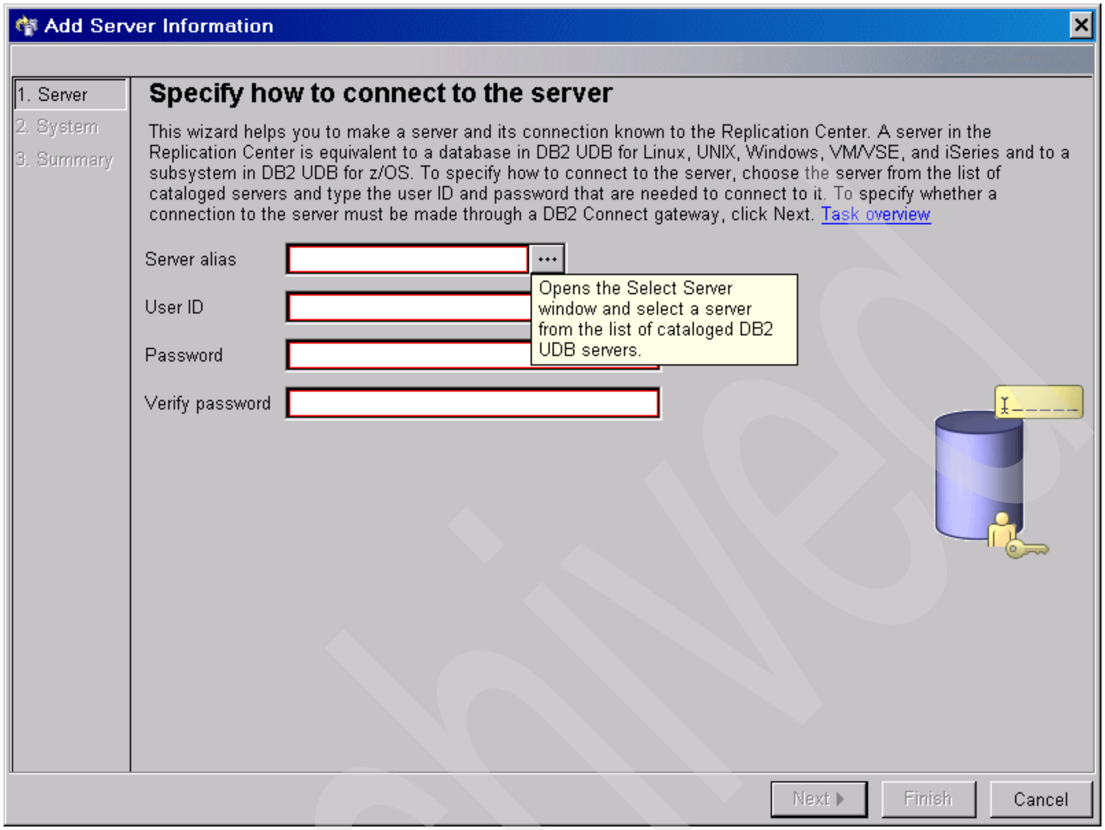


Figure H-4 Specify how to connect to the server

In Figure H-4, click the **...** button to obtain a list of potential candidates for selection, as shown in Figure H-5. The list of servers displayed in Figure H-5 corresponds to all the databases cataloged in the Control Center.

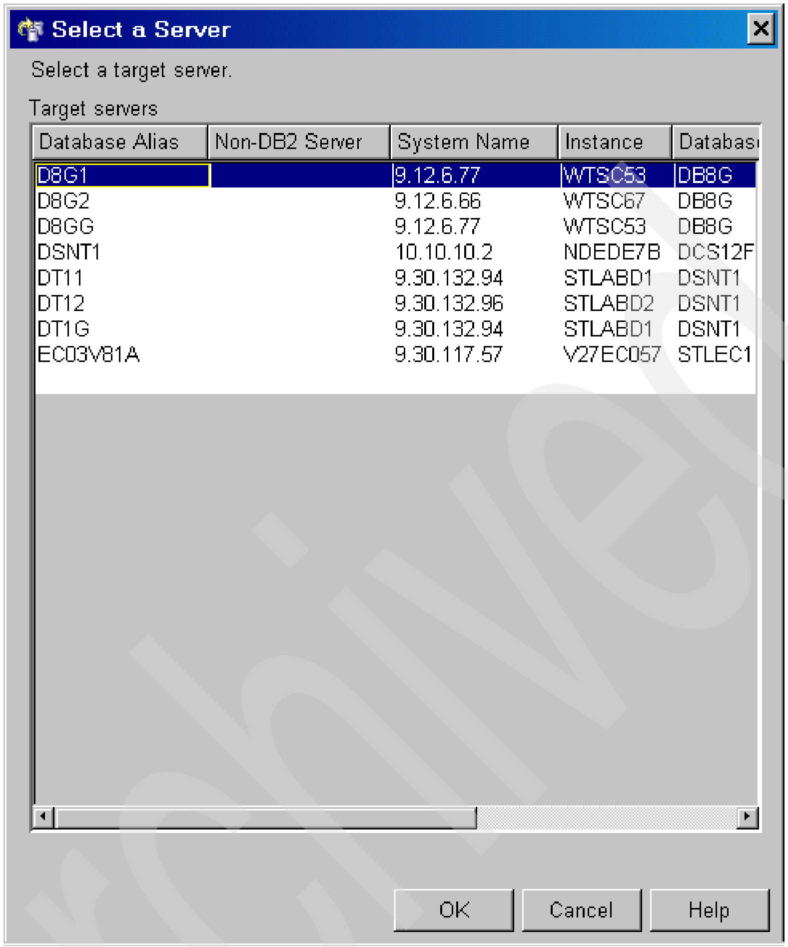


Figure H-5 Select a Server

Select the server of interest, as shown in Figure H-5. In this case, it happens to be a DB2 for z/OS database named D8G1 in a DB8G subsystem. Click **OK** to proceed to Figure H-6 to add connection information.

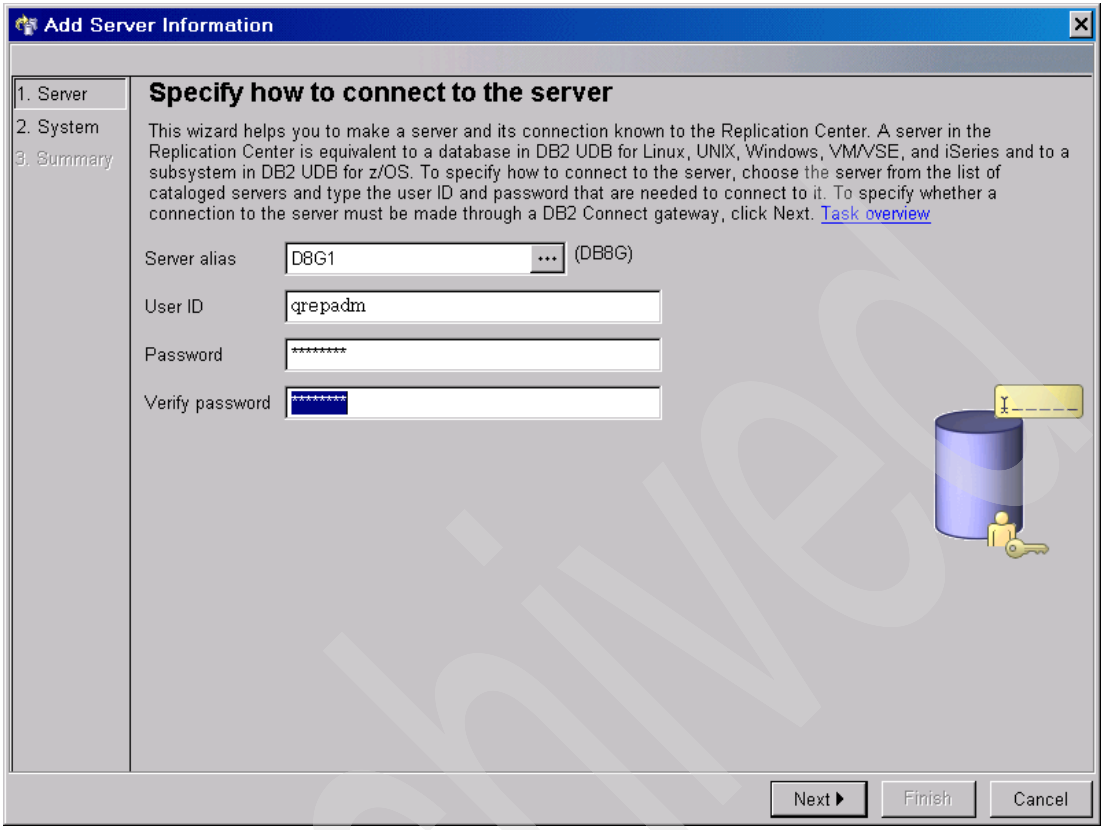


Figure H-6 Specify how to connect to the server

Provide the user ID and password for connecting to the selected server, as shown in Figure H-6, and click **Next** to proceed to Figure H-7 to specify the system where the server resides.

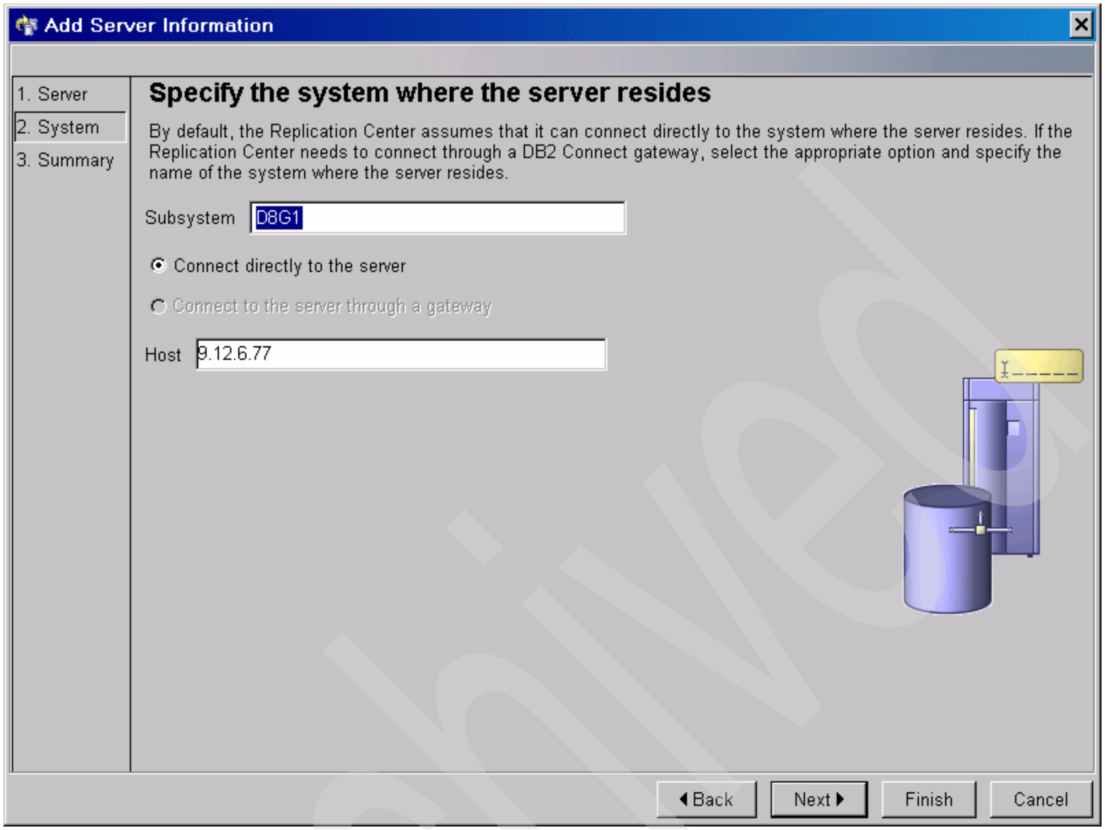


Figure H-7 Specify the system where the server resides

Specify the Subsystem and Host information as shown in Figure H-7, and click **Next** (Figure H-8) to review the summary of information provided.

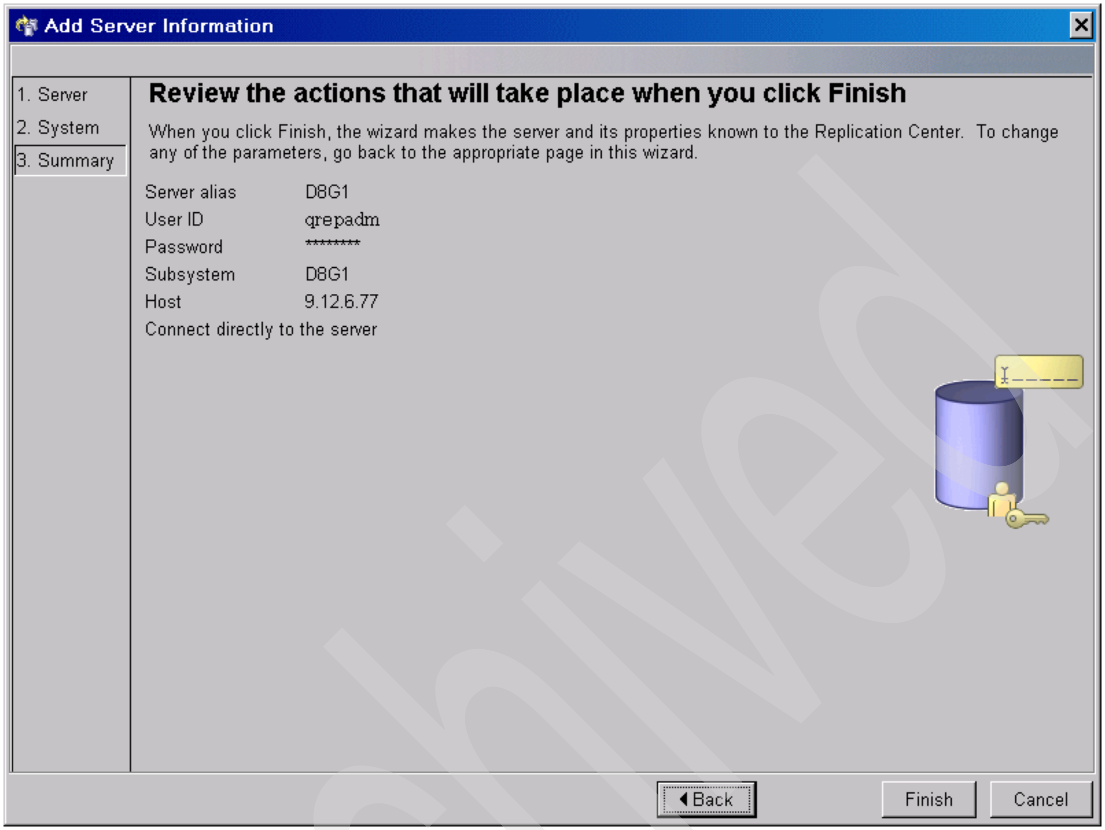


Figure H-8 Review the actions that will take place when you click Finish

After reviewing the summary information shown in Figure H-8 for accuracy, click **Back** to correct errors, or **Finish** to proceed to Figure H-8 to complete the cataloging of the database server in the RC.

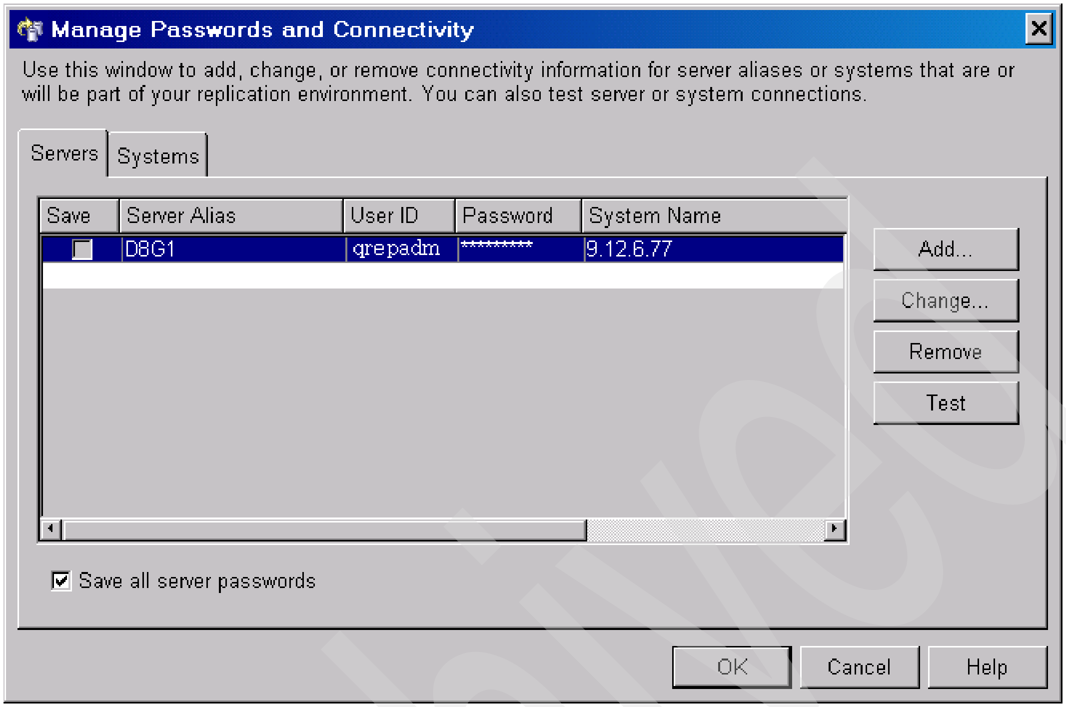


Figure H-9 Manage Passwords and Connectivity - D8G1 server added

The newly added database server is displayed in Figure H-8. One can test whether the configuration was successful by clicking the **Test** button. Figure H-19 on page 899 and Figure H-20 on page 899 show a successful test for our database server.

Add additional database servers and test their connectivity, as shown in Figure H-10 through Figure H-18 on page 898.

Then select the **Systems** tab (Figure H-9 on page 890) to ensure that the systems on which the newly added database servers are displayed, as shown in Figure H-21 on page 900.

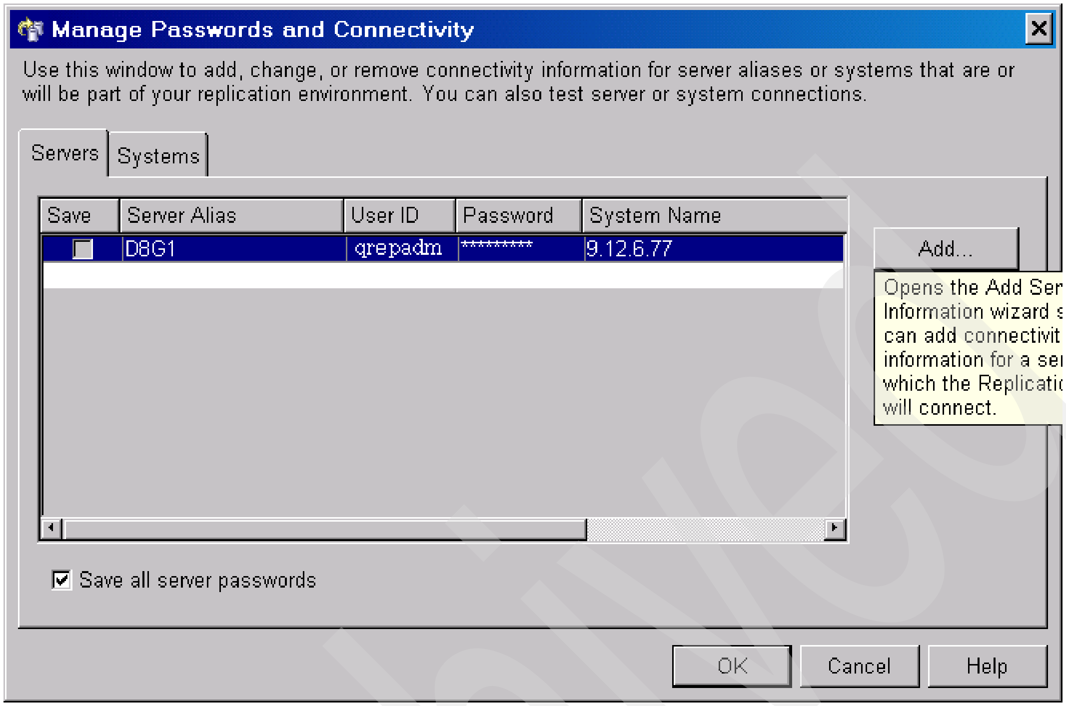


Figure H-10 Manage Passwords and Connectivity

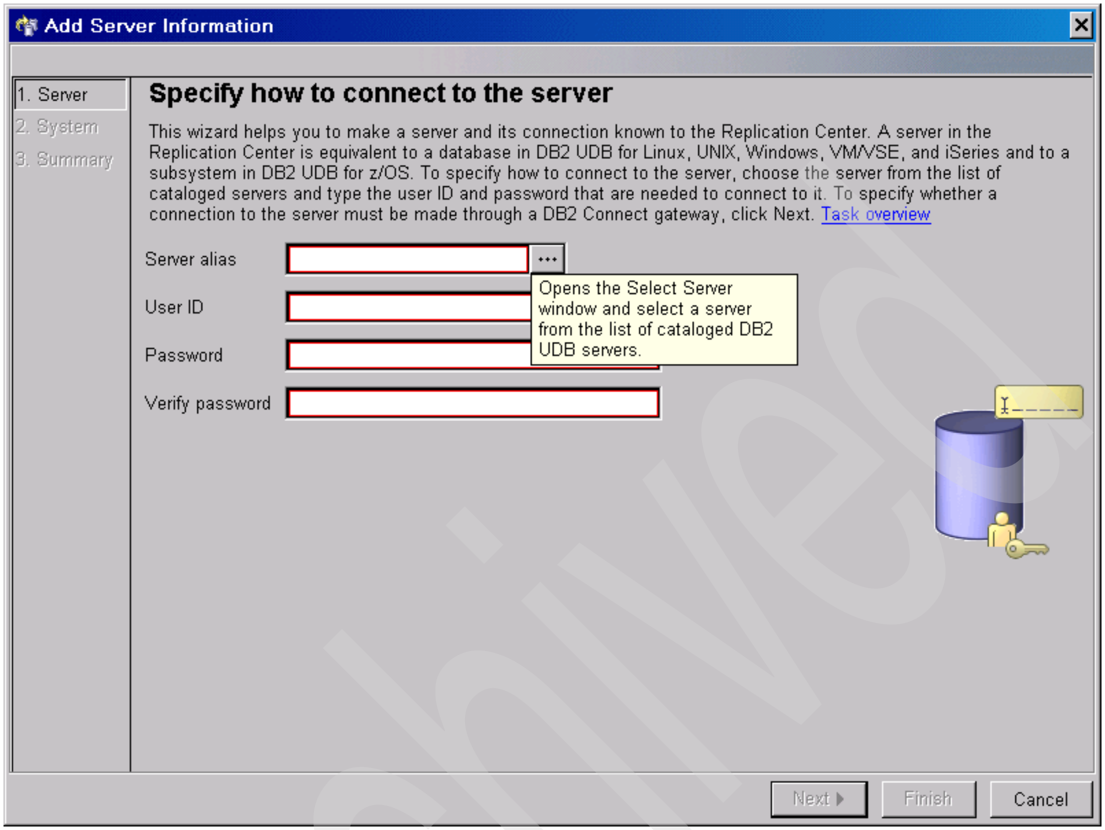


Figure H-11 Specify how to connect to the server

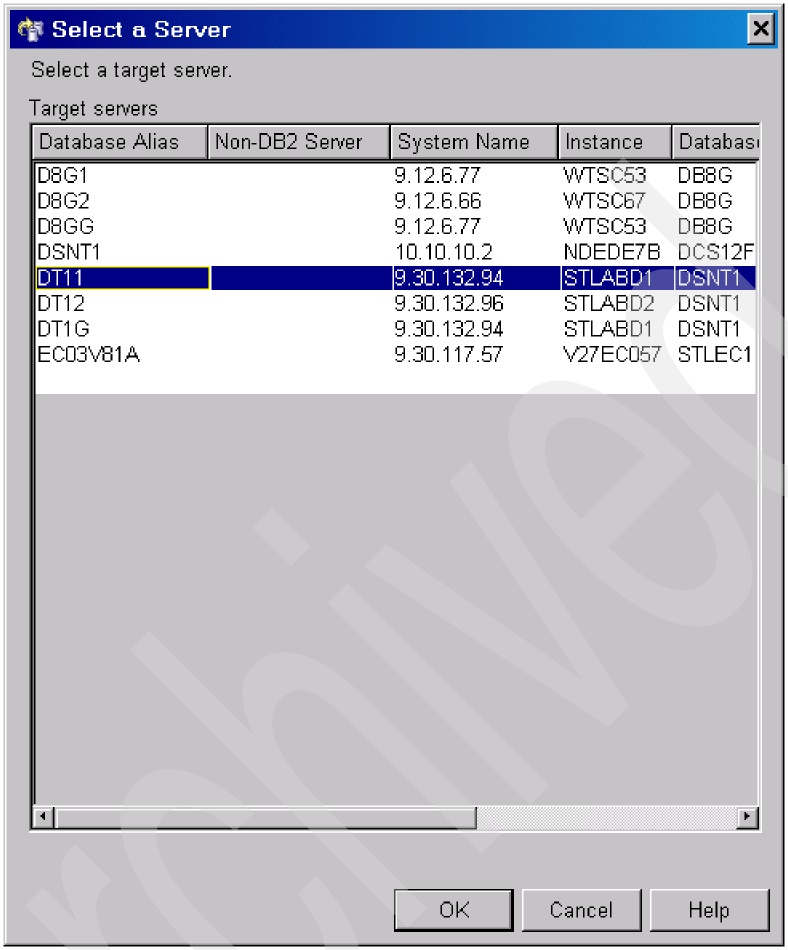


Figure H-12 Select a Server

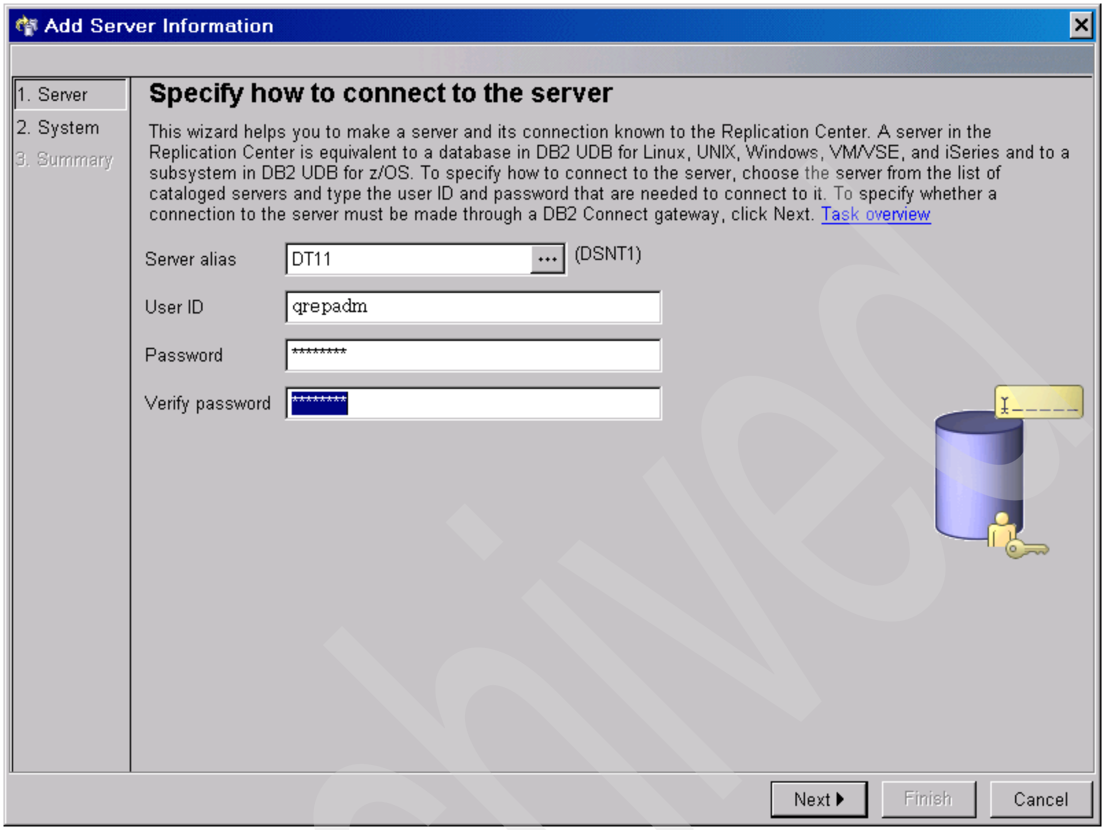


Figure H-13 Specify how to connect to the server

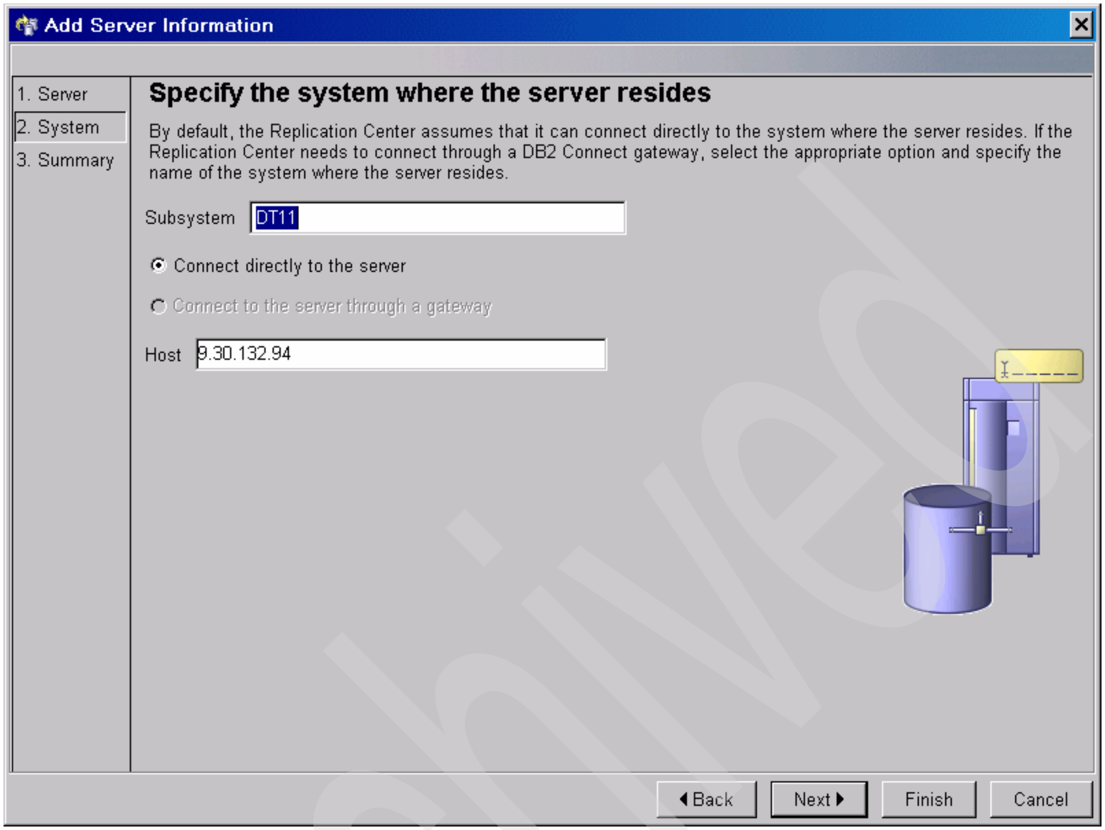


Figure H-14 Specify the system where the server resides

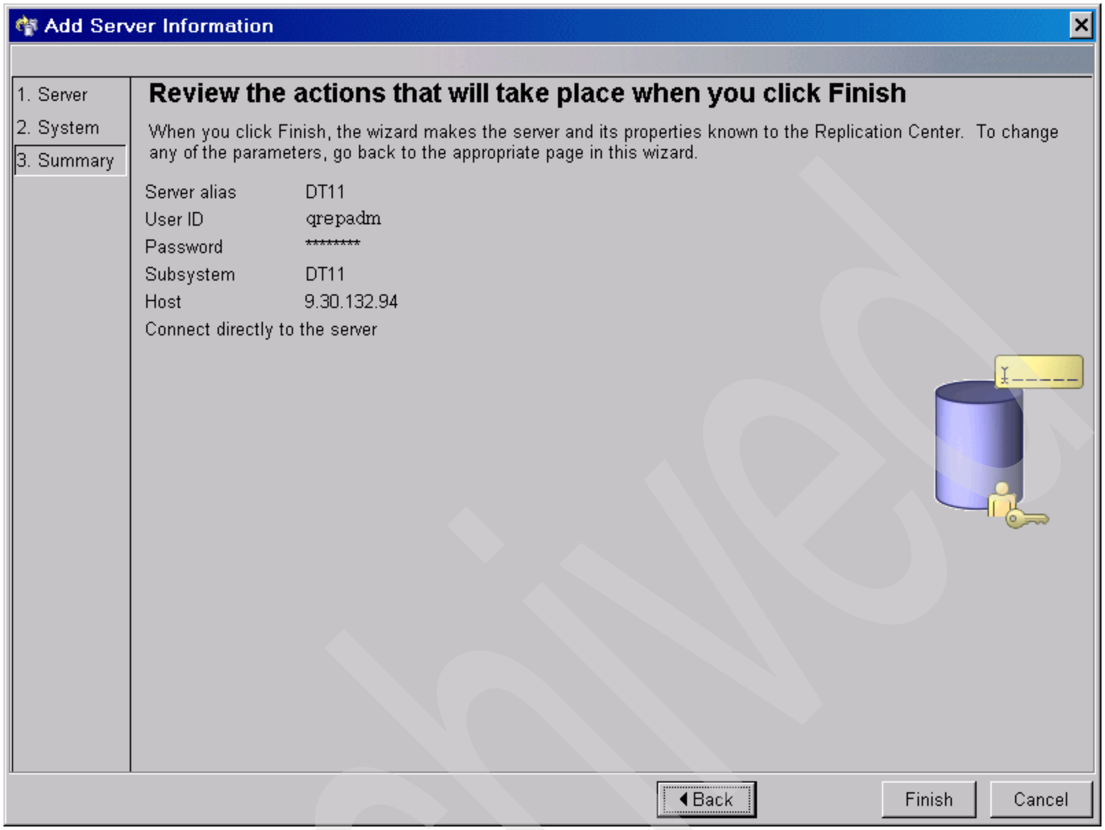


Figure H-15 Review the actions that will take place when you click Finish

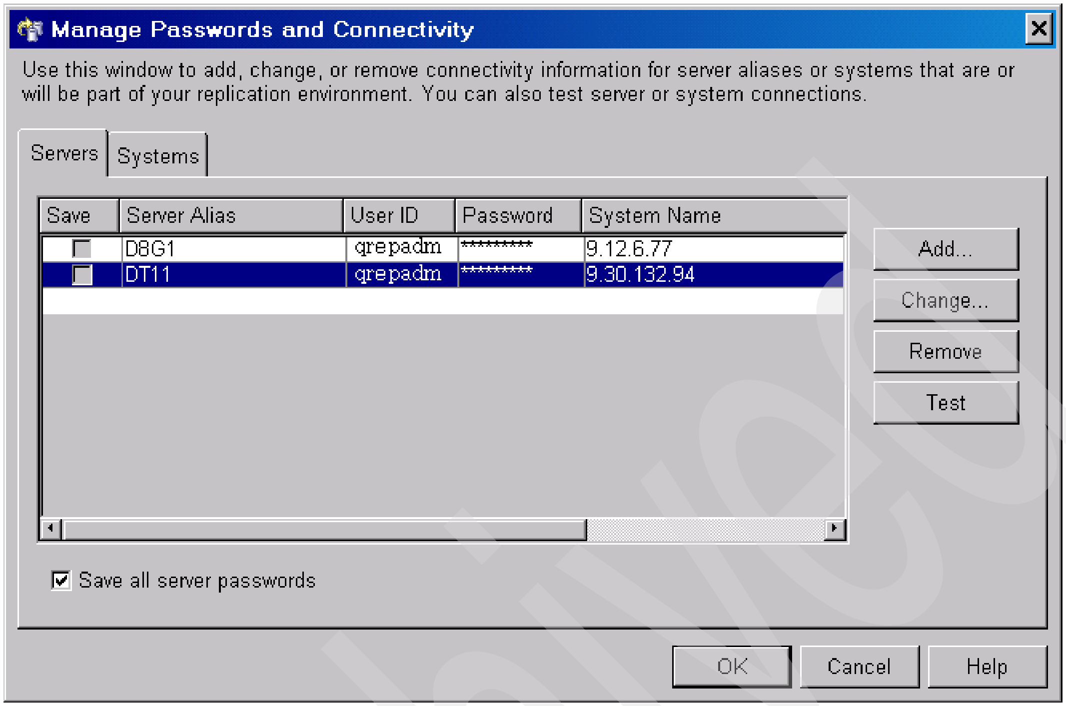


Figure H-16 Manage Passwords and Connectivity

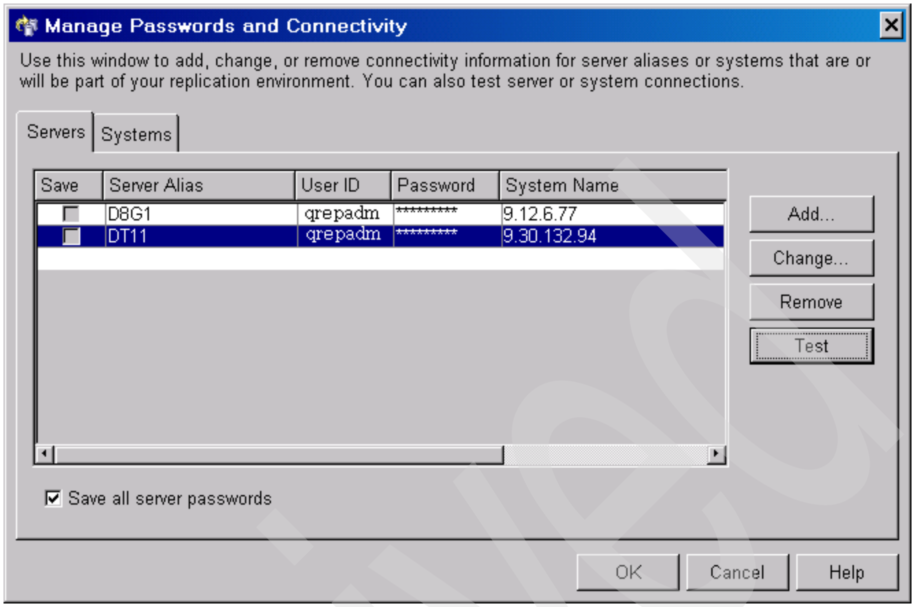


Figure H-17 Test connection to the database server

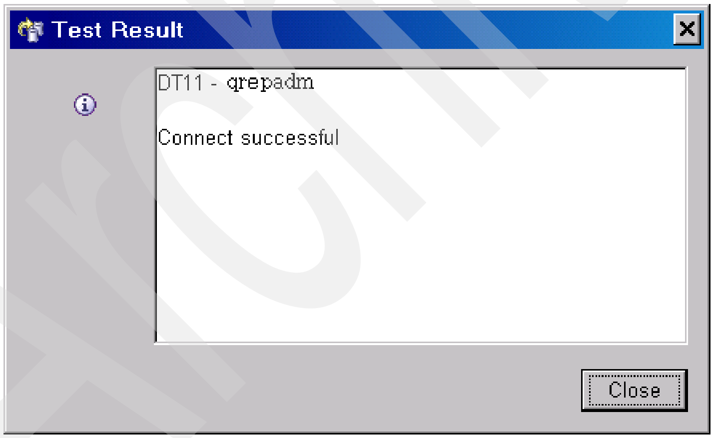


Figure H-18 Successful connect message

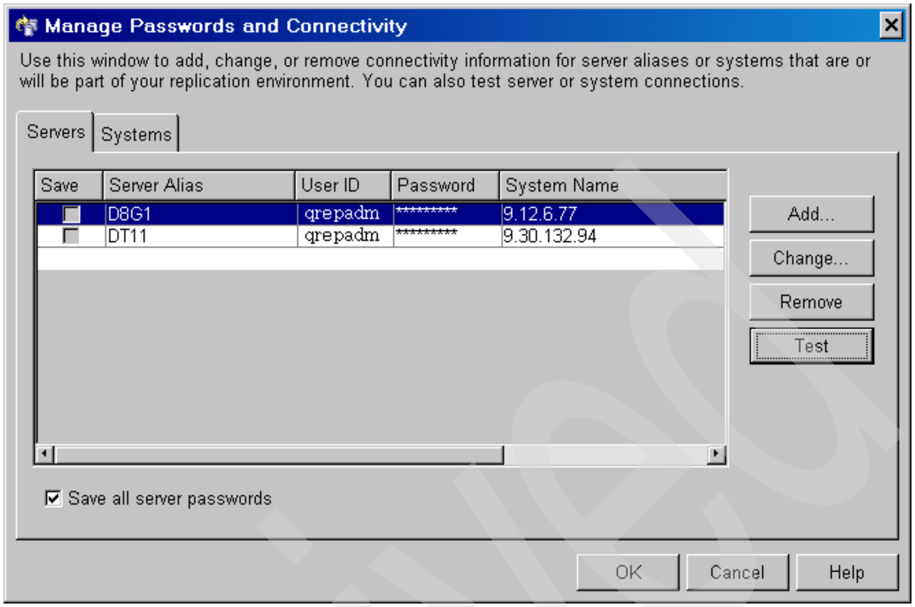


Figure H-19 Test connection to the database server

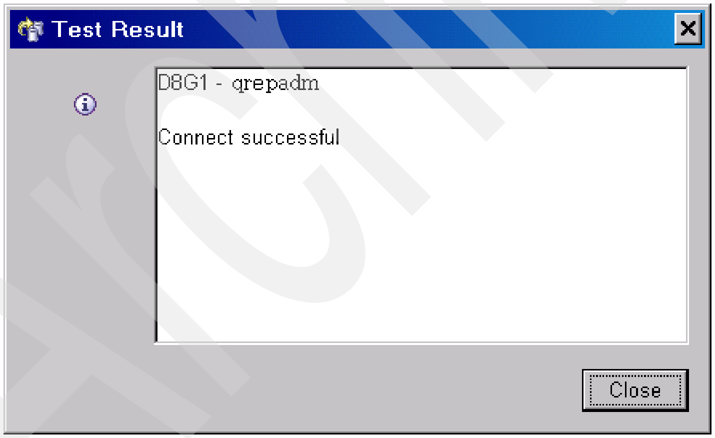


Figure H-20 Successful connect message

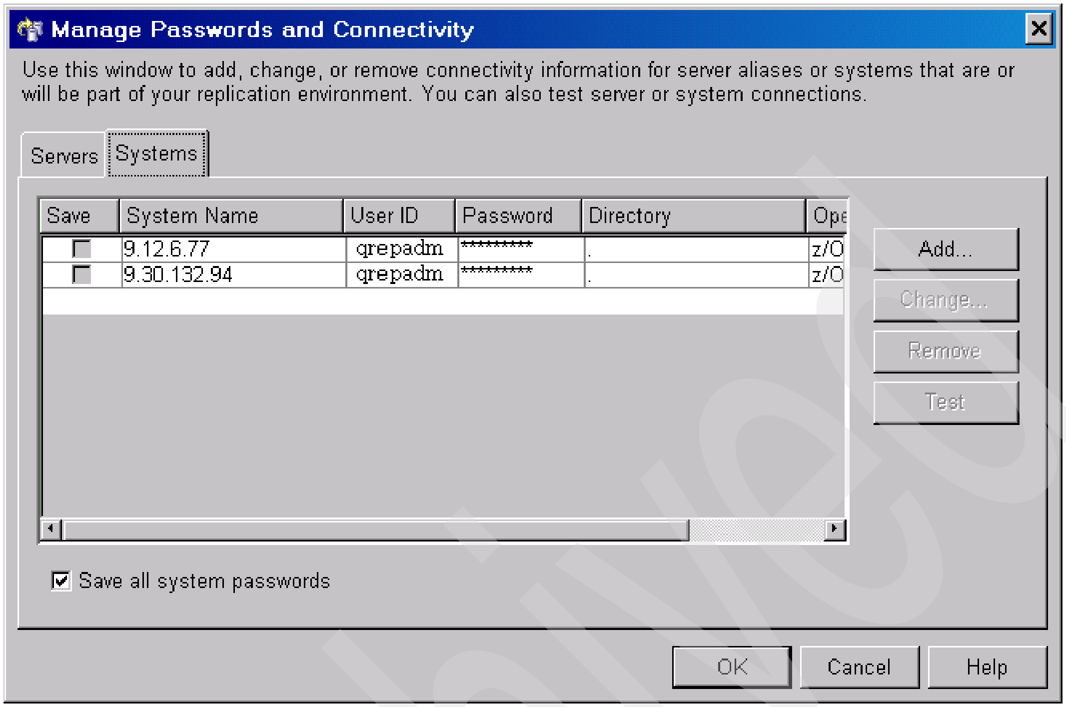


Figure H-21 Manage Passwords and Connectivity - Systems

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 902. Note that some of the documents referenced here may be available in softcopy only.

- *MQSeries Primer,* REDP-0021-00

Other publications

These publications are also relevant as further information sources:

- *IBM DB2 Information Integrator, Replication and Event Publishing Guide and Reference Version 8.2*, SC18-7568-00
- *IBM DB2 Information Integrator, ASNCLP Program Reference for Replication and Event Publishing Version8.2*, SC18-9410-00
- *WebSphere MQ for AIX, Quick Beginnings Version5.3*, GC34-6079-01
- *WebSphere MQ, System Administration Guide*, SC34-6069-01
- *WebSphere MQ, Script (MQSC) Command Reference*, SC34-6055-01
- *WebSphere MQ, Security Version5.3*, SC34-6079-01
- *WebSphere MQ, Intercommunication*, SC34-6059-01

Online resources

These Web sites and URLs are also relevant as further information sources:

- 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

- Q Replication V8.2 Information Roadmap

  http://www-106.ibm.com/developerworks/db2/roadmaps/qrepl-roadmap-v8.2.html

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

C

Q

T

IBM

Redbooks

WebSphere Information Integrator Q Replication: Fast Track Implementation Scenarios

WebSphere Information Integrator Q Replication: Fast Track Implementation Scenarios

Redbooks

WebSphere Information Integrator Q replication overview

Bidirectional and P2P scenarios on z/OS

Bidirectional and P2P scenarios on AIX

This IBM Redbook provides an overview of WebSphere Information Integrator Q replication, and provides guidelines for exploiting its high throughput, low latency, and other capabilities in the design and implementation of high-availability and high-performance distributed business solutions on the z/OS and AIX platforms. It also describes a step-by-step approach to implementing bidirectional and peer-to-peer replication solutions in a two-server environment on the z/OS and AIX platforms.

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

SG24-6487-00 ISBN 0738492434