Rational Software Corporation®

Rational® ClearQuest® MultiSite

Administrator's Guide

VERSION: 2003.06.00 AND LATER

PART NUMBER: 800-026170-000

UNIX/WINDOWS EDITION

Rational
the software development company

support@rational.com
http://www.rational.com

Third Party Notices, Code, Licenses, and Acknowledgements

Contents

MultiSite Overview

Replication and Synchronization

MultiSite Management

Troubleshooting

MultiSite Reference Pages

Figures

Tables

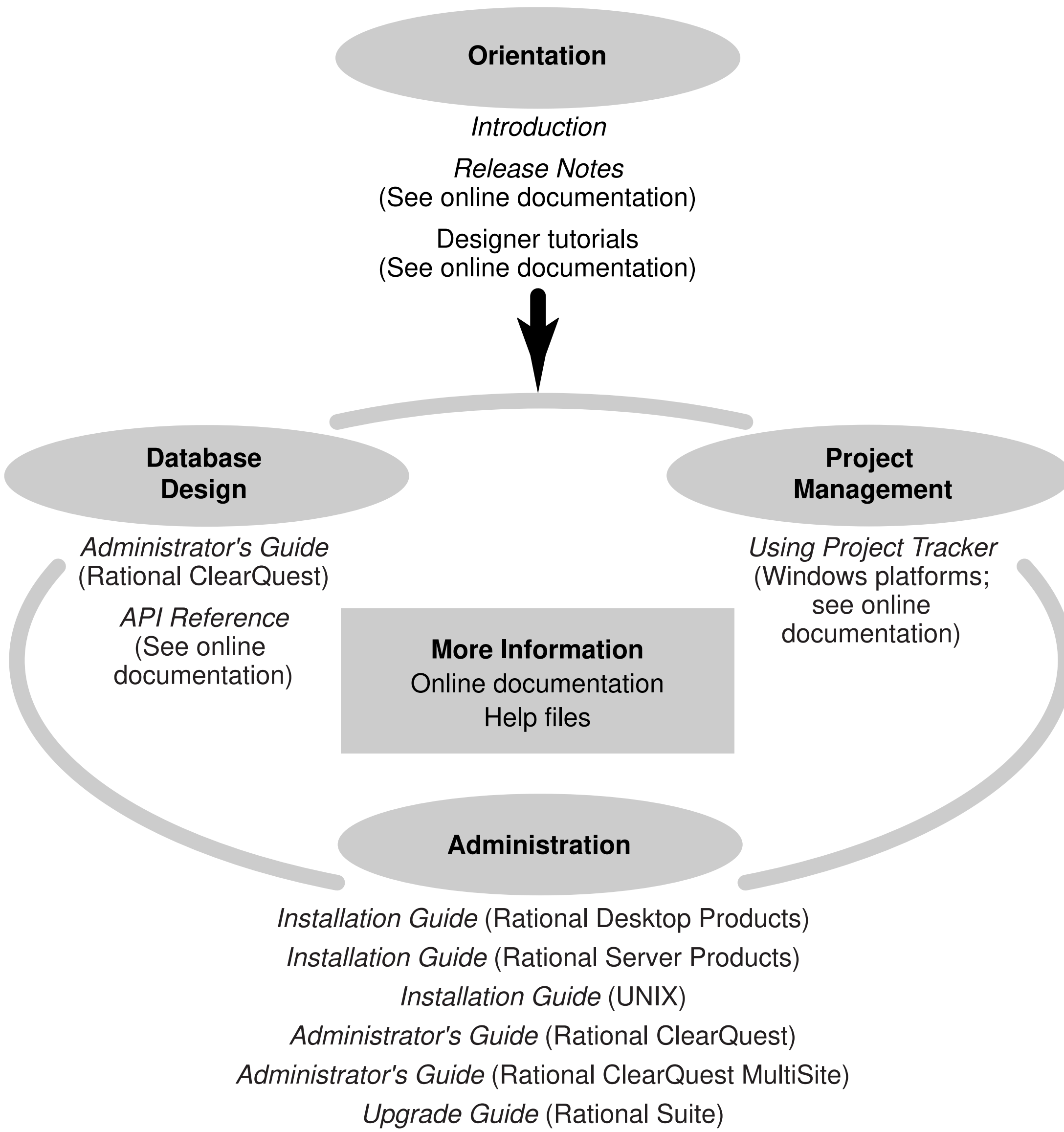
Preface

Rational ClearQuest MultiSite (abbreviated to "MultiSite" in this manual) is a layered product option to Rational ClearQuest. It supports parallel software development across project teams distributed geographically and provides automated, error-free replication of ClearQuest databases.

About This Manual

This manual is for all MultiSite administrators. It assumes you have experience with ClearQuest. The manual provides an overview of MultiSite, describes how to set up and use it, and gives troubleshooting suggestions.

ClearQuest Documentation Roadmap



ClearQuest Integrations with Other Rational Products

Integration	Description	Where it is documented
ClearQuest- Base ClearCase	Associates change requests with versions of ClearCase elements.	ClearCase: *Developing Software* ClearCase: *Managing Software Projects* ClearQuest: *Administrator's Guide*
ClearQuest-PurifyPlus	Allows developers to invoke ClearQuest from PurifyPlus.	PurifyPlus Help ClearQuest: *Administrator's Guide*
ClearQuest- RequisitePro	Allows developers to invoke ClearQuest from RequisitePro and to associate requirements with ClearQuest change requests.	Rational Suite: *Administrator's Guide* ClearQuest: *Administrator's Guide*
ClearQuest-SoDA	Collects information from ClearQuest and presents it in various report formats.	*Using Rational SoDA for Word* *Using Rational SoDA for Frame* SoDA Help
ClearQuest- Test Manager	Allows developers to invoke ClearQuest from TestManager.	*Using Rational TestManager* ClearQuest: *Administrator's Guide*
ClearQuest-Robot	Allows developers to invoke ClearQuest from Robot.	Rational Robot User's Guide Rational Robot Help
ClearQuest-UCM	Links UCM activities to ClearQuest records.	ClearCase: *Developing Software* ClearCase: *Managing Software Projects* ClearQuest: *Administrator's Guide*

Typographical Conventions

This manual uses the following typographical conventions:

- *ccase-home-dir* represents the directory into which the ClearCase Product Family has been installed. By default, this directory is /opt/rational/clearcase on UNIX and C:\Program Files\Rational\ClearCase on Windows.
- *cquest-home-dir* represents the directory into which Rational ClearQuest has been installed. By default, this directory is /opt/rational/clearquest on UNIX and C:\Program Files\Rational\ClearQuest on Windows.
- **Bold** is used for names the user can enter; for example, command names and branch names.
- A sans-serif font is used for file names, directory names, and file extensions.
- **A sans-serif bold font** is used for GUI elements; for example, menu names and names of check boxes.
- *Italic* is used for variables, document titles, glossary terms, and emphasis.
- `A monospaced font` is used for examples. Where user input needs to be distinguished from program output, **bold** is used for user input.
- Nonprinting characters appear as follows: <EOF>, <NL>.
- Key names and key combinations are capitalized and appear as follows: SHIFT, CTRL+G.
- [ ] Brackets enclose optional items in format and syntax descriptions.
- { } Braces enclose a list from which you must choose an item in format and syntax descriptions.
- | A vertical bar separates items in a list of choices.
- ... In a syntax description, an ellipsis indicates you can repeat the preceding item or line one or more times. Otherwise, it can indicate omitted information.

  Note: In certain contexts, you can use "**...**" within a pathname as a wildcard, similar to "*" or "?". For more information, see the **wildcards_ccase** reference page.
- If a command or option name has a short form, a "medial dot" (·) character indicates the shortest legal abbreviation. For example:

  lsc·heckout

Online Documentation

Rational ClearQuest includes online documentation, as follows:

Help System: Use the **Help** menu, the **Help** button, or the F1 key.

Reference Pages: On Windows, to display a reference page for Rational ClearQuest MultiSite, at the command prompt type **multiutil man** *command-name*.

API Reference: Click **Start > Programs > Rational Software > Rational ClearQuest > Rational ClearQuest API Reference**.

Tutorial: Click **Start > Programs > Rational Software > Rational ClearQuest > ClearQuest Designer Tutorial**.

PDF Manuals: Navigate to:

- On UNIX, *cquest-home-dir*/doc/books
- On Windows, *cquest-home-dir*\doc\books

Customer Support

If you have any problems with the software or documentation, please contact Rational Customer Support by telephone, fax, or electronic mail as described below. For information regarding support hours, languages spoken, or other support information, click the **Support** link on the Rational Web site at **www.rational.com**.

Your location	Telephone	Facsimile	Electronic mail
North America	800-433-5444 toll free or 408-863-4000 Cupertino, CA	408-863-4194 Cupertino, CA 781-676-2460 Lexington, MA	**support@rational.com**
Europe, Middle East, and Africa	+31-(0)20-4546-200 Netherlands	+31-(0)20-4546-201 Netherlands	**support@europe.rational.com**
Asia Pacific	61-2-9419-0111 Australia	61-2-9419-0123 Australia	**support@apac.rational.com**

MultiSite Overview

Introduction to MultiSite

1

Rational ClearQuest MultiSite adds a powerful capability to Rational ClearQuest. With MultiSite, developers at different locations can use the same database set (a schema repository and its associated user databases). Each location has its own copies, or replicas, of the schema repository and user databases. At any time, changes made in one replica can be sent in update packets to other replicas. The update process can be automated or can be started manually with a command.

An organization can use MultiSite to distribute independent, but related, development efforts across multiple cities, nations, or continents. For example, a company in the United States has development and testing sites in India, Argentina, Japan, and Australia. Because it is not practical for all engineers to access the databases in the United States, the company uses MultiSite to distribute the development.

MultiSite can also be used at a single geographical location to allow independent groups to work with the same development data or to be a backup mechanism. For example, a company that wants extra reliability for backups can create local replicas for a database set.

This chapter gives an overview of the major features in MultiSite. Chapter 2, *MultiSite Operation*, contains more details about how the features work.

Understanding the Architecture of MultiSite

The following sections describe the MultiSite architecture.

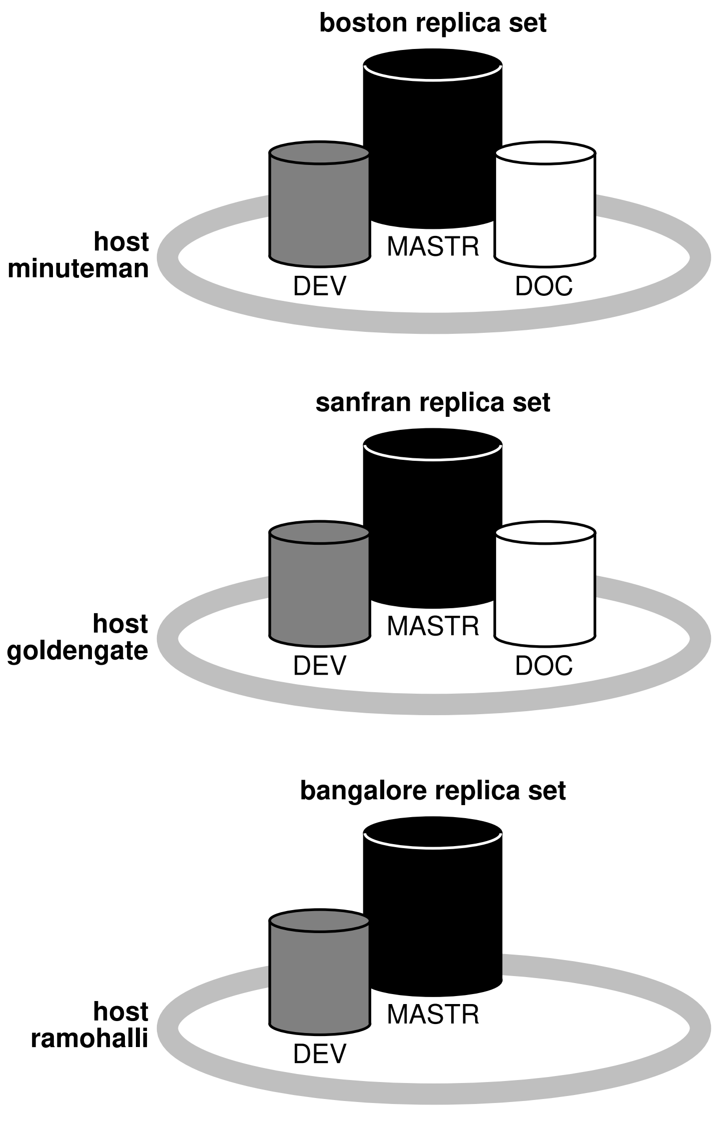
Replicated Database Sets

A database set consists of a schema repository and its associated user databases. A user database cannot exist without its corresponding schema repository, so when you replicate a database set, both the schema repository and the user database are replicated. When you work with a database replica, you are working with two physical databases: a schema repository replica and a user database replica.

Clans, Families, and Sites

A clan (Figure 1) consists of all replicas of a database set. Within a clan, replicas are grouped into replica families. A replica family is all the replicas of a specific database.

Figure 1 A MultiSite Clan



Schema repositories and user databases belong to separate replica families. For user databases, the family name is the same as the database name. The family name of a schema repository is always MASTR.

A site is a named collection of replicas in the same clan that reside at the same location. Each site has a schema repository replica and at most one replica from each user database family. Each site is served by a synchronization server, which receives and sends update packets to replicas within its family. Updates for a site can contain updates to the user database replicas, the schema repository replica, or both.

Kinds of Schema Repositories

Within a clan, one of the schema repository replicas is the working schema repository. At the working schema repository, you can change schemas and create additional user databases which can then be replicated. At other schema repositories, you cannot change schemas or create user databases. A clan can have only one working schema repository.

At either kind of schema repository, you can perform the following tasks:

- Run **multiutil** commands from an administrator's workstation
- Install and configure store-and-forward
- Submit new records
- Modify records mastered by the current replica
- Administer users mastered by the current replica

MultiSite Terminology

MultiSite documentation uses the following terms.

Term	Definition
Replica	A copy of a user database or a schema repository. To refer to a replica, use the site name and family name.
Family	All the replicas of a specific user database or all the replicas of a specific schema repository. The family name of a user database replica is the database name of the originating database. The family name of a schema repository is always MASTR.
Site	A schema repository replica and its user database replicas.
Clan	All the replicas of a schema repository and all the replicas of the associated user databases. Replicas originating from the same database set use the same clan name, which is specified when the database set is activated.
Host, or synchronization server	The LAN name or IP address of the network node that handles packets for a site. This host must have the Rational Shipping Server installed on it.

Databases and Database Replicas

Each replica is recorded in a table in the schema repository database. This table, which is replicated, includes information about the replica, including the associated synchronization server. In addition, each schema repository database contains information about how to connect to each database in its database set. This information is not replicated.

Most information stored in ClearQuest databases is replicated. The following information is not replicated:

- Checked-out copies of schemas.
- Schema of a user database. (Local administrators must choose when to upgrade the database.)

Synchronizing Replicas in a Family

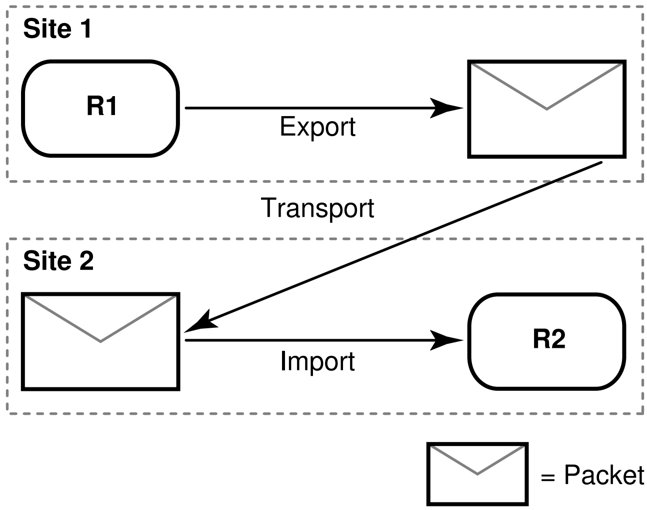
Because information in a replicated ClearQuest database is modified concurrently at different replicas, the contents of each replica in a family tend to diverge. In fact, the contents of a particular replica may never be identical to the contents of any other replica. To keep the replicas from diverging too much, each replica sends updates to one or more other replicas. Updating a user database replica may change both its database and its schema repository to reflect the activity that has taken place in one or more other replicas.

Information is exported from a replica in packets. A logical packet includes all the information required to create a new replica (replica-creation packet) or to update one or more existing replicas (update packet). For flexibility, and to accommodate limitations of data-transport facilities, each logical packet can be created as a set of physical packets.

After a logical packet is created with a **mkreplica** or **syncreplica** command invoked with the **–export** option and sent to a replica, it is processed at that replica by a **mkreplica** or **syncreplica** command invoked with the **–import** option. The changes that occurred originally at the sending replica (and perhaps at some other replicas, too) are added to the user database and schema repository of the importing replica. If the logical packet includes several physical packets, the import commands always process the physical packets in the correct order. No error occurs if the same packet is imported two or more times at a replica, unless the imports occur simultaneously.

Figure 2 illustrates the three phases of synchronization: export, transport, and import. At Site 1, a **syncreplica –export** command places records of operations from **R1** into a packet. The packet is sent to Site 2. At Site 2, a **syncreplica –import** command imports the contents of the packet into **R2**. Note that each synchronization is one-way. If two replicas update each other, two synchronizations are required.

Figure 2 Replica Synchronization



You can match the synchronization strategy for each family to its particular use patterns, your organization's needs, and the level of connectivity among the host machines. For one family, you can update replicas every hour, using a high-speed network; for another family, you can send updates only once or twice a month, using electronic mail or disk files as the delivery mechanism. For information about planning synchronization, see *MultiSite Use Model* on page 19. For information about creating and synchronizing replicas, see Chapter 6 and Chapter 7. *The Operation Log* on page 8 describes the mechanism that supports replication and synchronization.

Enabling Independent Development: Mastership

Because changes are made independently at multiple replicas, these changes may conflict. In a MultiSite environment, tracking changes and preventing data corruption are accomplished with an exclusive-right-to-modify scheme called *mastership*. Mastership determines when a user of a replica is allowed to modify data.

If the work done in different replicas were truly independent, the result would be chaos. If a record **SAMPL00001** is created in three replicas at the same time, it is impossible to determine which is the real record **SAMPL00001** and what must happen to the other two records.

Certain objects are assigned a master replica (or master). The initial master of an object is the replica where the object is created, and mastership can be changed subsequently (see Chapter 9, *Managing Mastership*). In general, an object can be modified or deleted only at its master replica.

Most objects in a ClearQuest database have a master replica. For more information about how mastership prevents conflicting changes, see *Mastership* on page 7.

Some conflicts are unavoidable. For example, a new user named **jsmith** can be created at two or more user database replicas during the same time period between synchronizations. You can minimize such conflicts by establishing naming conventions for objects, but if a conflict does occur, it is handled during import of an update packet. For more information, see *Conflict Resolution* on page 7.

Enforcing a Single Code Page in a Clan

When you use ClearQuest, each client that accesses your ClearQuest databases has its own code page. A code page specifies the set of characters that are valid in a given environment. With ClearQuest, a code page defines the set of characters that are manipulated correctly on a particular client.

Because ClearQuest and ClearQuest MultiSite have limited internationalization support, data corruption can occur when clients using different code pages attempt to modify the same data. These problems are more likely to occur in a MultiSite environment when data from one code page is exported to replicas running different code pages, causing data corruption and divergence (members of a replica family having different values for the same record). Therefore, all databases, database hosts, servers, and clients in a MultiSite environment must use the same code page or limit the data to ASCII characters (required for UNIX clients).

To enforce the use of a single code page and to ensure data integrity, a ClearQuest administrator must set the data code page value for a database set before activating it for replication. The code page of each database must also be set using the vendor database tools. Also, if clients running versions of ClearQuest prior to v2003.06.00 access your databases, the CharacterSetValidation package must be applied to prevent users from entering data from multiple code pages into a database record.

For more information about code pages and the data code page setting for a database set, see the *Administrator's Guide* for Rational ClearQuest.

MultiSite Operation 2

This chapter provides more detail about the topics introduced in Chapter 1.

Mastership

The following objects have a master replica:

- Records
- Users and groups
- Workspace items (queries, reports, charts and folders)
- Schema repository

For user database records, mastership information is stored as a field value in a record. Users can change the value of the mastership field to transfer mastership to another replica. Mastership of the record is sent to the new master replica during the next synchronization. For all other database objects, an administrator must change the mastership.

When you work in a MultiSite environment, you must adjust your workflow to account for any stage in your software lifecycle that requires a change of mastership for a record or defect.

For example, users in Paris can submit defects that should be worked on by developers in Boston. But without a change of mastership, developers in Boston cannot modify defects entered by users in Paris.

For more information about how mastership affects your workflow, see *Defining Mastership Policies* on page 18.

Conflict Resolution

Mastership restrictions prevent most inconsistent changes in different replicas, but some are unavoidable. To avoid many naming conflicts, the administrators for a family must create and enforce naming rules for objects. A use model that is enforced consistently across sites reduces the potential for conflicts. For example, the administrators for a family follow these rules:

- All location-specific objects must include a location identifier.
- Objects that will be used at multiple replicas are all created at one replica.

When naming conflicts occur, MultiSite displays the name of the originating replica (the keysite) in the names. If this happens, you should rename the conflicting objects as soon as possible; see *Resolving Naming Conflicts* on page 98.

The Operation Log

This section describes the mechanism that supports synchronization. This information is not required to use MultiSite, but is helpful when you want to deepen your understanding of the error-recovery facilities described in Chapter 10, *Troubleshooting MultiSite Operations*.

Most changes to a replicated database are recorded as entries in an operation log (oplog). These entries store all the information required to replay the changes in another replica:

- The identity of the replica in which the change originated.
- The specific changes to a database record or to a schema in the schema repository made during a single checkout; for example, submission of a new record, schema updates, and so on.
- An integer sequence number: 1 for the first change originating at a particular replica, 2 for the next change, and so on. This is called the oplog ID of the oplog entry.

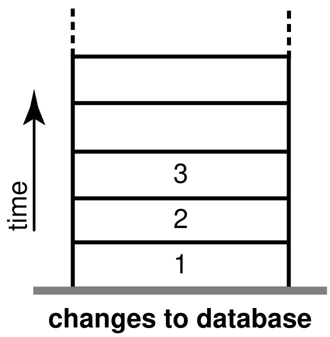
The exact kind and amount of information varies with the specific operation. For example, an oplog entry for the submission of a new record has different, and more, information than an oplog entry for modifying an existing record.

Note: You can delete a replica's oplog entries after they have been used to update other replicas. For more information, see *Scrubbing Parameters for Replicas* on page 29.

Tracking Operations for Each Replica

The history of an unreplicated database is a linear sequence of operations (Figure 3).

Figure 3 History of Changes to a Database

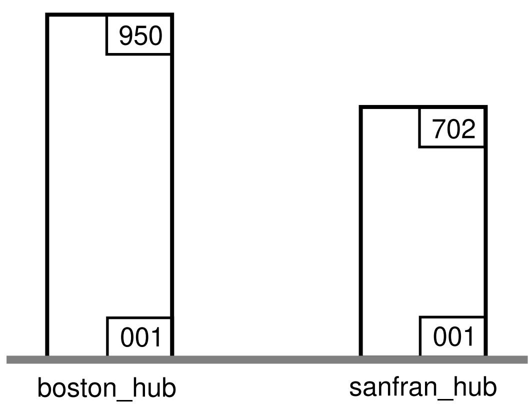


Within a replica family, changes are tracked for each replica. This is why an oplog entry includes the identity of the replica where the operation originated. Thus, the history of a replica family can be viewed as several stacks of oplog entries. Each stack is represented by a linear sequence of oplog IDs for the operations that originate in that replica.

Figure 4 shows the state of two replicas in a family:

- Operations with oplog IDs 1–950 have occurred at replica **boston_hub**.
- Operations 1–702 have occurred at replica **sanfran_hub**.

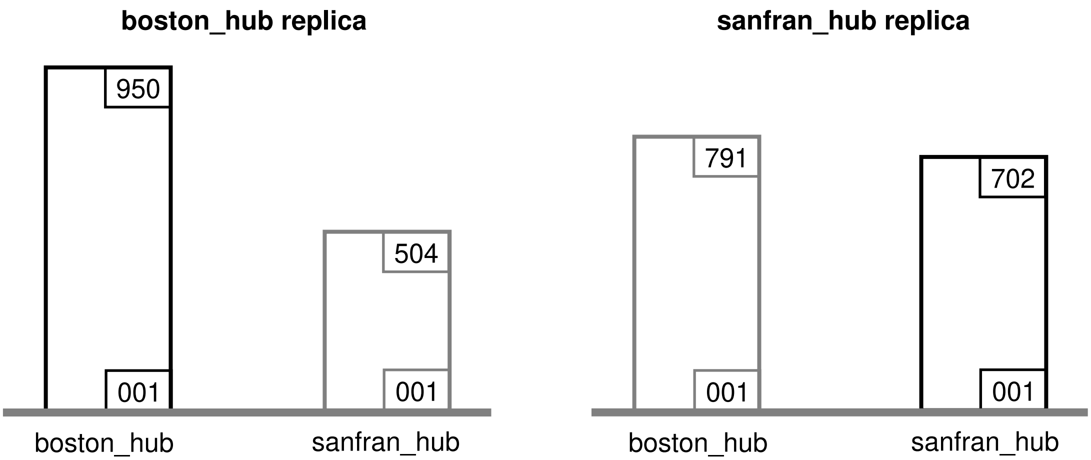
Figure 4 State of a Family



A replica has accurate data only about its own operations. Until it receives update packets, its information about other replicas is out of date. For example, replica **boston_hub** records 950 local operations, but has received update packets for only 504 **sanfran_hub** operations. Similarly, replica **sanfran_hub** records 702 local operations, but has received update packets for only 791 **boston_hub** operations.

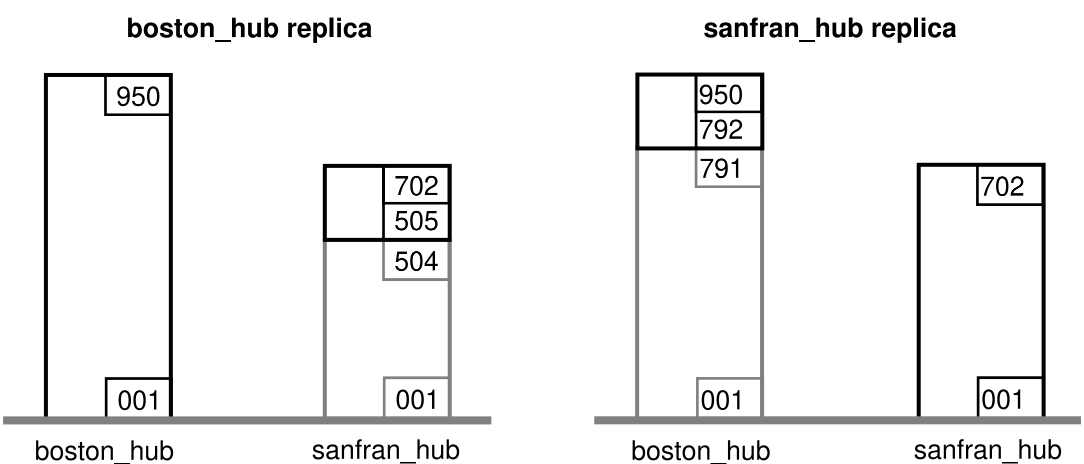
Figure 5 illustrates this scenario, in which each replica is out of date with respect to the operations originating at the other replica.

Figure 5 Out-of-Date Replicas



Picturing a replica family as a set of oplog stacks, shown in Figure 5, makes it easy to understand the synchronization process. For example, an update packet sent from replica **boston_hub** to replica **sanfran_hub** consists of increments to the stack for replica **boston_hub** (operations 792–950). Figure 6 shows the two increments. Because **sanfran_hub** knows its own state, it needs updates only for operations originating at other replicas. (In certain error-recovery situations, you must reset a replica's data about its own operations. See Chapter 10, *Troubleshooting MultiSite Operations*.)

Figure 6 Updates Between Two Replicas



Note: By the time the packet is imported at **sanfran_hub**, additional changes may have been made at **boston_hub**. Those changes are not included in the update packet.

Oplog IDs and Epoch Numbers

An epoch number is the total number of operations that originated at a particular replica. In Figure 4, the epoch number for **boston_hub** is 950.

The MultiSite synchronization scheme attempts to minimize the amount of data transmitted among replicas. Each replica keeps track of these epoch numbers:

- **Changes made in the current replica**. The number of operations that originated at the current replica.
- **Changes at sibling replicas that have been imported to the current replica**. When **syncreplica** writes an operation from an update packet to the current replica, it increments the epoch number that records the number of operations originating at the sibling replica that have been imported at the current replica.
- **Estimates of the states of other replicas**. For each other replica, an estimate of its own changes and other replicas' changes. The current replica keeps track of the operations it has sent to other replicas, and assumes that these operations are imported successfully.

Table 1 shows how these epoch numbers fall into an epoch number matrix. Each replica maintains its own such matrix, revising its rows as work occurs locally and as it exchanges update packets with other replicas:

- When work occurs in the **boston_hub** replica, its own epoch number is incremented.
- When the **boston_hub** replica receives an update from **sanfran_hub**, it revises its own row (**boston_hub**) and the **sanfran_hub** row in its epoch number matrix.
- When the **boston_hub** replica generates an update packet to be sent to **sanfran_hub**, it revises the **sanfran_hub** row in its epoch number matrix.

  Note that a **syncreplica –export** command updates epoch numbers immediately. It does not wait for acknowledgment from the importing replica that the packet has been received and applied correctly. During normal MultiSite processing, no manual intervention is required to maintain the accuracy of the epoch number matrices for the various replicas. However, failure to apply a packet may require manual intervention, as described in *Recovering from Lost Packets* on page 96.

Table 1 Two-Row Epoch Number Matrix at Replica boston_hub

	Operations originated at boston_hub	Operations originated at sanfran_hub
boston_hub's record of its own state	950	504
boston_hub's estimate of **sanfran_hub**'s state	912	504

The contents of this matrix are reported by the **lsepoch** command at the **boston_hub** replica:

multiutil lsepoch -clan telecomm -site boston_hub -family PRODA -user bostonadmin -password secret

```
Multiutil: Estimates of the epochs from each site replayed at site
'boston_hub' (@minuteman):
boston_hub: 950
sanfran_hub: 504
Multiutil: Estimates of the epochs from each site replayed at site
'sanfran_hub' (@goldengate):
boston_hub: 912
sanfran_hub: 504
```

A **syncreplica –export** command entered at **boston_hub** uses this matrix as follows to generate an update destined for **sanfran_hub**:

1 At the **boston_hub** replica, the number of local operations is 950 (number in upper left corner of matrix), and the estimate is that the **sanfran_hub** replica has imported all operations through oplog ID 912 (number in lower left corner).

2 The update packet that the **boston_hub** replica sends to the **sanfran_hub** replica includes **boston_hub** oplog entries 913-950. After the Boston administrator invokes **syncreplica –export**, the **sanfran_hub** row is updated:

   multiutil lsepoch -clan telecomm -site boston_hub -family PRODA -user lexadmin -password secret

   ```
   Multiutil: Estimates of the epochs from each site replayed at site
   'boston_hub' (@minuteman):
   boston_hub: 950
   sanfran_hub: 504
   Multiutil: Estimates of the epochs from each site replayed at site
   'sanfran_hub' (@goldengate):
   boston_hub: 950
   sanfran_hub: 504
   ```

Indirect Synchronization

If a family includes more than two replicas, synchronization can occur indirectly. A replica can include nonlocal changes in update packets. For example, if the **boston_hub** replica exchanges updates with the **sanfran_hub** and **bangalore** replicas, it sends **bangalore** oplog entries that it has received previously from **sanfran_hub**. These entries may or may not bring replica **bangalore** up to date on **sanfran_hub**'s changes. (An update sent from **sanfran_hub** to **bangalore** does bring **bangalore** up to date.)

Note: If a replica does not receive packets directly from some replicas in its family, its rows for those replicas may contain zeros. This is expected behavior.

Table 2 shows replica **boston_hub**'s epoch number matrix.

Table 2 Three-Row Epoch Number Matrix at Replica boston_hub

	Operations originated at boston_hub	**Operations originated at bangalore**	**Operations originated at sanfran_hub**
boston_hub's record of its own state	950	653	504
boston_hub's estimate of **sanfran_hub**'s state	912	653	504
boston_hub's estimate of **bangalore**'s state	709	653	221

The contents of this matrix are reported by the **lsepoch** command:

multiutil lsepoch -clan telecomm -site boston_hub -family PRODA -user susan -password passwd

```
Multiutil: Estimates of the epochs from each site replayed at site
'boston_hub' (@minuteman):
boston_hub: 950
sanfran_hub: 504
bangalore: 653
Multiutil: Estimates of the epochs from each site replayed at site
'sanfran_hub' (@goldengate):
boston_hub: 912
sanfran_hub: 504
bangalore: 653
Multiutil: Estimates of the epochs from each site replayed at site
'bangalore' (@ramohalli):
boston_hub: 709
sanfran_hub: 221
bangalore: 653
```

A **syncreplica –export** command at the Boston site uses this matrix to export an update for the **bangalore** replica:

1. At the **boston_hub** replica, there are 950 local operations (number in upper left corner of matrix), and the estimate is that the **bangalore** replica has imported all operations through oplog ID 709 (lower left corner).

2. For operations that originated at the **sanfran_hub** replica, **boston_hub** has imported all operations up to oplog ID 504 and estimates that **bangalore** has imported all operations through oplog ID 221.

3. The update packet that **boston_hub** sends to **bangalore** includes **boston_hub** operations 710-950 and **sanfran_hub** operations 222-504. The output of an **lsepoch** command at the **boston_hub** replica now looks like this:

   multiutil lsepoch -clan telecomm -site boston_hub -family PRODA -user susan -password passwd

   ```
   Multiutil: Estimates of the epochs from each site replayed at site
   'boston_hub' (@minuteman):
   boston_hub: 950
   sanfran_hub: 504
   bangalore: 653
   Multiutil: Estimates of the epochs from each site replayed at site
   'sanfran_hub' (@goldengate):
   boston_hub: 912
   sanfran_hub: 504
   bangalore: 653
   Multiutil: Estimates of the epochs from each site replayed at site
   'bangalore' (@ramohalli):
   boston_hub: 950
   sanfran_hub: 504
   bangalore: 653
   ```

Planning a MultiSite Implementation 3

Before you install and use Rational ClearQuest MultiSite, you need to plan your implementation. The plan should include the following items:

- MultiSite installation
- MultiSite licensing
- MultiSite use model
- Responsibilities of MultiSite administrators

This chapter describes these issues in more detail. We recommend that you document your plan and implement your design decisions in a set of test replicas before changing your development environment.

For a sample deployment checklist, see *Overview of Deployment Tasks* on page 32.

MultiSite Installation

For MultiSite installation instructions, see the *Installation Guide*.

Each site needs a synchronization server to handle packet transport. This host must have the Rational Shipping Server installed on it. Each site also needs a MultiSite administrative host, which must have the ClearQuest MultiSite Administration Tools (**multiutil**) installed on it. On this host, you run **multiutil** commands to synchronize and manage replicas.

The synchronization server host, administrative host, and machines that host ClearQuest clients and databases must all have the same code page. To prevent data corruption in a database family, the data code page value for a database set must be set to the code page used by these machines. For more information about code pages and about setting the data code page value, see the *Administrator's Guide* for Rational ClearQuest.

Each host where the shipping server will be used must have enough disk space for the MultiSite storage bay directories. The storage bays hold MultiSite packets, along with their corresponding shipping order files. Table 3 describes the amount of available disk space needed on the disk partition where the storage bay is located.

Table 3 Disk Space Needed for Storage Bay

Type of packet	Disk space needed
Replica-creation	At least twice the size of database and schema repository. Packets can be four times as large as the databases from which they were exported.
Update	On Windows, twice the size of the largest packet to be stored in the bay. The reason is that there may be two instances of the same packet in the bay at one time: one on its way to another destination, and another waiting to be applied to the replica on the current host.
	On UNIX, size of largest packet to be stored in the bay.

There is no formula for determining how large your update packets will be. The general rule is that synchronizing more frequently usually results in smaller packets. Yet even if you synchronize every hour, a large amount of development activity or release activity can occur in an hour and a large packet will be generated. If you are not sure that the available disk space can accommodate an unexpectedly large packet, you can configure MultiSite to limit the size of an update packet. For more information, see the **syncreplica** reference page.

For more information about specifying storage bays, see the **shipping.conf** (UNIX) and **MultiSite Control Panel** (Windows) reference pages.

MultiSite Licensing

A MultiSite license is required for any access to an object in a replica—by a MultiSite command or GUI, by a ClearQuest command or GUI, or by a standard operating system command. You can calculate the number of MultiSite licenses you need by determining how many developers will access replicated ClearQuest databases. If all developers will access these ClearQuest databases, you need the same number of MultiSite licenses as ClearQuest licenses. If some developers will not access replicated ClearQuest databases, you can purchase fewer MultiSite licenses.

For example, a company has two sites, with 20 developers at site A and 5 developers at site B. The company has three ClearQuest databases at site A; two will be replicated to site B and one will not be replicated. Five developers at site A will access only the unreplicated ClearQuest database, and the other 15 will work in all ClearQuest databases. Therefore, the company needs to purchase the following numbers of licenses:

Site	Number of ClearQuest licenses	Number of MultiSite licenses
A	20	15
B	5	5

Note: This example assumes that you purchase one ClearQuest license for each developer. If you have fewer ClearQuest licenses than developers, you can purchase a proportionate number of MultiSite licenses. For example, if the company purchased three ClearQuest licenses for site B, it would also purchase three MultiSite licenses for site B.

For more information about acquiring and setting up licenses, see the installation information for Rational ClearQuest.

Shipping Server Use with ClearCase and ClearQuest

If you use both Rational ClearCase MultiSite and Rational ClearQuest MultiSite, you use the same shipping server for both products. The shipping server is installed when you install ClearCase MultiSite.

Note: If you use ClearQuest MultiSite and ClearCase, or ClearQuest MultiSite alone, you must install the shipping server. For more information, see the installation information for Rational ClearQuest.

The following restrictions apply when you use both ClearCase MultiSite and ClearQuest MultiSite:

- You must use different storage classes for VOB replica packets and ClearQuest database replica packets. You can create multiple storage classes and use the **–sclass** option to specify a particular class. If you do not use the **–sclass** option, the default class is used:
  - For ClearCase MultiSite, the default storage class is **–default**. It is created when you install ClearCase MultiSite.
  - For ClearQuest MultiSite, the default storage class for **multiutil** commands that use the **–sclass** option is **cq_default**. The **shipping_server** and **mkorder** commands use **–default** as the default class.

    The **cq_default** class is not created during installation. If you plan to use this class, you must create the class and its shipping and return bays. For more information, see the **shipping.conf** (UNIX) and **MultiSite Control Panel** (Windows) reference pages.

If you do not create the **cq_default** storage class, you must create another class for use with ClearQuest MultiSite, and use the **–sclass** option in **multiutil** commands to specify that storage class. If the **cq_default** storage class does not exist and you do not specify the **–sclass** option in a **multiutil** command, the packet is placed in the storage bay associated with the **–default** class, which can cause problems at the importing site.

- You must use different bays for ClearQuest MultiSite storage classes and ClearCase MultiSite storage classes.
- If you uninstall one product, the other one may stop working. You must uninstall both products and then reinstall the one you want to continue using.

We recommend that you follow these guidelines when you use both ClearCase MultiSite and ClearQuest MultiSite:

- When you export a packet for a ClearQuest replica, use the **–sclass** option and specify a storage class.
- Enable e-mail notification for shipping server operations and specify an address to use only for messages originating from ClearQuest MultiSite operations. For more information, see the **control_panel** reference page.

Defining Mastership Policies

With ClearQuest MultiSite, you need to take mastership policies into consideration when planning your change management processes. Mastership adds another layer of control within your process.

For example, as a record moves from one state to another, different replicas can be assigned mastership of the record. Or you may choose to have all records of a certain type, regardless of state, mastered by a specific replica, which means that all modification of those records must take place at the master replica.

Mastership can affect different aspects of your process. For example:

- Hooks that modify records or field values can run only if the current replica masters the record.
- You must modify users and groups at the replica that masters the user or group.
- You must edit Workspace items (queries, reports and report formats) at the master replica.

- You can modify or customize schemas only at the working schema repository.

For more information about mastership, see Chapter 9, *Managing Mastership*.

Compatibility Issues

If you use the ClearCase/ClearQuest UCM integration, you must run **multiutil** from a machine that does not require this integration. This is because **multiutil** requires special database set names that are not supported by the UCM integration.

Must Kill cqintsrv Process Before Running mkreplica -export

The ClearQuest Integration Server (**cqintsrv**) caches information about its current session. You must terminate these processes before running the first **mkreplica –export** command on the working schema repository. If this is not done, error messages appear during ClearCase operations to indicate that the session is no longer valid.

Mastership Policies in a ClearQuest UCM Integration

If you use the ClearQuest UCM integration with MultiSite, the default behavior is to check mastership before delivery.

ClearQuest MultiSite Integrations with Other Products

There are restrictions on using the RequisitePro, TestManager, and Rational Administrator integrations in a MultiSite deployment. If your current replica masters a ClearQuest record, but the associated Rational Project record is not mastered by the same replica, you cannot make changes to the integration information captured in a ClearQuest record (for example, adding new requirements to the requirements tab).

MultiSite Use Model

The following sections describe the different aspects of your MultiSite use model.

Languages and Code Page Support

In a ClearQuest user database, all of the data that you enter must be from the same code page. In a MultiSite environment, enforcement of a single code page can be difficult because not all languages use the same code page. For example, English and many European languages use the 1252 code page, but Japanese uses the 932 code page. Before you configure MultiSite, you must decide on the language used by the majority of users and set the data code page value of the database set to the code page for that language.

For more information about code pages and about setting the data code page value, see the *Administrator's Guide* for Rational ClearQuest.

Running multiutil Commands at Multiple Machines

By default, only one machine per site is configured to administer schema repositories and user databases and use the **multiutil** commands. This machine is designated in two ways:

- By running **multiutil activate**. The machine on which **multiutil activate** is run is configured to run subsequent **multiutil** commands.
- By running **mkreplica –import**. The machine on which **multiutil mkreplica –import** is run is configured to run subsequent **multiutil** commands.

If you want to run **multiutil** from a machine other than those at which **activate** and **mkreplica –import** were run, you must configure the machine to access the schema repository (database set) at your site.

To configure a UNIX machine, use the **cqreg add_dbset** subcommand. For more information about this command, type **man cqreg** at a UNIX prompt.

To configure a Windows machine, use the **installutil adddbset** command:

installutil adddbset *dbset-name db-vendor server-hostname*
{ *db-pathfilename***.***suffix* | *database-name* }
ro-login-name ro-login-password connection-options

dbset-name is the name of the schema repository and is always specified in the following format:

CQMS.*clan-name***.***site-name*

Connection options:

Oracle database	**HOST=***host***;SID=***sid***; server_ver=***server-version-number***;** **client_ver=***client-version-number*
All others	""

The following example shows how to use the **installutil adddbset** command to connect to the schema repository in the **boston** site of the **telecomm** clan. Notice that the *dbset-name* is **CQMS.TELECOMM.BOSTON**.

`E:\Program Files\Rational\ClearQuest>` **installutil adddbset CQMS.TELECOMM.BOSTON ORACLE bar_host cquser cquser password client_ver=7.0**

For more information about **installutil** and connecting to schema repositories, see the *Administrator's Guide* for Rational ClearQuest.

Mastership Strategy

Your plan should state which replicas will master records and other objects. After you create the replicas in the family, you can change mastership of objects. For more information, see *Enabling Independent Development: Mastership* on page 5 and *Changing Mastership of Database Objects* on page 77.

Mastership changes are communicated among replicas by the standard synchronization mechanism. Depending on your workflow, mastership changes for some objects may need to occur more often. For example, the mastership of a record may need to be transferred among replicas several times during its lifecycle.

To facilitate these mastership changes, use one of the following methods to streamline the request for mastership process for records:

- Write an e-mail rule that sends a message to the administrator of the master replica when a change in mastership is needed.
- Allow other administrators access to your replica through ClearQuest Web so they can log on and change the mastership field when needed.
- Contact the administrator at the master replica to ask for a mastership change.

Synchronization Transport Method

There are several methods for transporting update and replica-creation packets. The method you choose depends on how your sites are connected, how quickly you must transfer packets, and how important security is. For more information, see Chapter 5, *Choosing a Transport Method*.

Synchronization Pattern

The synchronization pattern for a family defines which replicas exchange update packets and the direction of exchange. Figure 2 on page 5 shows a simple synchronization pattern, involving one point-to-point update. All updates need not be point to point, however, because they are cumulative. Suppose that the following updates take place among three replicas:

Update 1: Replica **1** sends changes to Replica **2**
Update 2: Replica **2** sends changes to Replica **3**

There is no need for Replica **1** to update Replica **3** directly, because the changes from Update 1 are included in Update 2. This feature gives you flexibility in devising update strategies and patterns. For efficiency, a single update can be targeted at multiple replicas, for example, all other replicas in a family.

In general, you can implement any update topology, as dictated by organizational structures, communications/transportation costs, and so on. Figure 7 shows a simple peer-to-peer synchronization pattern, and Figure 8 shows a double-hub hierarchical pattern.

Figure 7 Peer-to-Peer Synchronization Pattern

Figure 8 Hierarchical Synchronization Pattern

Your choice of pattern depends on the following factors:

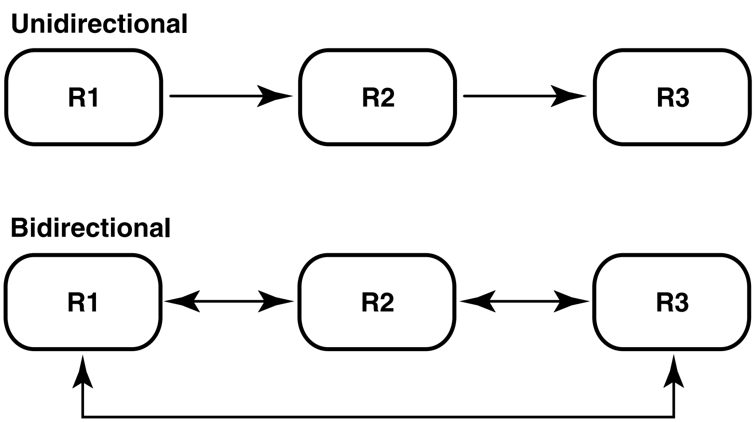
- Bandwidth between sites
- Network topology
- Latency of changes: how quickly changes made at one replica need to be received at another replica in the family
- Failure tolerance

The following sections describe unidirectional and bidirectional exchanges and the most common synchronization patterns.

Directions of Exchange

Synchronization can be unidirectional or bidirectional, as shown in Figure 9.

Figure 9 Unidirectional and Bidirectional Updating



In most cases, you will use bidirectional synchronization. Unidirectional synchronization is suitable in situations like these:

- You use a replica as a backup.
- Your company supplies information to another site (or company) for read-only use.
- A high-security development project uses the same data as a more open project. In this case, the open project sends updates to the high-security project, but no updates are sent in the other direction.

Unidirectional updates carry some risk. For example, an accidental change of mastership cannot be fixed, and restoring from a replica that does not exchange updates directly with the broken replica involves extra work. Also, you must ensure that no work is done accidentally in a read-only replica; you can do this by creating hooks.

One-to-One and Ring Synchronization

Figure 10 One-to-One Synchronization Pattern

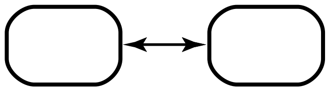
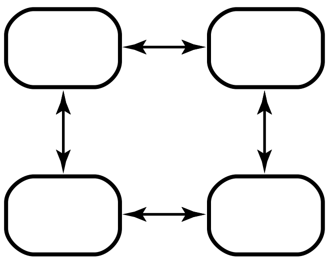


Figure 11 Ring Synchronization Pattern



The one-to-one and ring (or round-robin) patterns in Figure 10 and Figure 11 are simple patterns that are most suitable for small numbers of replicas. As the number of replicas increases, so does the amount of time for changes originating at one replica to be received at a replica at the other side of the ring.

One-to-Many Synchronization

Figure 12 Single-Hub Synchronization Pattern

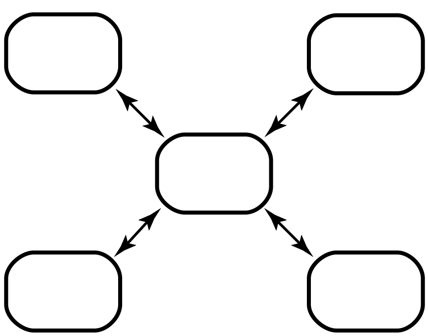


Figure 13 Multiple-Hub Synchronization Pattern

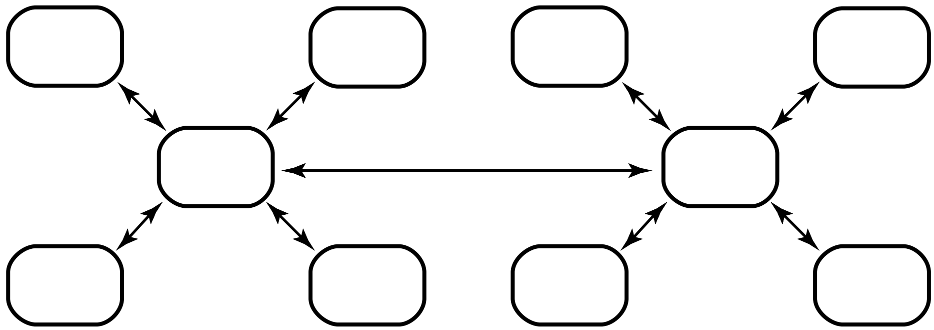
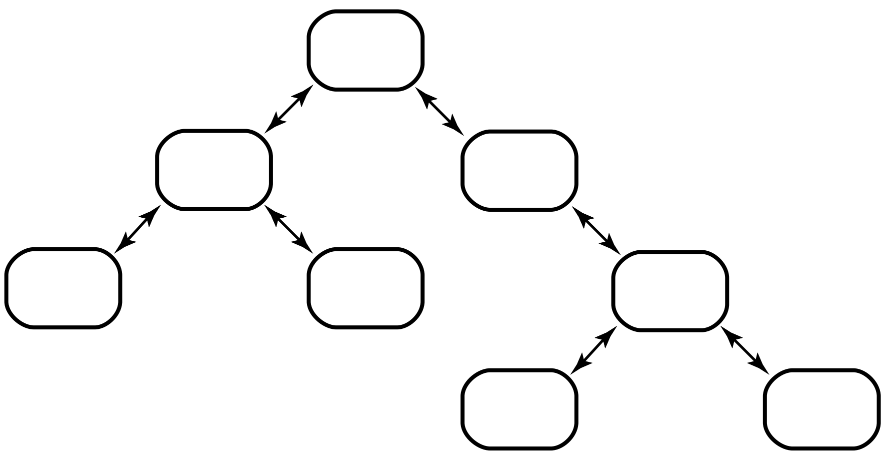


Figure 14 Tree Synchronization Pattern



In the hub patterns (Figure 12 and Figure 13), the hub replicas exchange packets with all spoke replicas. In the tree pattern (Figure 14), the root replicas exchange packets with branch replicas.

Advantages:

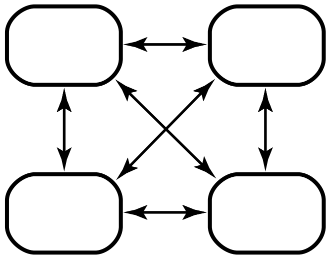
- More efficient for the spoke and branch replicas, which send to and receive from only one other replica.

Disadvantages:

- If the hub or root site goes down, all spoke/branch sites must reconfigure their pattern to continue communication.
- If you change the synchronization pattern so that replicas that did not synchronize directly now exchange packets, the first packets that are generated may be too large for the system.

Many-to-Many Synchronization

Figure 15 Many-to-Many Synchronization Pattern



In the many-to-many synchronization pattern (Figure 15), each replica exchanges packets with all other replicas

Advantages:

- For companies with few sites, this pattern keeps each replica's epoch table the most accurate for all siblings.
- If one site is unavailable, the other sites do not have to change their patterns to continue synchronizing.

Disadvantages:

- Each administrator must maintain more synchronization jobs and spend more time keeping track of packets.

Synchronization Schedule

The synchronization schedule for a family defines when replicas in the family send and receive updates. The schedule is affected by many factors, including the rate of development at different sites, the connections among sites, and whether you use MultiSite as a backup strategy.

Consider the following issues when planning your synchronization strategy:

- Rate of development

  If you schedule synchronizations frequently, you lose less work if a replica is deleted accidentally and you must restore it from backup. Also, merging is simpler because fewer changes have been made.

  Make sure that synchronizations do not overlap with backups.

- Time zone differences

  Take different time zones into account when you send an update or set up automated updates. Figure 16 illustrates synchronization among replicas in multiple time zones.

- Changes that affect both the schema repository and the user database

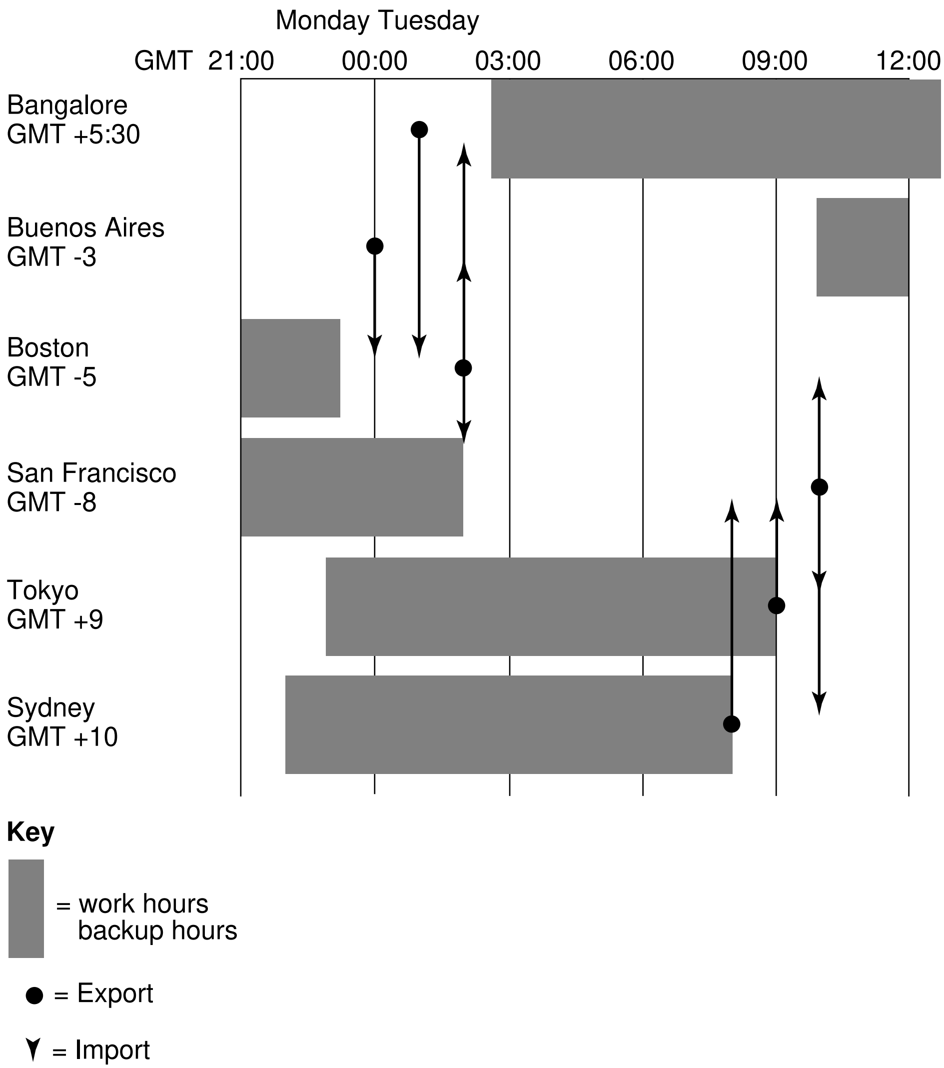
  Many changes are recorded in the schema repository and the user database, and oplog entries are created in both operation logs. We recommend that you synchronize your schema repositories first, and then synchronize the user databases.

For example, the administrators for the family in Figure 8 make the following decisions:

- The hub replicas, which undergo rapid development, synchronize every hour.
- Each hub replica synchronizes daily with its spoke replicas. Each spoke replica sends an update packet to the hub replica, and then the hub replica sends update packets back to the spoke replicas. Because these packets may be large and take a long time to import, the synchronization must not take place during working hours or during backups.
- All replica hosts use receipt handlers to import packets as soon as they are received.

Figure 16 shows the synchronization timeline for the hub-spoke updates (but not the hourly hub-to-hub updates). This timeline accounts for time zone differences and includes extra time to make sure that each synchronization phase completes before another begins.

Figure 16 A Synchronization Schedule



Backup Strategy for Databases

We recommend that you perform regular backups of your vendor databases at all sites. If the database server machine crashes or you lose the database storage area, you can restore the database from the backup copy and use the replica restoration procedure to replace any missing operations. (See *Restoring Database Replicas* on page 102.)

Scrubbing Parameters for Replicas

When a command makes a change to a replica, an entry is recorded in the replica's operation log. For more information about this mechanism, see *The Operation Log* on page 8. Also, when you export an update packet, an export_sync record is created for each target replica. These records are used by the **recoverpacket** command to reset a replica's epoch number matrix.

You can scrub oplog entries and export_sync records to reclaim disk space and database records, but you must keep them long enough to ensure that you can recover from replica failures and packet losses. The following sections give guidelines for configuring scrubbing frequency.

Oplog Scrubbing

Oplog entries must be kept for a significant period of time. They are required when the replica generates update packets. Oplog entries also may be required to help other replicas recover from catastrophic failures. If no replica can supply these entries, the replica being restored must be re-created. (See *Restoring a Replica from Backup* on page 103.) Because of the need to use oplog entries during synchronization, your synchronization strategy determines how often oplogs can be scrubbed.

By default, an oplog entry is never scrubbed. Do not change this setting until you establish the synchronization pattern in the family and verify that packets are being exported and imported successfully.

When it is safe to delete oplog entries for a replica:

1. Coordinate with other administrators to decide how long you must keep oplog entries.

   Each replica must keep entries for as long as necessary to allow **restorereplica** operations to complete successfully. The frequency with which you scrub oplog entries depends on the following factors:

   - The pattern of synchronization among replicas in the family
   - How often the replicas are synchronized

     Frequency of synchronization refers to both how often updates are exported and how often they are imported at other replicas. Also, consider setting up a verification scheme to ensure that packets are processed successfully at other replicas before any oplog entries are scrubbed.

- How often you back up the replicas

  For example, if a replica is backed up weekly at all sites and you want to be able to restore to the backup from two weeks ago, each replica must keep three weeks of oplog entries. If replicas synchronize weekly, you must assume that the weekly packet hasn't been sent to the other replica, and add another week. Finally, for extra security, add another month. The result is a scrubbing time of two months.

2 Synchronize the replicas.

3 Use the **scruboplog** command at the replica whose oplog you want to reduce. This example scrubs the oplog of the user database replica (indicated by the **PRODA** family) in the **sanfran_hub** site:

multiutil scruboplog -clan telecommunications -site sanfran_hub -family PRODA -user sfadmin -password secret -before 31-Oct-2001

Caution: If a replica's oplog entries are scrubbed before they are included in an update packet, you cannot export update packets from the replica. This is a serious error and compromises the integrity of the entire family.

export_sync Scrubbing

export_sync records are not necessary for normal synchronization operation. They are different from export event records, which also record synchronization exports.

export_sync records are date-based records used by the **recoverpacket** command to reset a replica's epoch number matrix. If you do not use this packet recovery method (because you use **lsepoch**/**chepoch**), you can scrub these records aggressively. If you use the **recoverpacket** command, you must keep export_sync records for the number of days that elapse between backups. (See *Recovering from Lost Packets* on page 96.)

export_sync records are scrubbed with the same frequency as oplog entries.

Handling Pathnames That Contain Spaces

On Windows, if the pathname of a receipt handler or a shipping order contains spaces, DOS "short name" resolution must be enabled for the file system on which the receipt handler or shipping order is located. This property is enabled by default. If this property is not enabled, the shipping server cannot invoke the receipt handler or process the shipping order.

Responsibilities of MultiSite Administrators

A MultiSite administrator must do the following:

- Help determine and implement the MultiSite use models

  When a new project is set up, the administrator works with project managers to determine which replicas master various objects. The administrator also changes mastership when necessary, and determines the appropriate data code page value for the database set.

- Monitor MultiSite replica creation and synchronization

  Administrators must check the storage bays to make sure that packets are not accumulating. Include the administrator's e-mail address in the **ADMINISTRATOR** entry in the shipping.conf file (UNIX) or in the MultiSite Control Panel (Windows).

- Monitor system log files

  Error and status messages are written to the **shipping_server_log** file on UNIX and the Event Viewer on Windows.

- Install new versions of MultiSite and new patches

  Patches and information about new versions are available on the Rational Software Web site. Install the mandatory and recommended patches for your architecture.

  Compatibility issues for versions of MultiSite are described in the *Installation Guide* for Rational ClearQuest.

- Coordinate issues with all other MultiSite administrators

  After initial setup and synchronization of replicas, administrators also must coordinate recovery efforts, which may involve exchanges of update packets, and changes of mastership, which require the administrator at the master replica to transfer mastership to the replica that needs to master the objects.

  We recommend that you create a representation of your MultiSite deployment and record information about a family. Table 4 shows an example of information that may be useful. You may also want to draw a picture of the family's synchronization pattern.

Table 4 Family Information

Replica name	Replica host	Administrator	E-mail, phone number	Location	Time zone offset
sanfran_hub	goldengate	John Cole	jcole, x1462	San Francisco, CA, USA	GMT-8
boston_hub	minuteman	Susan Goechs	susan, x3742	Boston, MA, USA	GMT-5
tokyo	shinjuku	Masako Ito	masako, x7761	Tokyo, Japan	GMT+9
sydney	taronga	Bruce Fife	bfife, x5080	Sydney, Australia	GMT+10
bangalore	ramohalli	Sonia Kumar	kumar, x2347	Bangalore , India	GMT+5:30
buenosaires	mardelplata	Juan Fangio	fangio, x4300	Buenos Aires, Argentina	GMT-3

- Ensure that replicas receive any necessary special handling

  Restoring a replica from backup is a significant event. Failure to follow the procedure described in the section *Restoring a Replica from Backup* on page 103 leads to irreparable inconsistencies among the replicas in a family.

  There are no special requirements for backing up a replica. Use the backup instructions in the *Administrator's Guide* for Rational ClearQuest.

Overview of Deployment Tasks

Before deploying MultiSite, test your plan and do a test run to ensure that your synchronization and defect-tracking process are effective.

1 Plan your deployment.

   a Review MultiSite documentation.

   b Create a MultiSite Workflow document that describes the changes and policies that will be enforced.

This document should describe the mastership policies; the replication strategy; the synchronization method, pattern, and schedule; and the workflow for users who perform tasks in a replicated environment. The document should also describe the changes necessary to implement MultiSite; for example, adding the mastership field to your schema, modifying existing hooks, creating hooks that automate transfer of mastership. Representatives of the user community should review this document.

c Determine the appropriate data code page value for your replicas. See the *Administrator's Guide* for Rational ClearQuest.

d Determine whether you need to validate and clean up your production database.

2 Test your deployment.

a Set up a test bed environment with test database instances. See the *Administrator's Guide* for Rational ClearQuest.

b Install MultiSite in the test environment.

Remember that installing ClearQuest MultiSite involves a ClearQuest upgrade. For more information, see the installation information for ClearQuest.

Note: All user databases associated with a schema repository must be upgraded to the same version of ClearQuest before you can begin using ClearQuest MultiSite.

c Apply the CharacterSetValidation package if your databases will be accessed by clients running versions of ClearQuest MultiSite earlier than v2003.06.00. See the *Administrator's Guide* for Rational ClearQuest.

d Set the data code page value for the schema repository and, if you applied the CharacterSetValidation package, set the data code page value for the user databases. See the *Administrator's Guide* for Rational ClearQuest.

e Replicate the test database, set up synchronization, make any necessary schema changes, and test your process. Be sure to test your own backup and recovery processes on the replica.

f Review the results from the test and make any necessary workflow changes.

3 Replicate the production database.

   a Set up licensing for MultiSite at the original site.

   b Install MultiSite in your production environment.

   c Back up your databases.

   d Apply the CharacterSetValidation package if your databases will be accessed by clients running earlier versions of ClearQuest MultiSite.

   e Apply schema changes to the production databases.

   f Upgrade the production databases.

   g Set the data code page value for the schema repository and, if you applied the CharacterSetValidation package, set the data code page value for the user databases.

   h Upgrade client machines.

   i Activate the production database set.

   j Replicate the production database set.

   k Validate your MultiSite use cases with the replicated databases.

   l Make any necessary schema changes, upgrade the databases, and synchronize.

   m Set up unidirectional synchronization from the original replica to the new one. Test your synchronization scripts to ensure that synchronization is occurring correctly. Do not let users access the new replica. At this point, the new replica can be removed without loss of data if problems develop.

   n Set up licensing for MultiSite at the new site. Install MultiSite on the appropriate machines at the new site.

   o Validate use cases at the new site.

   p Set up bidirectional synchronization between the production replicas.

   q Validate use cases with test data in production databases at the sites.

4 Inform users that they can begin to use the new replica. Publish workflow documentation to users at both sites.

   The new workflow rules are now in effect. We recommend that you set up a Web server for the new database replica to provide remote access for personnel at other sites.

MultiSite Command Set

4

This chapter summarizes MultiSite commands and the API functions that display or modify MultiSite information. Reference pages for the MultiSite commands are available in Chapter 11, *MultiSite Reference Pages*.

multiutil Use

The **multiutil** command is used to perform operations on replicas. The command has the following features:

- It has a set of subcommands that perform product functions, such as replica creation, synchronization, and management; mastership changes of objects; and failure recovery.
- Some subcommands and command options can be abbreviated, as indicated in the reference pages.
- You can use **multiutil** in single-command mode. For example:

  multiutil lspacket

  Also in interactive mode:

  multiutil
  `multiutil>` **lspacket**
  `multiutil>` **quit**

- Commands and options are case sensitive and must be typed in lowercase.
- The **help** command and the **–help** option display syntax summaries.

  multiutil help chreplica

  ```
  Usage: chreplica [-cl.an name] [-site name]
  ...
  ```

  multiutil chreplica –help

  ```
  Usage: chreplica [-cl.an name] [-site name]
  ...
  ```

- On Windows, the **man** command displays reference pages.

Descriptions of Subcommands

The following sections describe the different kinds of **multiutil** subcommands.

Replica Creation, Synchronization, and Management Commands

The commands in Table 5 create new replicas, change replica characteristics, and synchronize replicas.

Table 5 Replica Creation, Synchronization, and Management Commands

Command	Description
activate	Prepares a database set to be replicated
chreplica	Changes the properties of a replica
dumpoplog	Displays the contents of a replica's oplog
lspacket	Lists one or more packet files created by **mkreplica** or **syncreplica**
lsreplica	Lists one or more replicas
mkreplica	Creates a new replica
rmreplica	Removes a replica
scruboplog	Deletes oplog entries
syncreplica	Synchronizes a replica with one or more replicas in its family

Object Mastership Commands

To avoid introducing conflicting changes at different replicas, certain objects are assigned a master replica (master). The initial master of an object is the replica where the object is created. For more information about mastership, see *Enabling Independent Development: Mastership* on page 5. Table 6 lists the commands you can use to manage mastership.

Table 6 Object Mastership Commands

Command	Description
chmaster	Transfers mastership of an object
describe	Lists the master replica of an object

Failure Recovery Commands

Each replica uses an epoch number matrix to track its own state and the state of all other replicas. (Because replicas are always changing, a replica knows what changes have been made to itself, but it has only an estimate of the states of other replicas.) Each time a replica sends an update packet, it updates its own epoch number matrix, under the assumption that the packet will be delivered to its destinations and applied to the appropriate replicas. For more information, see *The Operation Log* on page 8.

Use the failure-recovery commands in Table 7 when this assumption of successful delivery does not hold true.

Table 7 Failure-Recovery Commands

Command	Description
chepoch	Changes a replica's epoch number matrix
lsepoch	Lists a replica's epoch number matrix
recoverpacket	Resets a replica's epoch number matrix so lost packets are resent (required when a packet is lost or unusable)
restorereplica	Restores a replica from backup. This command places a replica in a special state, in which it sends epoch number matrix corrections to other replicas. The replica cannot be used for normal development work until it receives special updates that inform it of the current states of other replicas.

multiutil Utility Commands

These **multiutil** commands perform miscellaneous tasks.

Table 8 multiutil Utility Commands

Command	Description
cd	Changes current working directory
exit	Ends interactive **multiutil** session
help	Displays **multiutil** command syntax
man	Displays a reference page on Windows. On UNIX, displays command syntax.
quit	Ends interactive **multiutil** session

Additional MultiSite Commands

The MultiSite commands that are not **multiutil** subcommands are listed in Table 9. These commands are located under the installation directory for Rational ClearCase.

Table 9 Additional MultiSite Commands

Command	Location under *ccase-home-dir*	Description
mkorder	etc (UNIX) bin (Windows)	Creates shipping order for use by store-and-forward
notify	bin	Mail program for store-and-forward
shipping_server	etc (UNIX) bin (Windows)	Store-and-forward packet transport server

MultiSite API Functions

You can use API functions in hooks and external applications to determine whether you are working with a replicated database and whether your current replica masters the record or object you want to modify.

Table 10 describes three API methods you can use with MultiSite. For a complete list of API commands, see the *API Reference* for Rational ClearQuest.

Table 10 MultiSite API Functions

API method	Associated object	What it does
SiteHasMastership	Entity Workspace User	Returns a value that indicates which replica masters a record, Workspace item, user or group
GetSiteExtendedName	Entity Workspace User	Returns the value of the **ratl_keysite** name, which helps you determine which records, users or groups have naming conflicts and must be renamed
GetLocalReplica	Session object	Lists replica information. You can use this method to determine whether the database you are working with is a replica.

Specifying Replicas in Commands

When you specify replicas in a **multiutil** command, you must indicate site, family and clan, when necessary. If there is only one clan at your site, the **–clan** argument is optional. The **–site** argument is also optional, except when creating replicas.

For example, the following command specifies the **boston_hub** replica in the **PRODA** family, which is part of the **telecomm** clan.

multiutil lsreplica -clan telecomm -site boston_hub -family PRODA -user susan -password passwd

Choosing a Transport Method 5

This chapter describes the methods for transporting packets between replicas. The method you choose depends on connectivity between replicas. If your replicas do not have IP connectivity, you must use a file-based method. If your replicas have connectivity, you can use the store-and-forward facility of Rational ClearQuest MultiSite.

Table 11 lists the recommended methods for various situations.

Table 11 Choosing a Packet Transport Method

Your situation	Recommended methods
Sites are connected with high-speed lines	Store-and-forward
One or more sites have firewalls	File-based methods (e-mail, **ftp**, physical media), store-and-forward
Must transfer packets quickly	File-based methods (e-mail, **ftp**), store-and-forward
No electronic connection between sites	File-based methods (physical media)

File-Based Methods

These transport methods include e-mail, **ftp**, and physical media (like CDs, magnetic tapes, and diskettes).

Using Electronic Mail

You can use an existing electronic mail mechanism as the transport method for packets. On the sending end, compress and encode the packet; then send the resulting data to a specific mail alias at the receiving site. On the receiving end, redirect the mail alias to a script that decodes and decompresses the incoming information. To ensure that a mail message is not too large to be delivered, you can specify the maximum size for a packet by using the **–maxsize** option, the shipping.conf file (UNIX), or the MultiSite Control Panel (Windows).

Advantages:

- Transport mechanism is well understood and widely available.
- Little effort is required from the system administrator.

Disadvantages:

- No control over routing of data.
- Possibility that messages can be intercepted or lost without notification.
- Less efficient than **ftp** or store-and-forward.

Notes:

- You can write scripts to automate e-mail transport. The sending script creates the packets, compresses and encodes them, and divides them into multiple small packets so they are not too big for the e-mail process. The script must mark the multiple packets with the correct sequencing. The script then sends the packets to an address at the target location or replica.

  At the target location, the account that receives the packets redirects or pipes them to a process that reassembles, decodes, and uncompresses them and places them in the replica's storage bay.

  MultiSite import commands handle out-of-sequence and missing packet problems, so your scripts do not have to address these issues.

- Using **ssh** and **scp** (secure shell and secure copy) provides a secure way to move files through firewalls.

- For security, you must encrypt the packets.

Using FTP

The **ftp** utility can transport packets between replicas. On the sending end, the MultiSite administrator or a script creates and compresses the packet, and uses **ftp** to transfer the file to a location that is accessible by MultiSite administrators at other sites. Scripts at receiving sites poll the drop site, looking for any new files. When new files arrive, the scripts retrieve them using **ftp**, decompress them, and process them.

Advantages:

- Transport mechanism is well understood and widely available.
- More reliable and efficient than electronic mail.

Disadvantages:

- Use of a drop site is required.
- Polling of the drop site is required.
- More complicated to implement, because of the interactive nature of the **ftp** utility.
- More administration is required because a third system (the drop site) is used.

Using Physical Media

You can create packets as files, write them to a CD, magnetic tape, or diskette, and then send the media to another site. The **mkreplica** and **syncreplica** commands include the **–out** option, which places packets in physical files.

When you use a file-based method for transport, you may need to use the **–maxsize** option to ensure that the file is a manageable size.

Store-and-Forward

The MultiSite store-and-forward facility (the shipping server) is a file-transfer service that automates the transport phase of replica creation and synchronization. It can handle packets of any size, can route files through a series of MultiSite hosts (one hop at a time), and includes support for handling data-communications failures. This is how the store-and-forward process works:

1. During the export phase, a packet file and a shipping order file are created. The shipping order file contains delivery instructions for the packet.
2. The packet and shipping order are stored in one of the storage bay directories on the synchronization server associated with a ClearQuest database replica.

   If the packet is associated with a storage class, the packet is stored in the storage bay specified by the storage class. You can define storage classes in the shipping.conf file on UNIX and the MultiSite Control Panel on Windows.
3. The shipping server uses the instructions in the shipping order to transfer the packet file from the storage bay at the local site to the corresponding bay on a host at another site.
4. If necessary, the shipping server on the receiving host sends the packet to its next destination.

Directories for Packets

Each storage class has storage bays and return bays, which are directories that hold packets. Storage bays are used for normal shipping operations, and return bays are used for packets that could not be delivered successfully.

Each storage bay and return bay directory contains two subdirectories, incoming and outgoing, which hold the packets and their corresponding shipping order files. Shipping operations look in these directories for packets.

Note: On Windows, the amount of available space on the disk partition where the bays are located must be at least twice the size of the largest packet that will be stored in the bays. There may be two copies of the same packet in the bay at one time: one on its way to another destination and another waiting to be applied to the replica on the host.

When you install the Rational Shipping Server on a host, the **–default** storage class is created, along with its storage and return bays. The storage bay is named ms_ship and the return bay is named ms_rtn. The incoming and outgoing directories in each bay are also created. When you use the MultiSite Control Panel (Windows) to create a new storage or return bay, the bay and its subdirectories are created. On UNIX, you must create the bays and their incoming and outgoing subdirectories and then specify the bays in the shipping.conf file.

Packet Transport

An explicit command, manual or automated, invokes the shipping server on the sending host. The shipping server process contacts the **albd_server** process on the receiving host, which in turn invokes the shipping server on the receiving host in receive mode. After a TCP/IP connection has been established between the sending and receiving invocations of the shipping server, the file is transferred.

Store-and-Forward Issues

The following sections describe issues to consider when you use the store-and-forward method.

Communication Between Replica Hosts

The hosts must be able to communicate with each other. If your network uses host names, the sending host must be able to resolve the receiving host's name to an IP address. To accomplish this, you may have to update the hosts file, **hosts** NIS map, or Domain Name Service. To verify TCP/IP access, use **rcp** on each sending host to copy a file to the receiving hosts or use store-and-forward to send a packet (see *Submitting Packets to Store-and-Forward* on page 45).

Note: If hosts in your network are known only by their IP addresses, you can use the IP addresses instead of host names, and no resolution is necessary.

Limiting the Size of a Packet

The **mkreplica** and **syncreplica** commands fail if they try to create a packet larger than the size supported by your system. To prevent this problem and improve reliability, use the **–maxsize** option to divide the packet into multiple packets:

multiutil mkreplica –export –maxsize 1g ...

multiutil syncreplica –export –maxsize 500m ...

You can also specify maximum packet sizes in the shipping.conf file (UNIX) or MultiSite Control Panel (Windows).

For information about default packet size limits, see the **mkreplica** reference page.

Configuring the Store-and-Forward Facility

The settings for the store-and-forward facility are host specific. You can specify locations of storage and return bays, routing information to support multihop packet delivery, specifications to handle failure-to-deliver situations, receipt handlers, and so on.

Before you use store-and-forward, verify that you have the appropriate disk space, and configure the shipping.conf file or the MultiSite Control Panel and create storage classes for packets.

For more information about specifying settings, see the **shipping.conf** reference page on UNIX or the **MultiSite Control Panel** reference page on Windows.

Submitting Packets to Store-and-Forward

When you generate a replica-creation or update packet, you can specify that the store-and-forward facility must deliver it. Both **mkreplica** and **syncreplica** support the following options:

- The **–fship** option places the packet files and shipping order files in one of the host’s storage bays and runs the shipping server to send the packet files to their destination host or to route them to an intermediate host.
- The **–ship** option places the packet files and shipping order files in a storage bay, but does not invoke the shipping server. The packet files are sent the next time the shipping server polls the bay.

Differentiating Packets with Storage Classes

You can configure the store-and-forward facility to handle packets in different ways. Each packet can be assigned to a storage class, and each storage class can have its own storage bay, return bay, and expiration period.

Note: On UNIX, a storage class can be assigned several storage and return bays; in this case, the shipping server uses the size of the packet to select one of the bays. Conversely, several storage classes can share one or more bays.

The default storage class for packets from schema repository and user database replicas depends on the command you're using. The **mkorder** and **shipping_server** commands use the **–default** storage class, which is created when the Rational Shipping Server is installed. All **multiutil** commands that use the **–sclass** argument use the **cq_default** storage class, which is not created during installation.

You can use multiple storage classes to segregate the packets for replicas that belong to different clans. By adjusting the operating system permissions on the storage bay directories, you can protect the packets from unauthorized use. You can also use a separate storage class when you use the store-and-forward facility to transfer non-MultiSite files between sites.

If you are using the store-and-forward facility to transport packets from VOB replicas and from ClearQuest database replicas, you must use different storage classes. Because the **mkorder** and **shipping_server** commands are used for both ClearCase MultiSite and ClearQuest MultiSite, you must specify the storage class when you use these commands on a packet from a ClearQuest replica. Also, if you do not create the **cq_default** storage class, you must use the **–sclass** option in **multiutil** commands and specify a ClearQuest MultiSite storage class.

Follow these guidelines when you create a storage class:

- The storage bay must be unique. Do not use the same name or directory that you use for packets from VOB replicas.
- The directory you specify must be on a partition that has enough room for the packets.
- Storage class names are case sensitive. We recommend that you use only lowercase letters in names of storage classes, or define a case convention for all storage classes you create.

Setting Up an Indirect Shipping Route

The shipping order for a packet includes the host name of the packet's final destination or several such host names. By default, the store-and-forward facility sends the packet directly to its destination host. You can specify that the packet must be sent to an intermediate host by associating it with a routing hop in the shipping.conf file (UNIX) or in the MultiSite Control Panel (Windows).

For example:

- On a UNIX host, the shipping.conf file includes this line:

  ```
  ROUTE  sydney_fw    sanfran_hub boston_hub tokyo
  ```

- On a Windows host, the Routing Information section in the MultiSite Control Panel specifies host **sydney_fw** in the **Next Routing Hop** box and hosts **sanfran_hub**, **boston_hub**, and **tokyo** in the **Destination Hostnames** box.

Any packet whose final destination is host **sanfran_hub**, **boston_hub**, or **tokyo** is forwarded to host **sydney_fw**. At this point, the local host has completed its task, and responsibility for delivering the packet now belongs to **sydney_fw**. Host **sydney_fw** can transmit the packet to its final destination directly, or send it to yet another intermediate host, depending on the settings in its shipping.conf file or in the MultiSite Control Panel.

Note: In a multihop transmission, using the **–fship** option on the original host causes the first hop to occur immediately. Subsequent hops occur when the shipping server is invoked on the intermediate hosts, which may not be immediately after the packets are received.

Retries, Expirations, and Returned Data

The shipping server makes one attempt to transmit a packet to another host. If the packet cannot be transmitted (for example, because the receiving host is unavailable), the shipping server generates an error message and log file entry and exits. You can set up a retry scheme to control its frequency:

- After successful transmission of a packet, the shipping server deletes the packet and its shipping order. After a failure, the packet and shipping order remain in the storage bay.
- **shipping_server –poll** transmits all packets it finds in one or more storage bays. Thus, any packets that remain after a transmission failure are sent (if possible) by the next invocation of **shipping_server –poll**.

Attempts to transmit an undelivered packet can continue indefinitely, through repeated invocations of the **shipping_server** command. However, you usually want to fix problems with failed transmissions instead of letting the attempts continue. Accordingly, each shipping order can include an expiration date-time, specified with one of the following:

- The command option **–pexpire**

- (UNIX) An **EXPIRATION** entry in the shipping.conf file on the sending host
- (Windows) A **Packet Expiration** value in the MultiSite Control Panel on the sending host

By default, shipping orders expire 14 days after they are created.

When the shipping server encounters a shipping order that has expired, it does not attempt to transmit the corresponding packet to its destination. Instead, it does the following:

- It modifies the shipping order to return the packet to the original sending host, where it is placed in a return bay.
- It sends an electronic mail message to one or more addresses on the original sending host. (Another message is sent when the returned packet arrives at the original sending host.)

The return trip may involve multiple hops, as described in *Setting Up an Indirect Shipping Route* on page 46. During such a trip, a packet is placed in the return bay of each intermediate host. Each hop is handled by **shipping_server –poll**, which processes a host's return bay in addition to its storage bays. The expiration time for a packet's return trip is 14 days; a packet that cannot be returned in that interval is deleted.

Setting a Timeout Period for Unreachable Hosts

If the shipping server tries to send a packet to a target host and determines that the host is unreachable, it creates a file in the /var/adm/rational/clearcase/shipping/ms_downhost directory (UNIX) or the *ccase-home-dir*\var\shipping\ms_downhost directory (Windows). The name of the file is the name of the unreachable host.

If one of the following parameters is set, the shipping server checks the directory for target hosts during future shipping operations:

- (Windows) **Timeout for Unreachable Host (minutes)** value in the MultiSite Control Panel
- (UNIX) **DOWNHOST-TIMEOUT** setting in the shipping.conf file or the SHP_DOWNHOST_TIMEOUT_RETRY environment variable (If both parameters are set, the shipping server uses **DOWNHOST-TIMEOUT**.)

If the target host is found in the ms_downhost directory, and the difference between the current time and the last modification time of the file is less than the timeout setting on the shipping server host, the shipping server does not try to send packets to the target host. If the difference is equal to or greater than the timeout setting, the shipping server tries to send packets to the target host. If the timeout setting is not set, the shipping server attempts to send the packet to the target host. (Each attempt to send a packet to an unreachable host takes about 30 seconds.)

Error Notification in a Mixed Environment

If a packet is delivered through a Windows host on which e-mail notification is not enabled, a failure on that Windows host means that no notification message is sent by electronic mail. Instead, a message is written to the event log; this message contains a request that the appropriate users be informed of the failure. For information about enabling e-mail notification, see the **MultiSite Control Panel** reference page.

Sending Files That Are Not Packets

You can use the store-and-forward facility to send any file if you create a shipping order for the file with the **mkorder** utility. You can send the file immediately or wait for the shipping server to send it.

- To send a file immediately, use the **–fship** option with **mkorder**:

  /opt/rational/clearcase/etc/mkorder –data /usr/rptgen/brdcst.0702 –fship –copy boston_hub tokyo

- To store the file in a shipping bay so that the shipping server will send the file the next time it runs, use the **–ship** option:

  /opt/rational/clearcase/etc/mkorder –data /usr/rptgen/brdcst.0702 –ship –copy boston_hub tokyo

  Note: The shipping order must be located in the same directory as the file.

After you invoke the **mkorder** command, you can delete the original file.

If a file with the same name already exists on the receiving host, the file you send is renamed to *filename*_1. If you transmit another file with the same name, it is renamed to *filename*_2, and so on.

Using Store-and-Forward Through a Firewall

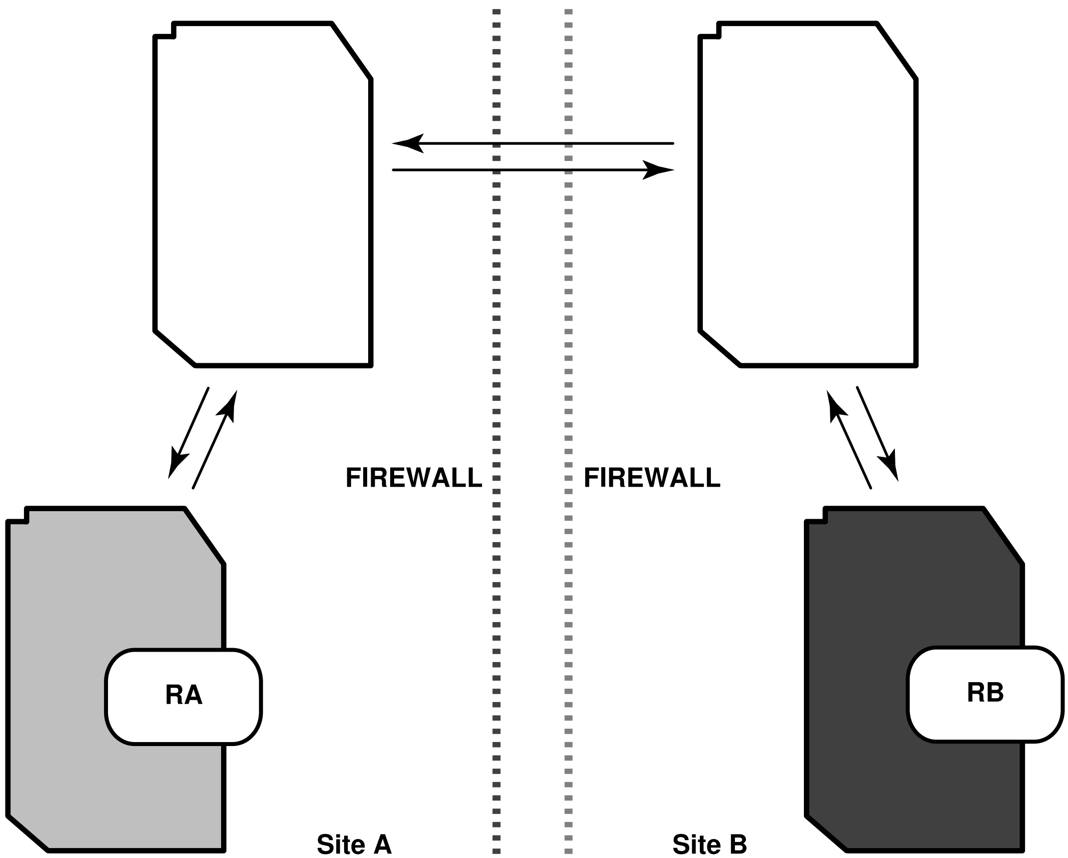
By default, the store-and-forward facility (the shipping server) cannot operate through a firewall. Passing through a firewall is usually accomplished by granting access to specific ports for certain IP addresses. Because the shipping server picks any available port number on the sending and receiving replica hosts to make the connection, there is no single port number (or even small range of port numbers) to which special access can be granted.

If your site uses a firewall, you can set up an “exposed host,” a host that you configure to communicate through the firewall and on which you install the shipping server software. You configure the shipping servers on the synchronization servers at your site to send packets to the exposed host, and the shipping server on the exposed host forwards the packets to hosts on the other side of the firewall. To maximize security on the exposed host, you must specify the range of port numbers that the shipping server can use.

Note: To enhance site security, we recommend that you install the shipping server on an exposed host only if other transport methods are unsuitable for your site. For information about other methods, see *File-Based Methods* on page 41.

Figure 17 is an example of an exposed host configuration. The exposed hosts communicate through the firewall. The store-and-forward software is installed on them, but ClearQuest software is not installed on them.

Figure 17 Store-and-Forward Configuration



Firewall Issues

Before installing the shipping server on an exposed host, consider the following issues:

- Shipping bays can be filled.

  Using the shipping server on an exposed host enables anyone coming in from the network to fill shipping bays on the local network, on any machine where a shipping server is available. To avoid full disks and the related problems:

  - Create all shipping bays in the local network on their own partitions, so that filling the bays does not degrade system performance.
  - Install the shipping server only on machines that need it: synchronization servers and machines used by administrators.

- Packets are susceptible to snooping.

  In normal update packets, information is not encoded. Therefore, anyone shipping packets across an unsecured network must encrypt the packets. Also, the format of an update packet is not very complicated; a dedicated programmer could figure out the format and create a packet with operations that damage a schema repository or user database. Encrypting the data makes this kind of attack much more difficult.

Configuring Your Firewall to Limit Access

We recommend that you specify the ports to which programs can connect and the IP addresses that are allowed to access the firewall. Limiting the allowed port numbers and IP addresses limits the possibility that unauthorized machines can breach the firewall.

You must allow access to the following ports on the exposed host:

- TCP port 371 (**albd_server** port)
- The range of ports that you specified with the CLEARCASE_MIN_PORT and CLEARCASE_MAX_PORT environment variables (see *Controlling Ports Used by albd_server and shipping_server* on page 52)

You must allow access through the firewall for IP addresses of hosts that send packets through the firewall to the exposed host at your site.

For information about configuring your firewall, see the documentation for your firewall.

Installing the Shipping Server on an Exposed Host

On UNIX, the ClearCase Product Family installation includes an option to install only the shipping server software. Follow the instructions in the *Installation Guide* for the ClearCase Product Family and select only the **ClearCase MultiSite Shipping Server-only Installation** option. Do not install ClearCase on the exposed host.

On Windows, use the Rational Shipping Server installation option.

Controlling Ports Used by albd_server and shipping_server

The environment variables CLEARCASE_MIN_PORT and CLEARCASE_MAX_PORT specify the range of port numbers that the **albd_server** and the shipping server can allocate for communication purposes. When the shipping server needs to assign a port number, it starts with the value of CLEARCASE_MIN_PORT and continues through the range until it reaches CLEARCASE_MAX_PORT. If a port in the range cannot be allocated, the shipping server sleeps and tries the ports again.

When the shipping server on the sending host detects that the port environment variables are set, it tries to use TCP to make the connection with the **albd_server** on the receiving host. If this connection fails, the shipping server tries UDP. Therefore, if you have TCP connectivity, you do not have to enable UDP or open UDP ports on the exposed host.

Running an individual shipping server does not require more than two ports at a time. When there are multiple requests to be sent, the shipping server forks. Child processes handle individual requests. The shipping server starts no more than 10 child processes (and starts that many only if there are 10 requests to process simultaneously), so the maximum range is 20 ports. If the range is smaller, it may result in failed attempts, which can be retried later.

Specifying Port Values

The value range for CLEARCASE_MIN_PORT is 1024 through 65534, and the value range for CLEARCASE_MAX_PORT is 1025 through 65535. The value of CLEARCASE_MAX_PORT must be greater than the value of CLEARCASE_MIN_PORT.

Note: We recommend that you use the range 49152 through 65535, which is the Dynamic/Private Port Range.

To specify minimum and maximum port values on UNIX, set the CLEARCASE_MIN_PORT and CLEARCASE_MAX_PORT environment variables in the following places:

- The shipping.conf file on the exposed host. For more information, see the **shipping.conf** reference page.

- The clearcase script on the exposed host:

  a Edit the file *ccase-home-dir*/etc/clearcase.

  b Add the following lines, replacing *min-port* and *max-port* with your minimum and maximum port values. These lines must precede the section that starts the **albd_server**.

```
#
# Set values for minimum and maximum port numbers
#
CLEARCASE_MIN_PORT=min-port
CLEARCASE_MAX_PORT=max-port
export CLEARCASE_MIN_PORT
export CLEARCASE_MAX_PORT
```

To specify minimum and maximum port values on Windows:

1. On the exposed host, open Control Panel and click the **System** icon.
2. Create two system environment variables, CLEARCASE_MIN_PORT and CLEARCASE_MAX_PORT, and specify their values.

Checklist for Using Store-and-Forward Through a Firewall

This checklist summarizes the steps you must follow to use store-and-forward through a firewall.

1. Determine the port ranges that the shipping server can use and the IP addresses of the hosts that will send packets to your site's exposed host.
2. Configure your firewall to limit the allowed port numbers and IP addresses. Remember that you must allow access to TCP port 371 in addition to the port ranges.
3. Install the shipping server software on the exposed host.
4. Set the CLEARCASE_MIN_PORT and CLEARCASE_MAX_PORT environment variables.
5. On each replica server host at your site, specify the exposed host as the next-hop host for packets sent to other sites. For example, your company has three sites (SiteA, SiteB, SiteC), each with one exposed host running the shipping server (**SSA**, **SSB**, **SSC**), and three replica server hosts.

   On UNIX, edit the shipping.conf file and add **ROUTE** options. For example, on each replica server host at SiteA:

```
ROUTE SSA SiteB_host1 SiteB_host2 SiteB_host3 SiteC_host1
SiteC_host2 SiteC_host3
```

On Windows, open the MultiSite Control Panel and set the appropriate values in the Routing Information section. For example, on each replica server host at SiteA, the **Next Routing Hop** is **SSA** and the **Destination Hostnames** are **SiteB_host1**, **SiteB_host2**, **SiteB_host3**, **SiteC_host1**, **SiteC_host2**, and **SiteC_host3**.

6 On the exposed host, edit the shipping.conf file and add **ROUTE** options for the next destination of the packets.

Using the same example as in Step 5, on the exposed host at SiteA, you add the following **ROUTE** options to the shipping.conf file:

```
ROUTE SSB SiteB_host1 SiteB_host2 SiteB_host3
ROUTE SSC SiteC_host1 SiteC_host2 SiteC_host3
```

On the exposed host at SiteB, you add the following **ROUTE** options to the shipping.conf file:

```
ROUTE SSA SiteA_host1 SiteA_host2 SiteA_host3
ROUTE SSC SiteC_host1 SiteC_host2 SiteC_host3
```

On the exposed host at SiteC, you add the following **ROUTE** options to the shipping.conf file:

```
ROUTE SSA SiteA_host1 SiteA_host2 SiteA_host3
ROUTE SSB SiteB_host1 SiteB_host2 SiteB_host3
```

Replication and Synchronization

Creating Database Replicas

6

This chapter describes how to plan and create database replicas. Before creating a replica, you must make decisions about mastership and methods of packet delivery. Be sure to read Chapter 3, *Planning a MultiSite Implementation*.

Overview of Replica Creation

Replica creation includes these phases:

1. Store-and-forward configuration: If you will use store-and-forward, you must configure the shipping.conf file (UNIX) or MultiSite Control Panel (Windows) at the exporting and importing replicas. See *Configuring the Store-and-Forward Facility* on page 45.
2. Activation: If you are replicating a database set for the first time, you must activate the database set.
3. Export: Enter a **mkreplica –export** command, which creates a new replica object and a replica creation packet.
4. Transport: Send the replica-creation packet to one or more other sites.
5. Database creation: At the location where the new replica is to be created, create empty vendor databases for the schema repository and each user database replica.
6. Import: At the location where the new replica is to be created, import the replica-creation packet by using **mkreplica –import**.

This procedure is the same for all methods of packet delivery and for all platforms.

The following sections describe the phases in more detail.

Activating a Database

Before you can create the first replica of a database, you must activate the database set (a schema repository and its associated user databases). After you activate a database set, you can replicate it multiple times. A database set needs to be activated only once.

When you activate a database set, you assign it a clan name and site name and specify its synchronization server (the host that handles packets).

You activate a database set with the **activate** command. For more information, see the **activate** reference page.

Exporting a Replica-Creation Packet

You create a replica by generating replica-creation packets and sending them to the sites that will host the database replicas. You do this with the **mkreplica –export** command. For more details about using **mkreplica –export**, see the **mkreplica** reference page.

During the export phase of replica creation, the replica creation command locks the database while copying it. The database is locked for the entire length of time the command runs; logins are not allowed.

Before running the **mkreplica –export** command on a database:

- Verify that no users are logged in to the database. If users are logged in to the database during a **mkreplica –export**, any changes they make even after the database is unlocked are lost.
- Upgrade the database to use the most recent version of the schema.
- Verify that the data code page value is set to the code page used by the site that will import the replica creation packet. If the code pages are incompatible, packet import fails. For more information about the data code page value, see the *Administrator's Guide* for ClearQuest.

The time required to create the packet depends on the size of the database and can be up to twice as long as the time required to make a copy of the database or run a backup procedure. Therefore, you must schedule the export phase of replica creation during nonbusiness hours for your site. You must also cancel any scheduled backups for the duration of the export phase.

In this example, a replica of the **PRODA** user database and its schema repository is created. The new site is named **sanfran_hub** and uses the synchronization server **goldengate**. This example uses store-and-forward to ship the replica-creation packets.

multiutil mkreplica –export –clan telecomm –site boston_hub –family PRODA –user susan –password passwd –maxsize 50m –fship –workdir c:\temp\packets –sclass cq_default goldengate:sanfran_hub

Creating Empty Vendor Databases

Before you import a replica-creation packet, you must create an empty vendor database to contain the replica data. For instructions about creating vendor databases and a list of supported databases for MultiSite, see the installation information for Rational ClearQuest.

Note: To prevent data corruption, the code page, or character set, of your vendor databases should match the data code page value of your database set. For more information about code pages and the data code page value, see the *Administrator's Guide* for Rational ClearQuest.

Caution: Do not create a ClearQuest database before receiving a database replica. A ClearQuest database is created when you import a replica packet into an empty vendor database. If you have created a ClearQuest database with the Maintenance Tool or ClearQuest Designer, the replica import fails.

Importing a Replica-Creation Packet

When you import a replica-creation packet, you import raw data into existing vendor databases. Replica-creation packets do not contain databases; they contain the metadata and record data that form a database. Packets are not vendor specific and can be used to create a schema repository or user database from any supported vendor database.

When you import a replica, remember the following:

- You must use the site name that was used in the export command. You cannot change the site name when you import a replica-creation packet.
- Import fails if the code page used at the importing site does not match the data code page value of the replica being imported.
- Initially, you can access the replica with **multiutil** commands only from the machine on which you ran the first **mkreplica –import** command. If you want to run subsequent **multiutil** commands on a different machine, you must configure that machine to access the replica with **multiutil**. See *Running multiutil Commands at Multiple Machines* on page 20.

To import a replica-creation packet, run the **mkreplica –import** command to import the replica data into the empty vendor database you've created. Enter the database parameters and login information for both the schema repository and user database you are importing.

Adding Additional Replicas

If you want to add a new user database replica to an existing site, you do not need to create a vendor database for the schema repository. If the new replica is in the same clan, the **mkreplica –import** command associates the new replica with the existing schema repository in the site. See the **mkreplica** reference page.

Recovering from a Failed Import

If the **mkreplica –import** process is interrupted or fails for any reason, follow these steps:

1. Verify when the import failed. **mkreplica –import** generates error messages containing this information.
2. Delete the vendor database where the import failed and create a new vendor database:
   - If the import failed during the import of the schema repository, delete the vendor database for the schema repository and create a new one.
   - If the import failed after a successful import of the schema repository, delete the vendor database for the user database replica and create a new one.
3. Run **mkreplica –import** again.

Replica Creation Scenario

For the example in this section, the company's software development takes place in Boston and in a new development office in San Francisco. Work is about to begin on a new release.

Relevant characteristics of the two sites:

Location	Synchronization server	Replica name (site name)
Boston	**minuteman**	**boston_hub**
San Francisco	**goldengate**	**sanfran_hub**

Prerequisites

Before you create a new replica, you must perform these steps at the original site:

1 Make sure MultiSite licenses are installed.

After you enter the **activate** command for a database set, users at the original database cannot access it without a MultiSite license (in addition to a ClearQuest license).

2 When you create a replica of a specific database for the first time, all users must log off the database.

The **mkreplica –export** command locks the database after you start to export the replica. All users must log off before the procedure begins and log on after it is finished. Data is lost if sessions are left open during the replication process.

3 If you use the ClearCase/ClearQuest UCM integration, you must terminate all **cqintsrv** processes before running the first **mkreplica –export** on the schema repository.

4 Determine the size of the user database and schema repository.

Replica-creation packets can be four times as large as the respective databases. Verify that the working directory you use has enough free space. You must have write permission for the directory, and the directory you specify must not exist.

Activating the Database Set

The following command activates the database set in Boston. It names the clan **telecomm** and the site **boston_hub**, and it specifies **minuteman** as the synchronization server.

multiutil activate –user susan –password passwd –clan telecomm –site boston_hub –host minuteman

Export Phase

In Boston, perform the following steps.

1 Use the **mkreplica –export** command to create the replica for San Francisco.

The following command creates the **sanfran_hub** replica of the **PRODA** user database in the **telecomm** clan. It also creates the **sanfran_hub** replica of the schema repository in the **telecomm** clan. The synchronization server for the new site is **goldengate**. The administrator uses the **–fship** option to send the packet immediately using the Rational Shipping Server.

multiutil mkreplica –export –clan telecomm –site boston_hub –family PRODA –user susan –password passwd –maxsize 50m –fship –workdir c:\temp\packets –sclass cq_default goldengate:sanfran_hub

2 Back up the newly replicated database.

This backup records the fact that the database is replicated. If you try to restore a database from a backup copy that was made before the database was replicated, the replica restoration procedure fails. (Although the **restorereplica** command may succeed, you cannot import update packets from other replicas because the original database is marked as unreplicated.)

Transport Phase

3 Send the replica-creation packet to the new site. This process differs depending on the options you used in Step 1:

- If you used **–fship**, the packet was sent to the new site immediately.
- If you used **–ship**, you must run **shipping_server** to send the packet to the new site. For example:

  shipping_server –sclass cq_default –poll

- If you used **–out** to write the packet to a file, you must transport the file to the new site.

Import Phase

These steps take place in San Francisco, which has no replicas in the clan.

4 If you used store-and-forward, verify the packet's arrival by entering the **lspacket** command on the synchronization server.

multiutil lspacket –short

```
Multiutil:Packet
'd:\temp\ms_ship\incoming\mk_sanfran_hub_21-May-01_19-28-01.xml'...
```

5 Create empty vendor databases for the new schema repository and user databases.

6 Enter the import form of the replica-creation command.

In the **mkreplica –import** command, you must specify the pathname of the incoming packet as listed by the **lspacket** command. For example:

multiutil mkreplica –import –site sanfran_hub –repository ORC1 –vendor ORACLE –dbologin orcadmin password –connectopts host=sanfran_dbserver;SID=ORC1;server_ver:8.1;client_ver=8.0;log_type=long –database ORC1 –vendor ORACLE ORC1 –dbologin orcuser password –connectopts host=sanfran_dbserver;SID=ORC1;server_ver:8.1;client_ver=8.0;log_type=long –comments "Importing the initial replicas of the PRODA database and its schema repository for the San Francisco site in the telecommunication clan" d:\temp\ms_ship\incoming\mk_sanfran_hub_21-May-01_19-28-01.xml

7 Confirm that import was successful, and then delete the replica-creation packet. (Update packets are deleted automatically.)

8 Begin development.

Users in San Francisco can access the new replica in the same way they access an unreplicated database.

Synchronizing Replicas

7

This chapter describes the process of synchronization. Synchronization uses the same export-transport-import procedure that is used during replica creation:

1. Export: At one replica, a **syncreplica** (synchronize replica) command is invoked with the **–export** option. This creates a packet of data.
2. Transport: The packet is sent to one or more other replicas.
3. Import: At the other replicas, a **syncreplica** command is invoked with the **–import** option. This applies the changes in the packet to an existing replica.

The **syncreplica** command is optimized for performance; it creates a packet that contains only the information required to update the target replicas specified on the command line.

Assumption of Successful Synchronization

The export and import phases of synchronization always occur at different times. A sending replica does not require acknowledgment from a sibling replica that a packet has been received and processed successfully. Instead, the sending replica assumes success. This assumption enables an optimization: subsequent updates from a replica do not include the data sent in previous updates.

If a failure does occur (for example, a packet is lost in transit or a CD is unreadable at the sibling replica), you must adjust the epoch numbers at the sending replica to enable the lost data to be resent. For more information, see Chapter 10, *Troubleshooting MultiSite Operations*.

Applying Packets That Include Schema Updates

Packets exported from the working schema repository may include new schema revisions that require database upgrades at other sites. In this case, the **syncreplica –import** command cannot finish its process until the user database replica is upgraded to the new schema version.

If a packet contains updates for both the schema repository and the user database replica, **syncreplica –import** stops the process and prints the following message:

packet_name `is destined for schema revision` *revision_number*`, not` *revision_number*`; re-execute syncreplica after site admin has upgraded database.`

In this case, you must upgrade the affected user database replica and run **syncreplica –import** again.

If you've automated your synchronization process, the automation script fails and additional packets that depend on the schema changes cannot be applied.

Manual Synchronization

This section describes how to synchronize replicas by entering explicit **syncreplica** commands.

Export Phase

1 Create an update packet at the sending host. Use the **syncreplica –export** command with the appropriate transport option.

If your sites are connected electronically, you can use store-and-forward to send the packet (**–fship**) or place it in a storage bay (**–ship**).

The following example uses the **–fship** option to send the packet immediately:

multiutil syncreplica -export -clan telecomm -site sanfran_hub -family PRODA -user jcole -password passwd -maxsize 50m -workdir c:\temp\packets -fship -sclass cq_default bangalore

Transport Phase

If you did not use the **–fship** option, send the packets:

- If you used **syncreplica –export –ship**, invoke **shipping_server** in either of the following ways:

  shipping_server –sclass cq_default –poll
  shipping_server *shipping-order-pathname*

- If you did not use **–fship** or **–ship**, send the packets using electronic mail, regular mail, or your preferred delivery method.

Import Phase

2 (If you used diskettes, CDs, tape, or electronic mail) Copy the packet files into a directory on the receiving replica's synchronization server.

3 Use the **lspacket** command to verify that the packet has arrived.

4 Apply the packet at the receiving replica. Use the **syncreplica –import** command to apply the changes in the packet to the replica.

   This example specifies the **–receive** option; **syncreplica** imports any packets it finds in the incoming shipping directories.

   multiutil syncreplica -import -family PRODA -user kumar -password secret -receive -sclass cq_default

   This example specifies a directory pathname as an argument. **syncreplica –import** looks in this directory for update packets and applies them to the replica on the host.

   multiutil syncreplica -import -family PRODA -user kumar -password secret \\ramohalli\shipping\incoming

Automated Synchronization

To automate the phases of synchronization, you can use **cron** (UNIX) or **at** (Windows) jobs or a third-party scheduling tool. You can specify a receipt handler in the shipping.conf file (UNIX) or MultiSite Control Panel (Windows) to automate the import phase.

On Windows, you can use the script *cquest-home-dir*\msimportauto.bat as a template.

MultiSite Management

Managing Replicas 8

This chapter describes how to manage existing replicas, including how to delete a replica. For information about creating a replica, see Chapter 6, *Creating Database Replicas*.

Displaying Properties of a Replica

The **lsreplica** command lists information about a replica.

For example, to display the names of all replicas in the **DOC** family:

multiutil lsreplica -clan telecomm -site boston_hub -family DOC -user susan -password passwd -short

```
BANGALORE
BOSTON_HUB
```

Moving or Renaming a Synchronization Server

You can change the synchronization server associated with a replica. For example, the machine you are using has a hardware failure, or you must rename the existing synchronization server. You must update the properties of the replica associated with that host so that the store-and-forward facility can determine how to route updates to the replicas.

To move the synchronization server:

1. Install the Rational Shipping Server on the new machine. (See the installation information for Rational ClearQuest.)
2. Use the **chreplica** command to associate the new synchronization server with the replica.

   multiutil chreplica -clan telecomm -site bangalore -family PRODA -user kumar -password secret -host server3 bangalore
3. If you automated the synchronization process on the old synchronization server, you must set up synchronization export and import scripts on the new server.

4 If you use routing hops, update the host name in the shipping.conf file (UNIX) or MultiSite Control Panel (Windows).

5 Export update packets to all sibling replicas.

Moving a Replica or Changing Vendor Database Software

To move a replica to a different host or change its vendor database software:

1 Follow the instructions in the *Administrator's Guide* for Rational ClearQuest.

2 Use the Maintenance Tool on the synchronization server to change the connection information for the database.

3 Export update packets to all sibling replicas.

Changing Allocation of ID Blocks to a Replica

MultiSite controls how many record ID numbers are allocated to each replica. This allocation is done by using ID blocks (groups of IDs).

By default, each replica is given an ID block of 4096 IDs when it is created. When a replica reaches a threshold of 1024 IDs remaining, it is allocated another ID block of 4096 IDs. This ensures that all IDs are unique. ID block allocation is handled internally by the working schema repository during synchronization.

Depending on the activity level of a replica family, you may want to increase the size of the ID blocks that are allocated to a replica, to ensure that synchronization flows smoothly. For example, with the default settings, if a synchronization packet contains enough new records that the receiving replica exceeds the number of IDs remaining in its current ID block, synchronization import fails.

To control the number of IDs allocated to a replica, you can do one of these things:

- Use the **–size** and **–threshold** options when you create a replica with the **mkreplica –export** command. For example, to create a new replica with an actual ID block size of 25000 and a threshold of 50%:

  multiutil mkreplica -export -clan telecomm -site boston_hub -family DEV -u susan -p passwd –size 250 –threshold 50 –out c:\cqms\boston_hub.xml goldengate:sanfran_hub

  ```
  Multiutil: Packet file 'c:\cqms\boston_hub.xml' generated
  ```

- Modify the size and threshold settings with the **chreplica** command. For example, to change all replicas in a site to have an actual ID block size of 50000 and a threshold of 30%:

  multiutil chreplica -clan testclan -site boston_hub -user susan -p passwd -host minuteman –size 500 -threshold 30 boston_hub

Changing Mastership of Replicas

For information about changing the working schema repository, see *Transferring Mastership of a Working Schema Repository* on page 83.

Deleting a Replica

This section describes how to remove a replica. You must complete all steps; if you do not, synchronization and mastership problems can occur in other replicas in the family.

When you remove a replica, the replicas in its family stop tracking epoch numbers for that replica. Removing a replica does not delete the database.

Removing a replica requires two synchronization cycles: one to transfer mastership of all of the replica's objects to another replica, and one to inform all other replicas that the removed replica is no longer participating in the update process. Because this information can be communicated only through the synchronization process, you cannot remove a replica at its own site, because doing so prevents the replica from creating update packets.

After a replica is removed from a family, it no longer participates in synchronization activities and MultiSite information is not tracked. The replica no longer updates its oplog, and you cannot transfer mastership of any object in that replica.

Removing a Functioning Replica from a Clan

To remove a replica that is still accessible and functioning, perform the following steps. The example syntax shows how to remove the replica in the **DOC** family at site **tokyo** and decommission the site named **tokyo** for a clan that also includes sites **sanfran_hub** and **boston_hub** (which contains the working schema repository). Each command must be entered on a single physical line.

1. Stop all work on the replica to be removed. Import all update packets.

2 Transfer mastership of all objects to another replica.

At site **tokyo**, run this command:

multiutil chmaster -clan telecomm -site tokyo -family DOC -user masako -password secret boston_hub -all -long

If the **chmaster** command reports errors, fix them and run the command again.

3 If you are decommissioning the entire site, you must also transfer mastership of users and groups in the site's schema repository.

At site **tokyo**, run this command:

multiutil chmaster -clan telecomm -site tokyo -family MASTR -user masako -password secret boston_hub -all -long

If the **chmaster** command reports errors, fix them and run the command again.

4 Send an update packet to the site receiving mastership.

At site **tokyo**, run this command:

multiutil syncreplica -export -clan telecomm -site tokyo -family DOC -user masako -password secret -workdir c:\work -fship boston_hub

At site **boston_hub**, run this command:

multiutil syncreplica -import -clan telecomm -site boston_hub -family DOC -user susan -password passwd -receive

5 Send an update packet from the site receiving mastership to all remaining sites.

At site **boston_hub**, run this command:

multiutil syncreplica -export -clan telecomm -site boston_hub -family DOC -user susan -password passwd -workdir c:\work -fship sanfran_hub

At site **sanfran_hub**, run this command:

multiutil syncreplica -import -clan telecomm -site sanfran_hub -family DOC -user jcole -password secret -receive

6 At the working schema repository site, run the **rmreplica** command. Be sure to include the final argument, which is the replica you want to remove.

At site **boston_hub**, run this command:

multiutil rmreplica -clan telecomm -site boston_hub -family DOC -user susan -password passwd tokyo

7 If you are decommissioning the site, you must run **rmreplica** on its schema repository.

At site **boston_hub**, run this command:

multiutil rmreplica -clan telecomm -site boston_hub -family MASTR -user susan -password passwd tokyo

8 Send an update packet from the working schema repository site to all other sites.

At site **boston_hub**, run this command:

multiutil syncreplica -export -clan telecomm -site boston_hub -family DOC -user susan -password passwd -workdir c:\work -fship sanfran_hub

At site **sanfran_hub**, run this command:

multiutil syncreplica -import -clan telecomm -site sanfran_hub -family DOC -user jcole -password secret -receive

9 Remove the vendor databases for the replicas that were removed.

Note: Rational does not support the use of a database after it has been removed from a clan using **rmreplica**. Attempting to use such a database may result in data corruption.

Removing an Inoperable Site from a Clan

If you have a site whose databases have been damaged beyond repair or deleted without a backup, and you want to remove the site from the clan, perform the following steps. The examples show how to remove the replica in the **DOC** family at site **tokyo** and decommission the site **tokyo** from a clan that also includes sites **sanfran_hub** and **boston_hub** (which contains the working schema repository). Each command must be entered on a single physical line.

1 Force the transfer of mastership of all objects from the unrecoverable replica to another replica.

At site **boston_hub**, run this command:

multiutil chmaster -clan telecomm -site boston_hub -family DOC -user admin -password secret boston_hub -all -force tokyo

2 If you are decommissioning the site, force the transfer of mastership for all users and groups.

At site **boston_hub**, run this command:

multiutil chmaster -clan telecomm -site boston_hub -family MASTR -user admin -password secret boston_hub -all -force tokyo

3 At the working schema repository site, run **rmreplica** to remove the unrecoverable replica. Be sure to include the final argument, the replica you want to remove.

At site **boston_hub**, run this command:

multiutil rmreplica -clan telecomm -site boston_hub -family DOC -user admin -password secret tokyo

4 If you are decommissioning the entire site, run the **rmreplica** command to remove its schema repository.

At site **boston_hub**, run this command:

multiutil rmreplica -clan telecomm -site boston_hub -family MASTR -user admin -password secret tokyo

5 Send an update packet from the site containing the working schema repository to all remaining sites.

At site **boston_hub**, run this command:

multiutil syncreplica -export -clan telecomm -site boston_hub -family DOC -user admin -password secret -workdir c:\work -fship sanfran_hub

At site **sanfran_hub**, run this command:

multiutil syncreplica -import -clan telecomm -site sanfran_hub -family DOC -user admin -password secret -receive

6 Remove the vendor databases for the replica and schema repository that were removed.

Using MultiSite After Removing the Last Replica in a Clan

If you use the **rmreplica –dbset** command on the last replica in your clan (leaving only the database at the site containing the working schema repository), the database is no longer part of a MultiSite environment. It is now an unreplicated database and cannot be used to create new replicas while it is in this state. For more information, see the **rmreplica** reference page.

To change the state of the database so that you can replicate it again, you do not need to run the **activate** command. You must use the Rational ClearQuest Maintenance Tool to change the database set name back to the form expected by **multiutil**; for example, **CQMS.***clan-name*.*site-name*.

Managing Mastership

9

This chapter describes how to manage the mastership of objects in a replica. Before reading this chapter, you should read the information in *Enabling Independent Development: Mastership* on page 5.

Mastership Commands for User Database Objects

The following **multiutil** commands are used to manage mastership of user database objects:

- **chmaster**
- **describe**

For more information about these commands, see their reference pages in this manual.

Displaying Mastership Information for Records

In query results and on record forms, a padlock icon indicates the records in the result set that are mastered at other sites.

Changing Mastership of Database Objects

You can transfer mastership of an object with a **chmaster** command or with the GUI. Mastership changes are appropriate in the following situations:

- You want to allow a user at another replica to modify a record or public query that is mastered by your replica.
- You want to make changes to user information that is mastered by another replica.
- You are decommissioning a replica, and you must transfer mastership of all objects mastered by that replica to one of the remaining replicas.

Mastership changes are communicated among replicas by the standard synchronization mechanism. The general procedure for changing mastership is as follows:

1. At the master replica, change mastership of one or more objects to another replica.
2. At the old master replica, export an update packet from the old master replica to the new master replica.
3. At the new master replica, import the update packet.

Until the update packet that contains the mastership change is imported at the new master replica, the mastership change is "in the packet," and the replicas in the database family have different information about which replica masters the object.

For example, the administrator at the **sanfran_hub** replica transfers mastership of the user group **QA_ENGINEERING** to the **bangalore** replica and exports an update packet. At this point:

- The **sanfran_hub** replica considers the user group to be mastered by **bangalore**.
- The **bangalore** replica considers the user group to be mastered by **sanfran_hub**.
- No one at any replica can modify the user group.

When you complete the mastership transfer by importing the update packet at **bangalore**, users at **bangalore** can modify the user group **QA_ENGINEERING**.

Notes on mastership changes:

- If your family includes any read-only or one-way replicas (replicas that import update packets but do not export them), be careful about transferring mastership to these replicas. After you give mastership of an object to a read-only or one-way replica, you cannot change the object's mastership unless you change the family's synchronization pattern.
- You cannot undo a mastership change made at your site by making the opposite change at your site. See *Fixing an Accidental Mastership Change* on page 84.

Transferring Mastership of a Record with the GUI

ClearQuest MultiSite includes a system field called **ratl_mastership**. The value of this field is the replica that masters the respective record. To allow users to change the mastership of a record, you must add this field to the form of the record type. You can add the **ratl_mastership** field to the form of a record type at any time.

Note: Only users at the master replica of a record can change its mastership.

For each record type for which you want to allow users to change record mastership, use the ClearQuest Designer at the working schema repository to modify the schema of the replica family. To add the **ratl_mastership** field to a record form:

1 In the Workspace, expand **Record Types**, and then expand the desired record type.

2 Double-click the form.

3 Click the tab to which you want to add the field, or click **Edit > Add Tab** to add a new tab to contain the field.

4 Using the **Field List**, drag the **ratl_mastership** field to the tab.

5 Check in the modified schema.

6 Upgrade the appropriate user databases to use the new schema.

7 Generate and send an update packet. Update packets automatically contain schema updates. Administrators at other sites must upgrade their user database replicas to use the new schema revision. For more information, see *Applying Packets That Include Schema Updates* on page 65.

For more information about modifying a schema, see the *Administrator's Guide* for Rational ClearQuest.

Transferring Mastership of a Record with chmaster

To transfer mastership of a record:

1 At the master replica (in this example, **boston_hub**), enter a **chmaster** command:

multiutil chmaster -clan telecomm -site boston_hub -family DOC -user susan -password passwd bangalore entity:DOC00013

```
multiutil: The mastership of entity:DOC00013 has been changed to
site 'bangalore'
```

2 At the old master replica, export an update packet to the new master replica:

multiutil syncreplica -export -clan telecomm -site boston_hub -family DOC -user susan -password passwd -workdir d:\shipping\temp -fship -sclass cq_default bangalore

3 At the new master replica, import the packet:

multiutil syncreplica -import -clan telecomm -site bangalore -family DOC -user kumar -password passwd -receive -sclass cq_default

4 At the new master replica, verify that mastership has been received:

multiutil describe -clan telecomm -site bangalore -family DOC -user kumar -password passwd entity:DOC00013

```
multiutil: The mastership of entity:DOC00013 is 'bangalore'
```

Transferring Mastership of a Workspace Item with the GUI

To change mastership of a Workspace item by using the ClearQuest client:

Note: You must have Public Folder Administrator privileges to modify Workspace items in the **Public Queries** folder.

1. In the Workspace, right-click the item you want to modify and click **Mastership**.
2. In the Change Mastership dialog box, select the new master replica from the **New Mastering Site** list.
3. Click **OK**.
4. Export an update packet from the old master replica to the new master replica, and import the packet at the new master replica.

Transferring Mastership of a Workspace Item with chmaster

Follow these guidelines to specify Workspace items:

- Enclose the Workspace name in quotes.
- Use the case that is shown for the item in the ClearQuest Workspace.
- Include the full Workspace path name (folders and subfolders) of the Workspace item.

For example:

"Workspace:Personal Queries(susan)\My Projects\My Query"

To change mastership of a Workspace item:

1. At the master replica (in this example, **boston_hub**), enter a **chmaster** command:

   multiutil chmaster -clan telecomm -site boston_hub -family DOC -user susan -password passwd bangalore "Workspace:Public Queries\Triage\project report"

2. At the old master replica, export an update packet to the new master replica:

   multiutil syncreplica -export -clan telecomm -site boston_hub -family DOC -user susan -password passwd -workdir d:\shipping\temp -fship -sclass cq_default bangalore

3. At the new master replica, import the packet:

   multiutil syncreplica -import -clan telecomm -site bangalore -family DOC -user kumar -password passwd -receive -sclass cq_default

4 At the new master replica, verify that mastership has been received:

multiutil describe -clan telecomm -site bangalore -family DOC -user kumar -password passwd "Workspace:Public Queries\Triage\project report"

```
multiutil: The mastership of Workspace:Public
Queries\Triage\project report is 'bangalore'
```

Transferring Mastership of a User or Group

If you receive a synchronization packet that contains user administration changes such as new users or groups, you must upgrade the user database to use the changes. For more information about administering users in a MultiSite environment, see the *Administrator's Guide* for Rational ClearQuest.

To change the mastership of a user or group, do the following things:

1 At the old master replica:

   a Use the **chmaster** command or the ClearQuest Designer to change the mastership of the users or groups to the new replica.

   b Export an update packet to the new master replica.

2 Notify the administrator of the new master replica that the incoming packet requires a user database upgrade.

3 At the new master replica:

   a Import the update packet.

   b Upgrade the associated user databases with the changes.

   c Export and send a synchronization packet to ensure that other replicas in the family are updated with the change.

Changing Mastership of Users or Groups with the GUI

To change mastership of a user:

1 In ClearQuest Designer, click **Tools > User Administration**.

2 In the User Administration dialog box, double-click the user you want to modify.

3 In the User Properties dialog box, open the **Mastership** list and select the new master replica.

4 Click **OK**.

5 Click **OK**.

6 Export and send an update packet from the old master replica to the new master replica, and import the packet at the new master replica.

To change mastership of a user group:

1 In ClearQuest Designer, click **Tools > User Administration**.

2 In the User Administration dialog box, select the user group you want to modify.

3 Click **Group Action > Edit Group**.

4 In the Group Property dialog box, open the **Mastership** list and select the new master replica.

5 Click **OK**.

6 Click **OK**.

7 Export and send an update packet from the old master replica to the new master replica, and import the packet at the new master replica.

Changing Mastership of Users or Groups with chmaster

Use the following conventions to specify users and groups:

- If the user or group name includes spaces, enclose it in quotes. For example:

  user:"John Smith"

- Use the same form of the user or group name assigned when it was created.

The following example changes the mastership of users John Smith and Jane Doe from the **boston_hub** replica to the **bangalore** replica.

1 At the master replica (in this example, **boston_hub**), enter a **chmaster** command:

   multiutil chmaster -clan telecomm -site boston_hub -family DOC -user susan -password passwd bangalore user:"John Smith" user:"Jane Doe"

```
multiutil: The mastership of records "Jane Doe" and "John Smith"of
type "user" has been changed to site ‘bangalore’
multiutil:The mastership of some users or groups have been
transferred from this site. The local user admin must update user
databases at the new mastering site 'BANGALORE' before these changes
will be visible to any user database.
```

2 At the old master replica, export an update packet to the new master replica:

   multiutil syncreplica -export -clan telecomm -site boston_hub -family DOC -user susan -password passwd -workdir d:\shipping\temp -fship -sclass cq_default bangalore

3 At the new master replica, import the packet:

multiutil syncreplica -import -clan telecomm -site bangalore -family DOC -user kumar -password passwd -receive -sclass cq_default

4 At the new master replica, upgrade the user database with the new user information. For more information about upgrading user databases, see the *Administrator's Guide* for Rational ClearQuest.

5 At the new master replica, verify that mastership has been received:

multiutil describe -clan telecomm -site bangalore -family DOC -user kumar -password passwd user:"John Smith" user:"Jane Doe"

```
multiutil: The mastership of user:John Smith is 'bangalore'
multiutil: The mastership of user:Jane Doe is 'bangalore'
```

Transferring Mastership of a Working Schema Repository

The administrator at the working schema repository is responsible for modifying schemas and adding new families to a clan. For more information, see *Kinds of Schema Repositories* on page 3. If you want to transfer responsibility for these tasks to another site, you must change mastership of the working schema repository.

To transfer mastership of a working schema repository:

1 At the working schema repository replica, enter a **chmaster** command:

multiutil chmaster -clan telecomm -site boston_hub -family MASTR -user susan –password passwd sanfran_hub -workingmaster

2 At the old working schema repository, export an update packet to the new working schema repository:

multiutil syncreplica -export -family MASTR -user susan -password passwd -workdir c:\temp\shipping -fship -sc cq_default sanfran_hub

3 At the new working schema repository, import the packet:

multiutil syncreplica -import -family MASTR -user jcole -password passwd -receive -sc cq_default

4 At the new working schema repository, verify that mastership has been received:

multiutil describe -clan telecomm -site sanfran_hub -family MASTR -workingmaster

Fixing an Accidental Mastership Change

If a mastership change is made in your replica by mistake, follow these steps:

1. At your replica, send an update packet to the new master replica.
2. At the new master replica:
   a. Import the packet.
   b. Change mastership back to your replica.
   c. Export and send an update packet to your replica.
3. At your replica, import the packet.

Troubleshooting

Troubleshooting MultiSite Operations 10

This chapter describes common situations in which running a Rational ClearQuest MultiSite command produces an unexpected result, sometimes accompanied by a warning or error message. The situations fall into these categories:

- **Expected conditions** occur because certain inconsistent changes at different replicas cannot be avoided. In many cases, a MultiSite operation resolves the inconsistency, so you need not take any action.
- **Recoverable errors** are user errors, hardware problems, and other problems that you resolve by performing a recovery procedure.
- **Serious errors** are problems that may require assistance from Rational Customer Support.

The organization of the descriptions follows the general MultiSite data flow from replica creation through the phases of replica synchronization—export, transport, and import.

Replica Export Problems

If the **mkreplica –export** command finds that a replica with the specified name exists in the family (`Replica` *replica-name* `already exists`), select another name for the new replica, and reenter the **mkreplica –export** command.

If **mkreplica –export –fship** fails while it is transporting the packet, it does not remove the new replica's replica object at the creating replica. To complete the replica creation, use **shipping_server** to transfer the replica-creation packet.

Recovering from a mkreplica –export Failure

If **mkreplica –export** fails, your database may be left in a locked state. To resolve this problem, use the procedures in this section or contact Rational Customer Support.

Unlock the Schema Repository and User Database

1. Unlock the schema repository with the **installutil unlockschemarepo** command. This command has the following syntax:

   installutil unlockschemarepo *db-vendor server database dbo-login dbo-password connection-options*

   Connection options:

Oracle database	**HOST=***host***;SID=***sid*
All others	""

   For example, to unlock the **SQL_SERVER** schema repository database **test_master_sitea** on server **QE_TEST1**:

   installutil unlockschemarepo SQL_SERVER QE_TEST1 test_master_sitea multisite multisite ""

   where **multisite** is the *dbo-login* and *dbo-password* for the **test_master_sitea** database.

2. Unlock the user database with the **installutil unlockuserdb** command. This command has the following syntax:

   installutil unlockuserdb *db-vendor server database dbo-login dbo-password connection-options*

   Connection options:

Oracle database	**HOST=***host***;SID=***sid*
All others	""

   For example, to unlock the **SQL_SERVER** user database **test_user_sitea** on server **QE_TEST1**:

   installutil unlockuserdb SQL_SERVER QE_TEST1 test_user_sitea multisite multisite ""

   where **multisite** is the *dbo-login* and *dbo-password* for **test_user_sitea**.

Subsequent multiutil Commands Fail

If subsequent attempts with **mkreplica –export** result in messages that indicate that the replica already exists or that another **multiutil** operation is in progress, contact Rational Customer Support.

Replica Import Problems

If the **mkreplica –import** command fails for any reason other than a code page mismatch (for example, a network outage or not enough disk space), you must rerun the command. If the **mkreplica –import** command fails because the data code page value of the exporting replica does not match the code page of the existing schema repository at the importing site, you must first reset one of the code page values and then rerun **mkreplica –import**.

mkreplica can fail during import of the schema repository or during import of the user database.

If **mkreplica –import** fails during the import of the schema repository, follow these steps:

1. Delete the vendor databases for the schema repository and user database replica.
2. Delete the newly created database set name, which is in the format **CQMS.***clan-name***.***site-name*. Use the following command:

   installutil dropdbset CQMS.*clan-name***.***site-name*
3. Re-create the vendor databases.
4. Enter the **mkreplica –import** command.

If **mkreplica –import** imports the schema repository, but fails during import of the user database, follow these steps:

1. Delete the vendor databases intended for the user database replicas.
2. Re-create the vendor databases.
3. Run the **mkreplica –import** command again, omitting the repository database options. For example:

   multiutil mkreplica -import -clan telecomm -site tokyo -user masako -p secret -database cq_userdb -vendor SQL_SERVER -dbologin juseradmin secret -rwlogin juseradmin secret

Synchronization Export Problems

This section describes problems that can occur during the export phase of synchronization.

Cannot Find Oplog Entry

syncreplica –export can fail with the following warning message:

```
Multiutil: Error: 'family' database has exported oplog entries
originating from replica 'site' through oplog-ID oplog-ID. The next
oplog-ID to be exported is oplog-ID; it should be oplog-ID. A gap in
oplog entries may indicate missing oplog entries.
```

(For more information about oplog entries, see *The Operation Log* on page 8.)

This error occurs when the sending replica's epoch number matrix does not match its set of oplog entries. For example:

- Before sending an update from **sydney** to **buenosaires**, **syncreplica** checks the epoch number matrix for **sydney**. It determines that the last **sydney** operation sent to **buenosaires** was 3620.
- **O**plog scrubbing in the **sydney** database has removed some of the operations that follow 3620. The earliest **sydney** operation remaining in the oplog is 5755.

This discrepancy may be an expected condition. For example, when you change the synchronization pattern for a family, replicas that have not communicated with each other in the past start exchanging update packets. Synchronizing two replicas (**syncreplica –export** followed by **syncreplica –import**) updates epoch number matrix rows for the sending and receiving replicas, but it does not revise the row for any other replica. If two replicas rarely (or never) send updates to each other directly, the relevant rows in their epoch number matrices are out of date (possibly consisting of all zeros). This is not a problem, as long as the replicas receive operations indirectly, for example, through a hub replica.

In this case, you must inform **sydney** about the true state of **buenosaires**, information that **sydney** has not received through the standard synchronization mechanism. This information enables **sydney** to determine which oplog entries to send to **buenosaires**.

This situation may also occur if you remove oplog entries with the **scruboplog** command before they are sent to other replicas. You must make sure that you have synchronized the replicas in a family before you scrub oplogs at any of the replicas. See *Scrubbing Parameters for Replicas* on page 29.

Packets Accumulate in Outgoing Storage Bay

Problems with packet delivery are recoverable errors. In many cases, the MultiSite automatic-retry capability recovers from errors.

A replica-creation or update packet submitted to the store-and-forward facility for transport to one or more other hosts is accompanied by a shipping order file. (A logical packet can include multiple physical packets, each with its own shipping order.) The shipping order typically has an expiration time, determined by one of the following:

- A date-time specified with the **–pexpire** option in the **syncreplica** or **mkreplica** command that generated the packet (or the **mkorder** command that submits an arbitrary file to the store-and-forward facility)
- On UNIX, the **EXPIRATION** value in the store-and-forward configuration file (shipping.conf) on the sending host
- On Windows, the **Packet Expiration** value specified in the MultiSite Control Panel on the sending host

Any number of delivery attempts may take place before the shipping order expires.

Replica Cannot Update Itself

You can receive the following message during export if you specify the sending replica as a destination:

```
A replica cannot update itself
```

If the sending replica is the only replica you specified, the **syncreplica –export** command fails. If you specified other replicas, this message is printed as a warning, and the **syncreplica –export** command continues its processing.

Transport Problems

This section describes problems that can occur during the transport phase of synchronization.

Error Messages

The messages in Table 12 are generated by the **mkorder**, **mkreplica**, **shipping_server**, and **syncreplica** commands.

Table 12 Shipping Error Messages (Part 1 of 2)

Error message	**Meaning**
`cannot find a storage bay for class` *class-name*`: no such bay specified`	No storage bay is assigned to storage class *class-name* in the shipping.conf file or the MultiSite Control Panel.
`cannot find a storage bay for class` *class-name*`: all applicable bays are either inaccessible or do not contain` *byte-count* `free bytes`	Lack of permission or lack of free disk space prevents use of storage bays for class *class-name*.

Table 12 Shipping Error Messages (Part 2 of 2)

`cannot process potential order file` *shipping-order-pname*`: user` *username* `(UID` *uid*`) is not the owner`	(UNIX) The shipping server is not running as **root**, and *username* does not own the shipping order file.
`cyclic delivery route detected to host` *hostname* `(via` *next-hop-hostname*`) for order` *shipping-order-pname*	The shipping order lists *next-hop-hostname* as a previous hop in the packet's delivery route. If the packet is sent to *next-hop-hostname* (which is specified in a **ROUTE** entry in the shipping.conf file or in the **Routing Information** section in the MultiSite Control Panel), it will eventually come back to the current host. Check the routing information on the hosts in the delivery path and fix any circular routes.
`file` *file-pname* `does not contain a valid shipping order`	The shipping server attempted to process a file that is not a shipping order.
`for security reasons, shipping order` *shipping-order-pname* `cannot be processed: data file` *file-pname* `must be in the same directory as the shipping order`	A shipping order and its associated packet file must be in the same directory.
`giving up trying to return order` *shipping-order-pname* `to host` *hostname* `(original data file was` *file-pname*`)`	The shipping server cannot return a packet or other file to its original sending host (for example, because its shipping order expired) and has deleted the shipping order and data file.
`ignoring shipping bay` *storage-bay-pname*`:` *reason*	The storage bay directory specified in the shipping.conf file or MultiSite Control Panel doesn't exist or is inaccessible.
`shipping order` *shipping-order-pname* `not found (perhaps previously sent?)`	During receipt handler processing, the shipping server cannot find the shipping order of a packet that is to be forwarded to another host. A **shipping_server –poll** invocation may have sent the packet already. (If the packet is to be applied to replicas on the host, the imports occur before the packet is forwarded. This leaves a window of opportunity for a scheduled polling operation to send the packet.)

Invalid Destination

The local host's hosts file, **hosts** NIS map, or Domain Name Service must list one of the following hosts:

- Destination host
- Next-hop host corresponding to the destination host (on UNIX, defined in a **ROUTE** entry in the host's shipping.conf file; on Windows, defined in the **Routing Information** section in the host's MultiSite Control Panel.)

Note: If hosts in your network are known only by their IP addresses, you can use the IP addresses instead of host names.

In the absence of such entries, the shipping server fails, because it cannot determine where to deliver the packet. In this case, it writes error messages to its log file (UNIX) or the Windows Event Viewer.

If the destination host name was misspelled, use the **mkorder** command to create a new shipping order with the correct host name. If a host name is misspelled in a **mkreplica –export** command, the incorrect host name is recorded. Verify the error with **lsreplica –long**, and correct the spelling with **chreplica**.

In other cases, you may have to revise the host's database of remote hosts. The sending host must be able to communicate with the receiving hosts through TCP/IP channels. Use the **rcp** command on the sending host to copy a file to the receiving host. If it fails, you have a setup or networking problem with your host. If the command succeeds, contact Rational Customer Support.

Delivery Fails

Each time the shipping server cannot deliver a packet to a valid destination host, it logs error messages:

- (On UNIX) In file /var/adm/rational/clearcase/log/shipping_server_log and writes a message to the terminal device, if there is one.
- (On Windows) In the Windows Event Viewer.

If the problem is temporary (remote host is down, network connections are down, and so on), a subsequent invocation of **shipping_server –poll** will transmit the packet successfully. If the problem is not temporary, the shipping order may expire eventually.

Shipping Server Fails to Start or Connection Is Refused

If the shipping server on the receiving host does not start or the connection is refused, check the **albd_server** log on the receiving host for an explanation of the failure.

A syntax error in the shipping.conf file on UNIX can cause the connection to be refused. For example, if there is an incorrect e-mail address in the file, the **albd_server** log displays an error like this:

```
Error: shipping_server(9951): Error: syntax error in configuration
file (line 60)
```

Shipping Order Expires

If the shipping server finds that a shipping order has expired, it attempts to return the packet to the originating host. Also, it sends a mail message to one or more administrators on the original sending host, and sends another mail message when the packet is returned to the original sending host. On Windows, if e-mail notification is not enabled, the shipping server writes a message to the Windows Event Viewer.

Use the **lspacket** command to check the return bays on your host. The packet files may have been returned by store-and-forward. If so, try again to deliver the packet:

- Fix the store-and-forward packet-delivery mechanism (for example, by fixing the network connection). Then, use **mkorder** to create a new shipping order for each physical packet file in the return bay.
- If you cannot fix the store-and-forward mechanism, deliver the packet by some other means. For example, copy the packet file to a CD, and mail the CD to the remote sites.

If the packet files are not in your host's return bays, they may be in transit. Search for the files immediately, because a packet that cannot be returned to its originating host within 14 days is deleted.

Synchronization Import Problems

This section describes problems that can occur during the import phase of synchronization.

Packets Accumulate in Incoming Storage Bay

A recoverable error occurs when an update packet is lost and is not applied to your replica. These are the symptoms:

- One or more replicas at your site are not being updated on their regular schedules.
- An **lspacket** command shows unprocessed packets accumulating in the storage bay. These packets depend on the missing packet and cannot be processed.

To verify that a packet is missing and determine which operations are needed:

1 Enter a **syncreplica –import –receive** command, which processes all incoming packets in the storage bay in the correct order. If **syncreplica** fails to process any of them, a packet is missing.

2 Enter a **syncreplica –import** command that specifies the oldest packet in the storage bay:

   multiutil syncreplica -import -clan telecomm -site sanfran_hub -family DEV -user jcole -p passwd *packet-pathname*

```
Multiutil: Packet packet-pathname not processed...
Multiutil: The UPDATE_PACKET packet sent from BOSTON_HUB at
2002-03-25 17:42:41  for 'DEV' cannot be replayed: This replica has
not replayed epoch 6 from replica BOSTON_HUB, it has only replayed
through 2.
Multiutil: The UPDATE_PACKET packet sent from BOSTON_HUB at
2002-03-25 17:42:41  for 'MASTR' cannot be replayed: This replica
has not replayed epoch 8 from replica BOSTON_HUB, it has only
replayed through 6.
```

In this example, one or more update packets are missing, containing operations 3-6 originally occurring in the user database in the **DEV** family at the **boston_hub** site and operations 7-8 in the schema repository at the **boston_hub** site. In general, a packet can contain operations from several replicas; the **syncreplica –import** command fails if operations are missing from any replica.

Locate the missing packets. They may be on media that you forgot to process or in packet files that were not processed because your store-and-forward configuration (the shipping.conf file on UNIX; the MultiSite Control Panel on Windows) specifies the wrong storage bay. If you locate the missing packets, do one of the following things:

- Process the missing packets by naming them in a **syncreplica –import** command. (Multiple packet files are imported in the correct order, regardless of the order of the command-line arguments.)
- Process all the update packets that have accumulated in the storage bay by entering a single **syncreplica –import –receive** command.

If you cannot locate the missing packets, see *Recovering from Lost Packets* on page 96.

Packet Is Not Applicable to Any Local Replicas

Import can fail with the following message:

```
multiutil: Error: Sync. packet pathname is not applicable to any local
replicas.
```

This error can occur when a synchronization server has been moved and the host-name property has not been updated with the **chreplica** command.

To verify that the host-name property is wrong, use the **lsreplica** command. For example, if the above error occurred at the **bangalore** replica:

multiutil lsreplica -site bangalore -user kumar -p secret -long bangalore
```
Name:bangalore; Clan:TELECOMM; Family:PRODA; Host:shiphost1; Status:
NORMAL, NOT CONNECTED; Description:Production database
```

If the host name is incorrect, use the **chreplica** command to change it. Then, send an update packet to the other replicas in the family.

Read from Input Stream Fails

If a **syncreplica –import** command fails with a message like this one, the packet is corrupted:

```
Error: Read from input stream failed: No such file or directory
```

Delete the packet and ask the administrator at the sending replica to re-create the packet and send it again (see *Recovering from Lost Packets* on page 96). Then import it.

Miscellaneous Problems

Processing of an incoming replica-creation or update packet may fail because of these conditions:

- Disk partition is full.
- Receiving replica is locked.
- Licensing failure.
- Multiple imports occur simultaneously.

Make sure that multiple **syncreplica –import** commands do not run in the same replica simultaneously. In such cases, fix the problem and reenter the **syncreplica –import** command.

Recovering from Lost Packets

There are several circumstances in which an update packet is generated but is never applied at one or more of its destinations:

- The packet is stored on media that is destroyed or is not readable at the destination host.
- A packet file is lost when a hard disk fails.
- The packet is intact, but cannot be applied because another packet has been lost. (See *Packets Accumulate in Incoming Storage Bay* on page 94.)

The **syncreplica –export** command assumes successful delivery of the update packet it generates. For example, when replica **boston_hub** sends an update to replica **sanfran_hub**, the **syncreplica** command assumes that the operations originating at **boston_hub** are imported to the **sanfran_hub** replica. For simplicity, this example does not reflect the fact that the update packet can also contain operations that originated at other replicas in the family.

If the packet is lost, **boston_hub** must reset its estimate of the state of replica **sanfran_hub**. After this correction is made, the next update packet sent from **boston_hub** to **sanfran_hub** contains the operations **sanfran_hub** needs.

To reset the epoch row, use the method described here.

1. At the receiving replica, **sanfran_hub**, display the replica's epoch number matrix:

   multiutil lsepoch -clan telecomm -site sanfran_hub -family PRODA -user jcole -p secret sanfran_hub

   ```
   Multiutil: Estimates of the epochs from each site replayed at site
   'sanfran_hub' (@goldengate):
   BANGALORE: 950
   BOSTON_HUB: 1300
   SANFRAN_HUB: 2000
   ```

2. Use this output in a **chepoch** command at the sending replica, **boston_hub**. This sets the **boston_hub** epoch number estimates for **sanfran_hub** to the actual values of the **sanfran_hub** epoch number matrix:

   multiutil chepoch -clan telecomm -site boston_hub -family PRODA -user bostonadmin -password secret sanfran_hub bangalore=950 boston_hub=1300 sanfran_hub=2000

   ```
   Multiutil: Change the estimate for the epochs of site 'bangalore'
   replayed at site 'sanfran_hub' to 950 [yes|NO|quit]yes

   Multiutil: Change the estimate for the epochs of site 'boston_hub'
   replayed at site 'sanfran_hub' to 1300 [yes|NO|quit]yes

   Multiutil: Change the estimate for the epochs of site 'sanfran_hub'
   replayed at site 'sanfran_hub' to 2000 [yes|NO|quit]yes

   Multiutil: 3 epoch estimate(s) for site 'sanfran_hub' successfully
   changed; 0 failures.

   Multiutil: Estimates of the epochs from each site replayed at site
   'sanfran_hub' (@goldengate):
   BANGALORE: 950
   BOSTON_HUB: 1300
   SANFRAN_HUB: 2000
   ```

Removing Circular Duplicate Links

In a replicated ClearQuest environment it is possible to create a circular duplicate link where a defect and its duplicate are both set to the Duplicate state. Circular duplicate links can be created by doing the following:

1. At **boston_hub**, make **Defect1** a duplicate of **Defect2**.
2. At **sanfran_hub**, make **Defect2** a duplicate of **Defect1**.
3. Synchronize **boston_hub** and **sanfran_hub**. Both defects are now in the Duplicate state.

To remove a circular duplicate link, unduplicate one of the defects at the site that masters it.

Resolving Naming Conflicts

If you don't impose a site-specific naming convention for Workspace items (queries, reports, charts, and so on), users and groups, and other stateless records, it is possible to have different objects with the same name.

For example, duplicate names can occur when user administrators at two sites within a clan add the same user name within a synchronization cycle. In this case, after the replicas are synchronized, two users have the same name.

Internally, ClearQuest ensures that records and Workspace names are unique:

- For record types that use states, ClearQuest uses database ID numbers to ensure uniqueness.
- For stateless record types (including users and groups), ClearQuest uses unique keys and stores the name of the originating site, or keysite.
- For Workspace items, ClearQuest stores the name of the originating site, or keysite, and the name of the Workspace item.

Workspace Naming Conflicts and ClearQuest Web

If two Workspace items (queries, reports, and so on) have the same name, both items work as expected in the Windows and UNIX clients, according to mastership restrictions and database privileges. However, in ClearQuest Web, only one of the items works. To avoid confusion, you must rename one or both of the items.

Renaming Workspace Items

In order for you to modify a Workspace item, your current replica must master it. To determine which replica masters a Workspace item, open the Workspace, right-click the item, and click **Mastership**.

To rename a Workspace item:

1 Right-click the Workspace item and click **Rename**.

2 Type a new name in the highlighted area and click **Enter**.

Working with Ambiguous Workspace Items

If you need to use **multiutil** commands to work with a Workspace item with a naming conflict, you must refer to its keysite name, which is the name of the site where the Workspace item originated. For example:

"Workspace:\Public Queries\Project Report<keysite-name>"

The following example uses the keysite name in the object selector:

multiutil describe -clan telecomm -site tokyo -family PRODA -user tokyoadmin -password secret "workspace:Public Folder\Project Report<boston_hub>"

```
Multiutil: Mastership of 'workspace:Public Queries\Project
report<boston_hub>' is 'boston_hub'.
```

Fixing Naming Conflicts for Stateless Record Types

To fix a naming conflict for a stateless record, you must rename one of the records.

Renaming Records

To rename a stateless record that has a naming conflict:

1 Find the record in question. See *Finding Stateless Records with Naming Conflicts* on page 100.

2 Change the name of the record. You must have mastership of the record to modify it.

   To rename a stateless record, you must modify a field that is used as the unique key for that record. To do this, use the action in your schema that allows you to modify records without changing their state.

3 Synchronize the family.

Ensuring That a Record Is Unique

To ensure that a record is unique, stateless record types use the **ratl_keysite** field. The **ratl_keysite** field is an internal system field that stores the name of the site where an object was created.

For example, a new customer named NetInc is created at two replicas during the same time period between synchronizations. When each replica synchronizes, there appear to be two customer records with identical names. To ensure uniqueness, ClearQuest references the **ratl_keysite** field.

If you need to use **describe** or **chmaster** commands to work with an ambiguous record, you must refer to its keysite name (originating site name). For example:

`customer:NetInc<`*keysite-name*`>.`

The following example uses the keysite name in the object selector:

multiutil describe -clan telecomm -site tokyo -family PRODA -user masako -password secret customer:NetInc<boston_hub>

```
Multiutil: Mastership of 'customer:NetInc<boston_hub>' is 'boston_hub'.
```

Finding Stateless Records with Naming Conflicts

You can use the **ratl_keysite** field in queries designed to find stateless records of the same name. Follow these guidelines when querying for stateless records with naming conflicts.

- Use the **ratl_keysite** field as both a Display field and Filter when creating a query on the respective stateless record type.
- If the query finds any duplicate record names, rename them according to a site-specific naming convention that has been agreed upon. Remember that your current replica must master a record in order for you to modify it.

To assist you in viewing and modifying records, you can add the **ratl_keysite** field to the form of any stateless record type where you expect naming conflicts to arise. For more information, see the *Administrator's Guide* for Rational ClearQuest.

Identifying User and User Group Naming Conflicts

To log on with an ambiguous user name, use the keysite name as part of the user logon name (for example, *username*<*keysite-name*>, where *keysite-name* is the site where the user was created). If you log on with an ambiguous user name without the keysite name, you get an invalid logon error. Clicking the detail displays the following error:

```
User name 'xxx' is ambiguous; rename or qualify with '<'SITE'>' to
proceed.
```

Renaming Users

In ClearQuest Designer, when you try to modify user information for a user who has a conflicting name, you get the following error message because of the **< >** characters in the name:

```
ERROR! The string value ("DupUser<SITE1>") is invalid: Names cannot
contain one of these characters:! "#$%&'()*+,./:;<=>?@[\]^`{|}~
```

You must rename the user. You cannot modify any user information, except the Name field, until the user has a unique name that does not include the **< >** characters.

You cannot rename or delete a user group.

To rename a user:

1. Click **Tools > User Administration**.
2. In the User Administration dialog box, double-click the user you want to modify.
3. In the User Properties dialog box, modify the name of the user.
4. Click **OK**.
5. Upgrade the associated user database by clicking **DB Action > Upgrade**.
6. In the Upgrade dialog box, select the user databases you want to upgrade.
7. Click **OK**.
8. Click **OK**.
9. The administrator at the new master replica must upgrade the database after receiving the synchronization packet that contains the changes. For more information, see the *Administrator's Guide* for Rational ClearQuest.

Using multiutil with Ambiguous Users and User Groups

If you need to use **describe** or **chmaster** commands to work with a user or group whose name is the same as another user or group, you must refer to their respective keysite name (originating site name).

Use the **describe** command to find out where the user is mastered. In this example, the keysite is the **boston_hub** replica:

multiutil describe -clan telecomm -site tokyo -family PRODA -user masako -password secret user:jsmith<boston_hub>

```
Multiutil: Mastership of 'user:jsmith<boston_hub>' is 'boston_hub'.
```

Updating Database Subscriptions After Replicating a Database

When you add a new user database at the working schema repository site, we recommend that you replicate the new user database before subscribing users to it.

Note: An exception to this rule is when a user is subscribed to all databases. Users who are subscribed to all databases can access a new database without problems.

Users who are subscribed to the new database before it is replicated cannot log in to the new database replica until their database subscription is updated at the working schema repository site.

1. Using the ClearQuest Designer, log in to the working schema repository. This must be done at the working schema repository site, and you must have at least User Administrator privileges.
2. In ClearQuest Designer, click **Tools > User Administration**.
3. Select the user who cannot log on to the replica.
4. Click **DB Subscriptions**.
5. In the Database Subscriptions dialog box, click **OK**.
6. Click **Yes** in the change confirmation box.
7. Repeat Step 3 through Step 6 for each user who has subscription problems.
8. Click **Yes** in the change confirmation box.
9. In the User Administrator dialog box, click **Upgrade the user DB**.
10. In the Select Site dialog box, select the user database that you want to upgrade (select the replica you just created).
11. Synchronize the change to all sites that have the subscription problem.
12. If the user is subscribed only to databases that are replicated, viewing the database subscriptions for the user now works at all sites that have a replica of the new user database. Sites that do not have this replica cannot view the database subscriptions for the user.

Restoring Database Replicas

Occasionally, a replica is lost. This loss is usually due to a hardware failure (for example, disk crash), a software failure (for example, OS-level file system corruption), or a human error. If an unreplicated database is lost, you can restore a recent copy from backup and resume development work. The changes made between the time of the backup and the time of the failure are not recoverable.

Similarly, if you lose a replica, you can restore a recent copy from backup. But matters are more complicated:

- Some of the work done between the time of the backup and the time of the failure may be recoverable. If some of the operations were sent to other replicas in update packets, these operations must be retrieved and imported.
- The restored copy of the replica is out of date. You must make this replica consistent with the other replicas in the family before development can proceed at the restored replica. Failure to reestablish consistency can lead to irreparable damage.

Because this procedure involves substantial effort, it is intended for situations where serious damage has occurred. (For example, the disk containing a replica is unusable.)

Restoring a Replica from Backup

To restore a replica from backup:

1. Use your vendor database tools to restore a copy of the replicated database from backup.
2. Use the **restorereplica** command to start the restoration procedure.

   This command places a special lock on the replica. Between this point and the completion of Step 6, the **syncreplica –import** command adjusts the lock temporarily to permit application of the update, and then restores the full lock. During this time, only **syncreplica –import** can modify the replica.
3. Verify that all update packets have been processed at their destination replicas.
4. At the restored replica, generate update packets for all other replicas in the family, and send the packets to the sibling replicas.

   You can send the packets using your standard synchronization method. To recover the replica more quickly, create the packets with **syncreplica –export –fship**.

   Because your replica is in the special restoration state, each outgoing update packet includes a special request for a return acknowledgment. It also includes your replica's old epoch numbers, which are now its current epoch numbers, by virtue of the restoration in Step 1. Each destination replica uses these numbers to roll back its row for your replica.
5. Wait for each replica in the family to send an update packet to the restored replica. As in Step 4, you can accelerate the creation and delivery of the update packets.

Collectively, these update packets include all the operations that occurred between the time of the backup and the last update that your replica sent out before its storage was lost, including operations that originated at your replica. (The packets also include more recent operations that originated at other replicas.) In addition, each incoming packet includes the requested return acknowledgment from the sending host.

6 Process the incoming update packets with **syncreplica –import**. When your replica has received return acknowledgments from all other replicas in its family, **syncreplica –import** reports that restoration of the replica is complete:

```
Database <name> is unlocked after restoration.
```

Development work in the replica can now resume.

MultiSite Reference Pages

MultiSite Reference Pages

11

This section of the *Administrator's Guide* contains MultiSite reference pages.

activate

Prepares a database set to be replicated

Applicability

Product	Command type
MultiSite	multiutil subcommand

Platform
UNIX
Windows

Synopsis

activate [**–dbset** *dbset-name*] **–u·ser** *user-name* [**–p·assword**] *password*
–cl·an *clan-name* **–site** *site-name* **–host** *hostname*

Description

The **activate** command prepares a database set (a schema repository and its user databases) to be replicated. Before you can activate a database set, you must upgrade all user databases in the set to the same version of ClearQuest.

When you activate a database set, you give it a clan name and site name. The logical name of the database set is changed to **CQMS.***clan-name***.***site-name*. After you activate a database set, it can be accessed by additional **multiutil** commands, and you can use the **mkreplica** command to replicate one or more user databases in that database set.

Note: You need to activate a database set only once, before creating the first replica.

Restrictions

You must have Super User privileges.

Options and Arguments

–dbset *dbset-name*
The name of the database set you want to activate. You can omit this argument

if your ClearQuest installation has only one database set. The database set names are listed in the Maintenance Tool under **Existing Connections**.

Specifying a User Name and Password

Default
: You must specify a user name and password.

–u·ser *user*
: Name of a user with Super User privileges.

–p·assword *password*
: Password associated with the specified user.

Specifying the Clan and Site

Default
: None.

–cl·an *clan-name*
: Name of the replica's clan.

–site *site-name*
: Name of the replica's site.

Specifying the Host

Default
: None.

–host *hostname*
: Name of the synchronization server, which is the host where the storage and return bays for the replica are located. The Rational Shipping Server must be installed on the synchronization server.

Examples

In this example, the lines are broken for readability. You must enter the command on a single physical line.

- Activate the default database set. Name the clan **telecomm** and name the site **boston_hub**. This site uses **minuteman** as its synchronization server.

  multiutil activate –dbset CLSIC –user susan –p passwd –clan telecomm –site boston_hub –host minuteman

See Also

mkreplica

chepoch

Changes epoch number estimates

Applicability

Product	Command type
MultiSite	multiutil subcommand

Platform
UNIX
Windows

Synopsis

chepoch [**–cl·an** *clan-name*] [**–site** *site-name*] **–fam·ily** *family-name*
–u·ser *user-name* [**–p·assword**] *password*
{ [**–f·orce**] *replica* [*replica=value* ...] }

Description

This command changes a replica's epoch number estimates for other replicas. You cannot change a replica's own epoch numbers because they record the actual state of the replica. For more information about epoch numbers, see *The Operation Log* on page 8. For descriptions of scenarios using **chepoch**, see *Cannot Find Oplog Entry* on page 90 and *Recovering from Lost Packets* on page 96.

Restrictions

You must have Super User privileges.

Options and Arguments

Specifying the Clan, Site, and Family

Default

Clan: First clan replicated at this site. If there is more than one clan at the site, **–clan** is required.
Site: Current site. If there is more than one site on this host, **–site** is required.
Family: No default; you must specify a family.

–cl·an *clan-name*

Name of the replica's clan.

–site *site-name*

Name of the replica's site.

–fam·ily *family-name*

User database family: Database name given to the user database when it was created.

Schema repository family: The family name is **MASTR**.

Specifying a User Name and Password

Default

You must specify a user name and password.

–u·ser *user*

Name of a user with Super User privileges.

–p·assword *password*

Password associated with the specified user.

Suppressing Interactive Prompts

Default

You must confirm each change.

–f·orce

Suppresses confirmation steps.

Specifying the Changes

Default

You must specify the replica whose estimated epoch numbers are to be changed. **chepoch** reads a set of *replica=value* pairs, one per line, from standard input. You can copy and paste **lsepoch** output, or type the data in the format described below. Extra white space is allowed. To terminate input, type a period character (**.**) and a carriage return (<CR>) at the beginning of a line.

replica

Site name of the replica whose estimated epoch numbers are to be changed; that is, it changes the current replica's estimate of the state of *replica*.

replica=value

One or more arguments, where

replica	Column of the epoch number matrix. This argument, along with the preceding *replica* argument, specifies a particular location in the matrix.

value New epoch number to be entered at the specified matrix location.

Examples

In these examples, the lines are broken for readability. You must enter each command on a single physical line.

- Change two columns of epoch estimates in the **boston_hub** replica's row for the **sanfran_hub** replica.

  multiutil chepoch -clan telecomm -site boston_hub -family SAMPL -user susan -p passwd sanfran_hub sanfran_hub=100 boston_hub=350

  ```
  Multiutil: Change the estimate for the epochs of site 'sanfran_hub'
  replayed at site 'sanfran_hub' to 100 [yes|NO|quit] yes
  Multiutil: Change the estimate for the epochs of site 'boston_hub'
  replayed at site 'sanfran_hub' to 350 [yes|NO|quit] yes
  Multiutil: 2 epoch estimate(s) for site 'sanfran_hub' successfully
  changed; 0 failures.
  Multiutil: Estimates of the epochs from each site replayed at site
  'sanfran_hub' (@goldengate):
  BOSTON_HUB: 350
  SANFRAN_HUB: 100
  ```

- Similar to preceding example, but use **–force** to suppress confirmation steps.

  multiutil chepoch -clan telecomm -site boston_hub -family SAMPL -user susan -p passwd sanfran_hub sanfran_hub=100 boston_hub=350 -force

  ```
  Multiutil: Estimates of the epochs from each site replayed at site
  'sanfran_hub' (@goldengate):
  BOSTON_HUB: 350
  SANFRAN_HUB: 100
  ```

See Also

lsepoch, **recoverpacket**, **restorereplica**

chmaster

Transfers mastership of an object

Applicability

Product	Command type
MultiSite	multiutil subcommand

Platform
UNIX
Windows

Synopsis

chmaster [**–cl**·**an** *clan-name*] [**–site** *site-name*] **–fam**·**ily** *family-name*
–u·**ser** *username* [**–p**·**assword**] *password new-master-replica*
{ { *entity-selector...* | { **–all** [**–l**·**ong**] | **–workingmaster** }
[**–force** *obsolete-replica* | **–forceall**] }
}

Description

This command transfers the mastership of one or more objects from one replica to another. Only the current replica is affected immediately; other replicas are notified of the mastership transfers through the normal exchange of update packets.

Restrictions

Identities: You must have Super User privileges.

Mastership: Your current replica must master the object. Using **–force** or **–forceall** overrides this restriction, but you must not use these options except in special circumstances.

Options and Arguments

Specifying the Clan, Site, and Family

Default

Clan: First clan replicated at this site. If there is more than one clan at the site, **–clan** is required.
Site: Current site. If there is more than one site on this host, **–site** is required.
Family: No default; you must specify a family.

–cl·an *clan-name*

Name of the replica's clan.

–site *site-name*

Name of the replica's site.

–fam·ily *family-name*

User database family: Database name given to the user database when it was created.

Schema repository family: The family name is **MASTR**. You must specify **MASTR** when you use the **–workingmaster** option.

Specifying a User Name and Password

Default

You must specify a user name and password.

–u·ser *user*

Name of a user with Super User privileges.

–p·assword *password*

Password associated with the specified user.

Specifying the New Master Replica

Default

None.

new-master-replica

The name of the replica to which you are transferring mastership.

Specifying Objects

Default

None. You must specify a particular object (*entity-selector*), all objects in a replica (**–all**), or only the working schema repository.

entity-selector

Specifies the object whose mastership you want to change. You can change mastership of the following objects:

Object	Syntax
Record	*record-type*:*record-id*
User or group	**user:***username* **group:***group-name*
Public Workspace item	"**workspace:Public Queries***folder-name**query-name*"
Personal Workspace item	"**workspace:Personal Queries(***username***)***folder-name**query-name*"
Stateless record whose name is not unique	*record-type*:*record-id***<***keysite-name***>**
Workspace item whose name is not unique	**"workspace:***query-name***<***keysite-name***>"**
User or group whose name is not unique	**user:***username***<***keysite-name***>** **group:***group-name***<***keysite-name***>**

For information about making names unique, see *Resolving Naming Conflicts* on page 98.

–a·ll [–l·ong]

Transfers to *new-master-replica* mastership of all objects located in and mastered by the replica you specify with **–clan**, **–site**, and **–family**. If errors occur, the command continues, but after finishing, it reports that not all mastership changes succeeded.

With **–long, chmaster** lists the objects whose mastership is changing.

Note: To change mastership of a working schema repository, use **–workingmaster**.

–workingmaster

Transfers mastership of a working schema repository to the site you specify. You can use this option only at the site of the working schema repository.

When you use this option, you must specify **–family MASTR**.

–f·orce *obsolete-replica*

Warning: Incorrect use of the **–force** option can lead to divergence among the replicas in a family.

With **–force**, **chmaster** transfers mastership of all objects in the replica specified with *obsolete-replica*. Use this form of **chmaster** only when replica *obsolete-replica* is no longer available (for example, it was deleted accidentally).

–f·orceall

Warning: Incorrect use of the **–forceall** option can lead to divergence among the replicas in a family.

With **–forceall**, **chmaster** transfers mastership of an object to a specified replica, even if the current replica does not master the object.

Examples

In these examples, the lines are broken for readability. You must enter each command on a single physical line.

- Transfer mastership of the user **admin** from the **boston_hub** replica to **sanfran_hub**.

  multiutil chmaster -clan telecomm -site boston_hub -family DEV -user susan -p passwd sanfran_hub user:admin

  ```
  Multiutil: The mastership of record 'admin' of type 'user' has been
  changed from 'BOSTON_HUB' to 'SANFRAN_HUB'.

  Multiutil: The mastership of some users or groups has been
  transferred from this site. The local user administrator must update
  user databases at the new mastering site 'sanfran_hub' before these
  changes will be visible to any user database.
  ```

- At the **tokyo** replica, which contains the working schema repository, transfer mastership of all schemas and working schema repository tasks to the **sydney** replica.

  multiutil chmaster -clan testclan -site tokyo -family MASTR -user masako -p passwd sydney -workingmaster

  ```
  Multiutil: The working master has been changed from 'TOKYO' to
  'SYDNEY'.
  ```

- Transfer mastership of all objects in the **DEV** database, mastered by the **sanfran_hub** replica, to the **boston_hub** replica.

  multiutil chmaster -clan telecomm -site sanfran_hub -family DEV -user jcole -p passwd boston_hub -all

  ```
  Multiutil: Total number of objects changed: 5.
  ```

- Similar to previous example, but use the **–long** option.

  multiutil chmaster -clan telecomm -site sanfran_hub -family DEV -user jcole -p passwd boston_hub -all -long

  ```
  Multiutil: The mastership(s) of the following object(s) in database
  'DEV' was(were) changed from 'SANFRAN_HUB' to 'BOSTON_HUB'.
  ```

```
Multiutil: Type: customer, display Name: John Smith.
Multiutil: Type: customer, display Name: Ethan Hunt.
Multiutil: Type: customer, display Name: Jane Smith.
Multiutil: Type: customer, display Name: Anne Johnson.
Multiutil: Type: customer, display Name: Joe Lee.
```

- At the **boston_hub** replica, transfer mastership of all items mastered by the **bangalore** replica to the **boston_hub** replica. Assume that **bangalore** is no longer available.

  multiutil chmaster -clan telecomm -site boston_hub -family DOC -user susan -password passwd boston_hub -all -force bangalore

  Change mastership of users and groups from the **bangalore** schema repository to **boston_hub**.

  multiutil chmaster -clan telecomm -site boston_hub -family MASTR -user susan -password passwd boston_hub -all -force bangalore

  Change mastership of the working schema repository from **bangalore** to **boston_hub**:

  multiutil chmaster -clan telecomm -site boston_hub -family MASTR -user susan -password passwd boston-hub -workingmaster -force bangalore

- At **boston_hub**, use **–forceall** to change mastership of user **admin** from **sanfran_hub** to **tokyo**.

  multiutil chmaster -clan telecomm -site boston_hub -family DEV -user susan -p passwd tokyo user:admin –forceall

  ```
  Multiutil: The mastership of record 'admin' of type 'user' has been
  changed from 'SANFRAN_HUB' to 'TOKYO'.
  ```

See Also

describe, **syncreplica**
Chapter 9, *Managing Mastership*

chreplica

Changes the properties of replicas in a site

Applicability

Product	Command type
MultiSite	multiutil subcommand

Platform
UNIX
Windows

Synopsis

chrep·lica [**–cl·an** *clan-name*] [**–site** *site-name*]
–u·ser *username* [**–p·assword**] *password*
[**–host** *hostname* | **–size** *id-block-size* | **–thres·hold** *id-block-threshold*]
replica-selector

Description

Use this command to change the synchronization server host information or ID block allocation for all replicas in a site. For more information, see *Moving or Renaming a Synchronization Server* on page 71 and *Changing Allocation of ID Blocks to a Replica* on page 72.

Restrictions

None.

Options and Arguments

Specifying the Clan and Site

Default

Clan: First clan replicated at this site. If there is more than one clan at the site, **–clan** is required.
Site: Current site. If there is more than one site on this host, **–site** is required.

–cl·an *clan-name*
: Name of the replica's clan.

–site *site-name*
: Name of the replica's site.

Specifying a User Name and Password

Default
: You must specify a user name and password.

–u·ser *user*
: Name of a user with Super User privileges.

–p·assword *password*
: Password associated with the specified user.

Specifying the New Values

Default
: You must specify the site to be changed. The default ID block size is 4096 and the default threshold is 25 percent.

–host *hostname*
: Name of the new synchronization server (on which the Rational Shipping Server is installed).

–size *id-block-size*
: Size of ID block. You can enter any number from 1 to 1023. The value of *id-block-size* is multiplied by 100 to obtain the actual ID block size. For example, to specify an ID block of 30,000, use the number 300; to specify an ID block of 25,000, use the number 250.

–thres·hold *id-block-threshold*
: Number of record ID numbers allocated to the replica. *id-block-threshold* is specified as a integer, representing a percentage. You can enter any number from 1 to 63. When the number of remaining record IDs to be used drops below the specified percentage of the current ID block size, an additional block is allocated.

replica-selector
: Site to be changed.

Examples

In these examples, the lines are broken for readability. You must enter each command on a single physical line.

- Associate the **sanfran_hub** replica with the synchronization server **goldengate**.

  multiutil chreplica -clan telecomm -site sanfran_hub -user jcole -p passwd -host goldengate sanfran_hub

- Associate the **sanfran_hub** replica with the synchronization server **goldengate** and specify an ID block size of 10,000.

  multiutil chreplica -clan telecomm -site sanfran_hub -user jcole -p passwd -host goldengate -size 100 sanfran_hub

- Associate the **sydney** replica with the synchronization server **taronga** and specify an allocation threshold of 55 percent.

  multiutil chreplica -clan testclan -site sydney -user bfife -p passwd -host taronga -threshold 55 sydney

See Also

chmaster, **syncreplica**

control_panel

Sets the e-mail parameters for Rational Shipping Server e-mail notification on Windows

Applicability

Product	Command type
MultiSite	multiutil subcommand

Platform
Windows

Synopsis

control_panel –admin *admin-email* **–smtp** *smtp-server-host*
[**–enable_shipping_server_email_notification**]

Description

Use this command to enable e-mail notification when you use the Rational Shipping Server with ClearQuest MultiSite on the Windows platform.

If you use ClearQuest MultiSite but not ClearCase MultiSite, use this command to specify the e-mail address and server to use to receive e-mail notification about Rational Shipping Server operations.

If you use ClearQuest MultiSite and ClearCase MultiSite, both products use the same e-mail address for notification e-mail.

- To use the same e-mail address for Rational Shipping Server operations for ClearQuest MultiSite and ClearCase MultiSite, do not use this command. Configure e-mail notification with the MultiSite Control Panel options, using the instructions in the *Administrator's Guide* for Rational ClearCase MultiSite.
- To use a different e-mail address for Rational Shipping Server operations originating from ClearQuest MultiSite, use this command to indicate the e-mail address you want to use.

Restrictions

None.

Options and Arguments

–admin *admin-email*
The e-mail address to use for receiving and sending e-mail notification for errors and information sent from the Rational Shipping Server. Specify the e-mail address in the following format, where *tld* is a top level domain:

username@domain.tld

–smtp *smtp-server-host*
The name of the SMTP host that is used with the e-mail address you specified with **–admin**. If you are using ClearCase MultiSite and specify a different e-mail address for use with ClearQuest MultiSite, you must use the same SMTP server that is used for ClearCase notification.

–enable_shipping_server_email_notification
Use this argument to enable e-mail notification for ClearQuest MultiSite operations that use the Rational Shipping Server.

Default: If this option is not used, ClearQuest MultiSite uses the e-mail settings for ClearCase MultiSite. If ClearCase MultiSite is not installed, you must use this option to enable e-mail notification for the Rational Shipping Server.

Examples

In these examples, the lines are broken for readability. You must enter each command on a single physical line.

- Enables e-mail notification for the Rational Shipping Server for ClearQuest MultiSite (ClearCase MultiSite is not installed):

  multiutil control_panel -admin susan@purpledoc.com -smtp mailsrv0.purpledoc.com -enable_shipping_server_email_notification

- Sets up a separate e-mail address to use for ClearQuest MultiSite when ClearCase MultiSite is also being used:

  multiutil control_panel -admin susan@purpledoc.com -smtp mailsrv0.purpledoc.com

See Also

MultiSite Control Panel

describe

Lists the master replica of an object

Applicability

Product	Command type
MultiSite	multiutil subcommand

Platform
UNIX
Windows

Synopsis

describe [**–cl**·**an** *clan-name*] [**–site** *site-name*] **–fam**·**ily** *family-name*
–u·**ser** *username* [**–p**·**assword**] *password*
[**–all** | **–local** | *object-selector* ...]

Description

This command lists the master replica of one or more objects in a replica. To determine which replica masters a record, look at the value of the **ratl_mastership** field.

Restrictions

You must have Super User privileges.

Options and Arguments

Specifying the Clan, Site, and Family

Default

Clan: First clan replicated at this site. If there is more than one clan at the site, **–clan** is required.
Site: Current site. If there is more than one site on this host, **–site** is required.
Family: No default; you must specify a family.

–cl·**an** *clan-name*

Name of the replica's clan.

–site *site-name*

Name of the replica's site.

–fam·ily *family-name*

User database family: Database name given to the user database when it was created.

Schema repository family: The family name is **MASTR**.

Specifying a User Name and Password

Default

You must specify a user name and password.

–u·ser *user*

Name of a user with Super User privileges.

–p·assword *password*

Password associated with the specified user.

Specifying Objects to Be Described

Default

Lists the master replica of all objects in the specified replica.

–all

Lists the master replica of each item in the specified family.

–local

Lists only those objects mastered by the current site in the specified family.

object-selector ...

Objects to be described. Specify *object-selector* in one of the following forms:

Object	**Syntax**
Record	*record-type*:*record-id*
User or group	**user:***username* **group:***group-name*
Public Workspace item	"**workspace:Public Queries***folder-name**query-name*"
Personal Workspace item	"**workspace:Personal Queries(***username***)***folder-name**query-name*"
Stateless record whose name is not unique	*record-type*:*record-id***<***keysite-name***>**
Workspace item whose name is not unique	**"workspace:***query-name***<***keysite-name***>"**

Object	Syntax
User or group whose name is not unique	**user:***username***<***keysite-name***>** **group:***group-name***<***keysite-name***>**

For information about making names unique, see *Resolving Naming Conflicts* on page 98.

Examples

In these examples, the lines are broken for readability. You must enter each command on a single physical line.

- At the **sanfran_hub** replica, list the master replica for the **Customer** object "Jane Smith".

  multiutil describe -clan telecomm -site sanfran_hub -family DEV -user jcole -p passwd Customer:"Jane Smith"

  ```
  Multiutil: Mastership of 'Customer:Jane Smith' is 'BOSTON_HUB'.
  ```

- List the master replicas of all objects in the **sydney** replica.

  multiutil describe -clan testing -site sydney -family TEST -user bfife -p passwd

  ```
  Multiutil: Mastership of 'Defect:TEST00000001' is 'TOKYO'.
  Multiutil: Mastership of 'Defect:TEST00000002' is 'TOKYO'.
  Multiutil: Mastership of 'Defect:TEST00000004' is 'TOKYO'.
  Multiutil: Mastership of 'Email_Rule:New Submissions' is 'TOKYO'.
  Multiutil: Mastership of 'bucket:Personal Queries' is 'TOKYO'.
  ```

See Also

chmaster, **syncreplica**

dumpoplog

Lists the contents of a replica's operation log

Applicability

Product	Command type
MultiSite	multiutil subcommand

Platform
UNIX
Windows

Synopsis

dumpoplog [**–cl·an** *clan-name*] [**–site** *site-name*] **–fam·ily** *family-name*
–u·ser *username* [**–p·assword**] *password*
[**–l·ong** | **–s·hort**] [**–at** *replica*]
[*oplog-ID...* | [**–from** *oplog-ID*] [**–to** *oplog-ID*]]
[**–since** *date-time*] [**–reverse**]

Description

Use **dumpoplog** to list information in a replica's operation log (oplog). The oplog tracks all database transactions, including record changes and schema modifications. Each oplog entry has an oplog ID.

Restrictions

You must have Super User privileges.

Options and Arguments

Specifying the Clan, Site, and Family

Default

Clan: First clan replicated at this site. If there is more than one clan at the site, **–clan** is required.
Site: Current site. If there is more than one site on this host, **–site** is required.
Family: No default; you must specify a family.

–cl·an *clan-name*
Name of the replica's clan.

–site *site-name*
Name of the replica's site.

–fam·ily *family-name*
User database family: Database name given to the user database when it was created.

Schema repository family: The family name is **MASTR**.

Specifying a User Name and Password

Default
You must specify a user name and password.

–u·ser *user*
Name of a user with Super User privileges.

–p·assword *password*
Password associated with the specified user.

Specifying the Information to Display

Default
If no format is specified, the **–short** format is used. All oplog entries are listed.

–l·ong | **–s·hort**
With **–long**, displays all columns of the oplog, including information about the schema revision that applies to the packet data. With **–short**, displays each database operation that took place.

–at *replica*
Lists the oplog entries that originated from the sites you specify.

oplog-ID...
Lists the oplog entries you specify.

–from *oplog-ID*
Lists a range of oplog entries starting with *oplog-ID* and ending with the latest one or the one specified with **–to**. Specify oplog IDs as integers.

–to *oplog-ID*
Lists a range of oplog entries ending with *oplog-ID* and starting with 1 or the one specified with **–from**. Specify oplog IDs as integers.

–since *date-time*
Lists all oplog entries after *date-time*. The *date-time* argument can have any of the following formats:

*date***.***time* | *date* | *time* | **now**
where:

date	:=	*day-of-week* | *long-date*
time	:=	*h*[*h*]**:***m*[*m*][**:***s*[*s*]] [**UTC** [[**+** | **-**]*h*[*h*][**:***m*[*m*]]]]
day-of-week	:=	**today** | **yesterday** | **Sunday** | ... | **Saturday** | **Sun** | ... | **Sat**
long-date	:=	*d*[*d*]**–***month*[**–**[*yy*]*yy*]
month	:=	**January** | ... | **December** | **Jan** | ... | **Dec**

Specify the *time* in 24-hour format, relative to the local time zone. If you omit the time, the default value is **00:00:00**. If you omit the *date*, the default value is **today**. If you omit the century, year, or a specific date, the most recent one is used. Specify **UTC** if you want the time to be resolved to the same moment in time regardless of time zone. Use the plus (+) or minus (-) operator to specify a positive or negative offset to the UTC time. If you specify **UTC** without hour or minute offsets, the default setting is Greenwich Mean Time (GMT). (Dates before January 1, 1970 Universal Coordinated Time (UTC) are invalid.)

Examples:

```
22-November-2002
sunday
yesterday.16:00
8-jun
13:00
today
9-Aug.10:00UTC
```

–reverse

Reverse the order of the list of oplog entries.

Examples

In these examples, the lines are broken for readability. You must enter each command on a single physical line.

- List the oplog of operations associated with the **DOC** family at the **boston_hub** replica.

  multiutil dumpoplog -clan telecomm -site boston_hub -family DOC -user susan -p passwd

  ```
  Item    1, 22-Jan-2002.10:50:21 TABLE_PACKLET    (epoch
  BOSTON_HUB.1    initiated 22-Jan-2002.10:50:21)
  Item    2, 22-Jan-2002.10:50:21 TABLE_PACKLET    (epoch
  BOSTON_HUB.2    initiated 22-Jan-2002.10:50:21)
  Item    3, 22-Jan-2002.10:50:21 TABLE_PACKLET    (epoch
  BOSTON_HUB.3    initiated 22-Jan-2002.10:50:21)
  Item    4, 22-Jan-2002.10:50:22 TABLE_PACKLET    (epoch
  BOSTON_HUB.4    initiated 22-Jan-2002.10:50:22)
  Item    5, 22-Jan-2002.10:50:22 TABLE_PACKLET    (epoch
  BOSTON_HUB.5    initiated 22-Jan-2002.10:50:22)
  Item    6, 22-Jan-2002.11:06:21 ACTION_PACKLET   (epoch
  BOSTON_HUB.6    initiated 22-Jan-2002.11:06:21)
  Item    7, 22-Jan-2002.11:10:35 TABLE_PACKLET    (epoch
  BOSTON_HUB.7    initiated 22-Jan-2002.11:10:35)
  Item    8, 22-Jan-2002.11:10:36 EXPORT_PACKLET   to SANFRAN_HUB
  ```

- List the oplog of the **sanfran_hub** operations at the **sanfran_hub** replica.

  multiutil dumpoplog -clan telecomm -site sanfran_hub -family DEV -user jcole -p passwd -short -at sanfran_hub

  ```
  Item    3, 07-Feb-2002.11:20:29 ACTION_PACKLET   (epoch
  SANFRAN_HUB.1    initiated 07-Feb-2002.11:20:29)
  Item    4, 07-Feb-2002.11:23:24 TABLE_PACKLET    (epoch
  SANFRAN_HUB.2    initiated 07-Feb-2002.11:23:24)
  Item    6, 07-Feb-2002.11:24:27 ACTION_PACKLET   (epoch
  SANFRAN_HUB.3    initiated 07-Feb-2002.11:24:26)
  Item    7, 07-Feb-2002.11:24:49 TABLE_PACKLET    (epoch
  SANFRAN_HUB.4    initiated 07-Feb-2002.11:24:49)
  ```

- List the oplog of all operations in the **bangalore** replica as of January 28, 2002

  multiutil dumpoplog -clan telecomm -site bangalore -family DOC -user masako -p passwd -short -since 28-Jan-2002

```
Item   1, 28-Jan-2002.10:11:37 TABLE_PACKLET   (epoch BOSTON_HUB.1
initiated 28-Jan-2002.10:11:37)
Item   2, 28-Jan-2002.10:11:37 TABLE_PACKLET   (epoch BOSTON_HUB.2
initiated 28-Jan-2002.10:11:37)
Item   3, 28-Jan-2002.10:11:37 TABLE_PACKLET   (epoch BOSTON_HUB.3
initiated 28-Jan-2002.10:11:37)
Item   4, 28-Jan-2002.10:11:37 TABLE_PACKLET   (epoch BOSTON_HUB.4
initiated 28-Jan-2002.10:11:37)
Item   5, 28-Jan-2002.10:11:37 TABLE_PACKLET   (epoch BOSTON_HUB.5
initiated 28-Jan-2002.10:11:37)
Item   6, 28-Jan-2002.10:11:37 TABLE_PACKLET   (epoch BOSTON_HUB.6
initiated 28-Jan-2002.10:11:37)
```

- List oplog entries 3 through 6 from the **boston_hub** replica.

  multiutil dumpoplog -clan telecomm -site boston_hub -family DEV -user susan passwd -long -from 3 -to 6

```
Item    3, 28-Jan-2002.10:11:37 TABLE_PACKLET      334 (epoch
BOSTON_HUB.3    initiated 28-Jan-2002.10:11:37) Schema: 14, Schema
Rev: 1, SchemaRev Ver:1
Item    4, 28-Jan-2002.10:11:37 TABLE_PACKLET      336 (epoch
BOSTON_HUB.4    initiated 28-Jan-2002.10:11:37) Schema: 14, Schema
Rev: 1, SchemaRev Ver:1
Item    5, 28-Jan-2002.10:11:37 TABLE_PACKLET      340 (epoch
BOSTON_HUB.5    initiated 28-Jan-2002.10:11:37) Schema: 14, Schema
Rev: 1, SchemaRev Ver:1
Item    6, 28-Jan-2002.10:11:37 TABLE_PACKLET      355 (epoch
BOSTON_HUB.6    initiated 28-Jan-2002.10:11:37) Schema: 14, Schema
Rev: 1, SchemaRev Ver:1
```

- List the first oplog entry in the **tokyo** replica.

  multiutil dumpoplog -clan testclan -site tokyo -family TEST -user masako -p passwd -long -from 1

```
Item    1, 29-Jan-2002.13:22:44 SPECIAL_PACKLET     88 (epoch
TOKYO.1    initiated 29-Jan-2002.13:22:44) Schema: 10, Schema Rev:
1, SchemaRev Ver:0
```

See Also

lsepoch, **scruboplog**

lsepoch

Lists epoch information

Applicability

Product	Command type
MultiSite	multiutil subcommand

Platform
UNIX
Windows

Synopsis

lsepoch [**–cl·an** *clan-name*] [**–site** *site-name*] **–fam·ily** *family-name*
–u·ser *username* [**–p·assword**] *password*
[*replica* ...]

Description

This command displays the epoch number matrix for a replica. The replica's own epoch row in its matrix represents its actual state. The other rows represent the replica's estimate of the other replicas' states.

Note: lsepoch output includes rows for deleted replicas, in addition to the rows for replicas still in use. Oplog records for deleted replicas are saved in case a replica undergoing restoration must receive operations from the deleted replica. (For example, a replica may be restored from a backup created before the deleted replica was removed.)

Restrictions

You must have Super User privileges. If you do not specify a clan or site, the user name and password must be valid for all local instances of clans and sites.

Options and Arguments

Specifying the Clan, Site, and Family

Default
Lists epoch estimates for all local clans and sites.

–cl·an *clan-name*
Clan for which you want to list epoch information.

–site *site-name*
Site for which you want to list epoch information.

–fam·ily *family-name*
Family for which you want to list epoch information.

User database family: Database name given to the user database when it was created.

Schema repository family: The family name is **MASTR**.

Specifying a User Name and Password

Default
You must specify a user name and password.

–u·ser *user*
Name of a user with Super User privileges.

–p·assword *password*
Password associated with the specified user.

Specifying the Replica

Default
Lists the epoch numbers for each replica in the family. If you do not specify a clan or site, the epoch estimates for all local instances of clans and sites, respectively, are listed.

replica ...
Site of the replica for which you want to list epoch information.

Examples

In these examples, the lines are broken for readability. You must enter each command on a single physical line.

- List epoch numbers in the **boston_hub** replica for the entire **DEV** family. In this example, the only replicas in the family are **boston_hub** and **sanfran_hub**.

  multiutil lsepoch -clan telecomm -site boston_hub -family DEV -user susan -p passwd

  ```
  Multiutil: Estimates of the epochs from each site replayed at site
  'BOSTON_HUB' (@minuteman):
  BOSTON_HUB: 4
  SANFRAN_HUB: 4
  Multiutil: Estimates of the epochs from each site replayed at site
  'SANFRAN_HUB' (@goldengate):
  BOSTON_HUB: 4
  SANFRAN_HUB: 4
  ```

- List the **boston_hub** replica's estimate of the state of replica **sanfran_hub** in the **DEV** family.

  multiutil lsepoch -clan telecomm -site boston_hub -family DEV -user susan -p passwd sanfran_hub

  ```
  Multiutil: Estimates of the epochs from each site replayed at site
  'sanfran_hub' (@goldengate):
  BOSTON_HUB: 5
  SANFRAN_HUB: 3
  ```

- List epoch numbers in the **tokyo** replica for the entire **MASTR** family. In this example, you can see that the **sydney** replica needs an update from the **tokyo** replica.

  multiutil lsepoch -clan testclan -site tokyo -family MASTR -user masako -p passwd

  ```
  Multiutil: Estimates of the epochs from each site replayed at site
  'TOKYO' (@shinjuku):
  TOKYO: 1
  SYDNEY: 0
  Multiutil: Estimates of the epochs from each site replayed at site
  'SYDNEY' (@taronga):
  TOKYO: 0
  SYDNEY: 0
  ```

See Also

chepoch, **recoverpacket**, **restorereplica**

lspacket

Describes contents of a packet

Applicability

Product	Command type
MultiSite	multiutil subcommand

Platform
UNIX
Windows

Synopsis

lspacket [**–l·ong** | **–s·hort**] [*pname* ...]

Description

This command lists a summary of the contents of one or more files that contain replica-creation or update packets. By default, the **lspacket** output includes this information:

- Pathname of each packet
- Type of each packet (replica creation or update)
- Date of generation
- Originating replica
- Clan and family to which the packet applies
- Replicas for which the packet is intended
- Packet sequence number (for a file storing one part of a logical packet that has been split into multiple physical packets)

Restrictions

None.

Options and Arguments

Listing Format

Default

Includes the information listed in the *Description* section.

–l·ong

In addition to the default information, lists the name of the replica where the packet was created and the oplog IDs that indicate the contents of the packet.

–s·hort

Lists only the pathname of a packet.

Specifying the Packets

Default

Lists all packets in all storage bays on the current host.

pname ...

One or more pathnames of files and/or directories.

Each file you specify is listed if it contains a physical packet. For each directory you specify, **lspacket** lists packets stored in that directory.

Examples

- List the contents of the lab.xml update packet.

  multiutil lspacket -long c:\cqms\lab.xml

```
Multiutil: Packet 'c:\cqms\lab.xml'...
Multiutil:      Type: 'CREATE_PACKET'
Multiutil:      Sent: 2002-01-22 10:58:11
Multiutil:      From: BOSTON_HUB
(B6A316BE-CCB4-11D5-AFB5-00B0D0682333)
Multiutil:      Clan: 'TELECOMM'
Multiutil:      Recipients: SANFRAN_HUB
Multiutil:      Family: 'DEV'
```

- List all packets in the local host's storage bays.

  multiutil lspacket

```
Multiutil: Packet
'C:\temp\cqms\ms_ship\incoming\mk_TOKYO_29-January-02_09-47-27.xml'
...
Multiutil:      Type: 'CREATE_PACKET'
Multiutil:      Sent: 2002-01-29 09:47:28
Multiutil:      From: TOKYO (B6A316BE-CCB4-11D5-AFB5-00B0D0682333)
Multiutil:      Clan: 'TESTING'
Multiutil:      Recipients: SYDNEY
Multiutil:      Family: 'TEST'
Multiutil: Packet
'C:\temp\cqms\ms_ship\incoming\sync_SANFRAN_HUB_07-February-02_11-2
4-49.xml'...
Multiutil:      Type: 'UPDATE_PACKET'
Multiutil:      Sent: 2002-02-07 11:24:49
Multiutil:      From: SANFRAN_HUB
(8AB1A196-BE48-47F1-9255-71FD18D7309D)
Multiutil:      Clan: 'TELECOMM'
Multiutil:      Recipients: BOSTON_HUB
Multiutil:      Family: 'DEV'
```

- List all packets in the local host's storage bays, with **–short**.

  multiutil lspacket -short

```
Multiutil: Packet
'C:\temp\cqms\ms_ship\incoming\mk_TOKYO_29-January-02_09-47-27.xml'
...
Multiutil: Packet
'C:\temp\cqms\ms_ship\incoming\sync_SANFRAN_HUB_07-February-02_11-2
4-49.xml'...
```

See Also

mkreplica, **MultiSite Control Panel**, **syncreplica**, **shipping.conf**

lsreplica

Lists database replicas

Applicability

Product	Command type
MultiSite	multiutil subcommand

Platform
UNIX
Windows

Synopsis

- List database replicas:

  lsrep·lica [**–cl·an** *clan-name*] [**–site** *site-name*] **–fam·ily** *family-name*
  –u·ser *username* [**–p·assword**] *password*
  [**–l·ong** | **–s·hort** | **–fmt** *format*]
  [**–sib·lings**
  | [**–sib·lings**] **–infa·mily** *family-name*
  | *replica* ...]

- List the working schema repository for a family:

  lsrep·lica [**–cl·an** *clan-name*] [**–site** *site-name*] **–fam·ily** *family-name*
  –u·ser *username* [**–p·assword**] *password* **–working·master**

Description

This command lists information about all active replicas known to the current replica. You can list all replicas in a clan or in a family within a clan. Other replicas may exist, but the packets that contain their creation information have not yet been imported at the current replica.

Restrictions

You must have Super User privileges.

Options and Arguments

Specifying the Clan, Site, and Family

Default

Clan: First clan replicated at this site. If there is more than one clan at the site, **–clan** is required.
Site: Current site. If there is more than one site on this host, **–site** is required.
Family: No default; you must specify a family.

–cl·an *clan-name*

Name of the replica's clan.

–site *site-name*

Name of the replica's site.

–fam·ily *family-name*

User database family: Database name given to the user database when it was created.

Schema repository family: The family name is **MASTR**.

You can use the **–siblings** option or the **–siblings –infamily** options to list the replicas in a specific family within the clan, as known to your schema repository.

Specifying a User Name and Password

Default

You must specify a user name and password.

–u·ser *user*

Name of a user with Super User privileges.

–p·assword *password*

Password associated with the specified user.

Listing Format

Default

Includes creation event information for each replica.

–l·ong

Lists each replica's creation information and synchronization server. If the current replica is in the process of restoration, this option annotates the listings of other replicas from which restoration updates are required. (See the **restorereplica** reference page.)

–s·hort

Lists only replica names.

–fmt *format-string*

Lists information using the specified format string, which uses conversion specifications that identify items to display and specify their display format. The conversion specification format resembles that of the C-language function **printf():** a percent sign (%) and a key letter (lowercase), which indicates the kind of data to display.

Unlike **printf()** specifiers, conversion specifications are not replaced by arguments supplied elsewhere on the command line; they are replaced automatically by **multiutil** with field values extracted from the replica.

format-string is a character string, composed of alphanumeric characters, conversion specifications, and escape sequences. It must be enclosed in double quotes (**"**).

Conversion specifications:

%h	Host name
%n	Name of replica
%c	Clan name
%f	Family name
%d	Description of replica, if any.
%s	Status of replica
%%	% character
%z	ID block size
%t	ID block threshold

Escape sequences:

\n	<NL>
\t	<TAB>
\′	Single quote
\\	Literal (uninterpreted) backslash
nnn	Character specified by octal code

Specifying the Replica

Default
Lists all known replicas in the current replica's family, including the current replica.

–working·master
Lists the working schema repository for the clan you specified.

–sib·lings
For a user database, lists the family members of the current replica, but does not list the current replica itself. For a schema repository, lists the family members of all replicas in the site, but not the replicas in the current site. This option is useful when you are writing scripts that process only sibling replicas.

–infa·mily *family*
Lists the replicas in the family of the specified replica. Use the site name to specify the replica. This option can be used only if you specified **MASTR** with the **–family** option.

replica ...
The **–site** option specifies the replica that you are querying for information; this argument specifies the site of the replica for which you want to list information. You can only list replicas that are members of the same family.

Examples

In these examples, the lines are broken for readability. You must enter each command on a single physical line.

- List the replicas in the **DEV** family of the **telecomm** clan.

  multiutil lsreplica -clan telecomm -site boston_hub -family DEV -user susan -p passwd -long

```
Name: BOSTON_HUB; Clan: TELECOMM; Family: DEV; Host: minuteman;
Status: NORMAL, NOT CONNECTED; Description: ; Block Size: 4096;
Block Threshold: 1024
Name: SANFRAN_HUB; Clan: TELECOMM; Family: DEV; Host: goldengate;
Status: NORMAL, NOT CONNECTED; Description: ; Block Size: 4096;
Block Threshold: 1024
Name: BANGALORE; Clan: TELECOMM; Family: DEV; Host: ramohalli;
Status: NORMAL, NOT CONNECTED; Description: ; Block Size: 4096;
Block Threshold: 1024
```

- List the siblings in the **DEV** family in the **telecomm** clan, but not the user database replica at the **boston_hub** site.

  multiutil lsreplica -clan telecomm -site boston_hub -family DEV -user susan -p passwd -long –siblings

  ```
  Name: SANFRAN_HUB; Clan: TELECOMM; Family: DEV; Host: goldengate;
  Status: NORMAL, NOT CONNECTED; Description: ; Block Size: 4096;
  Block Threshold: 1024
  Name: BANGALORE; Clan: TELECOMM; Family: DEV; Host: ramohalli;
  Status: NORMAL, NOT CONNECTED; Description: ; Block Size: 4096;
  Block Threshold: 1024
  ```

- List the clan members in the **telecomm** clan, but not the replicas at the **boston_hub** site.

  multiutil lsreplica -clan telecomm -site boston_hub -family MASTR -user susan -p passwd -long -siblings

  ```
  Name: SANFRAN_HUB; Clan: TELECOMM; Family: MASTR; Host: goldengate;
  Status: NORMAL, NOT CONNECTED; Description: ;Block Size: 4096; Block
  Threshold: 1024
  Name: SANFRAN_HUB; Clan: TELECOMM; Family: DEV; Host: goldengate;
  Status: NORMAL, NOT CONNECTED; Description: ;Block Size: 4096; Block
  Threshold: 1024
  Name: BANGALORE; Clan: TELECOMM; Family: MASTR; Host: ramohalli;
  Status: NORMAL, NOT CONNECTED; Description:; Block Size: 4096; Block
  Threshold: 1024
  Name: BANGALORE; Clan: TELECOMM; Family: DOC; Host: ramohalli;
  Status: NORMAL, NOT CONNECTED; Description: ;Block Size: 4096; Block
  Threshold: 1024
  ```

- List all user databases in the **TEST** family of the clan **testclan**, as known to the working schema repository.

  multiutil lsreplica -clan testclan -site sydney -family MASTR -user bfife -p passwd -long -infamily TEST

  ```
  Name: TOKYO; Clan: TESTING; Family: TEST; Host: shinjuku; Status:
  NORMAL, NOT CONNECTED; Description: ; Block Size: 4096; Block
  Threshold: 1024

  Name: SYDNEY; Clan: TESTING; Family: TEST; Host: taronga; Status:
  NORMAL, NOT CONNECTED; Description: ; Block Size: 4096; Block
  Threshold: 1024
  ```

- List the working schema repository for the **DEV** family in the **testclan** clan.

  multiutil lsreplica -clan testclan -site sydney -family DEV -user bfife -workingmaster

- Mimic the output from **lsreplica –long**. Note the use of single quotes to enclose the format string, which includes literal double quotes.

  multiutil lsreplica -clan testing -site tokyo -family TEST -user masako -password passwd -fmt "Name:%n; Clan:%c; Family:%f; Host:%h; Status:%s;"

  ```
  Name:TOKYO; Clan:TESTING; Family:TEST; Host:shinjuku;
  Status:NORMAL, NOT CONNECTED;
  Name:SYDNEY; Clan:TESTING; Family:TEST; Host:taronga;
  Status:NORMAL, NOT CONNECTED;
  ```

See Also

mkreplica

mkorder

Creates a shipping order for use by the store-and-forward facility

Applicability

Product	Command type
MultiSite	MultiSite command

Platform
UNIX
Windows

Synopsis

mkorder –dat·a *packet-pname* [**–scl·ass** *storage-class-name*]
[**–pex·pire** *date-time*] [**–not·ify** *e-mail-address*]
[**–c** *comment* | **–cq** | **–cqe** | **–nc**]
[**–shi·p –cop·y** | **–fsh·ip** [**–cop·y**] | **–out** *order-pname*] *destination* ...

Description

This command creates a shipping order file for an existing packet file or any other file. The shipping order is used by the shipping server to send the file to one or more destinations.

mkorder submits to the shipping server a packet that was created with **mkreplica –out** or **syncreplica –out**. You can also use **mkorder** to resubmit packets whose shipping orders have expired, and to transfer other files among sites. A shipping order must be located in the same directory as its associated packet or file.

Note: The shipping server deletes a packet after delivering it successfully (except when the destination is the local host). If you use this command to process a file that must be preserved at your site even after it is delivered to another site, you must specify the **–copy** option.

Restrictions

None.

Options and Arguments

Specifying the Packet File

Default

None.

–dat·a *packet-pname*

The pathname of the packet or file.

Note: If *packet-pname* contains a colon character (**:**), **mkorder** changes the colon to a period character (**.**) during processing. This change allows packets to be delivered to Windows machines, which do not allow colons in file names.

Specifying Where to Place the Shipping Order

Default

Creates a shipping order in the directory where the *packet-pname* file is located.

–scl·ass *class-name*

Specifies the storage class of the packet and shipping order. If you also use **–ship** or **–fship**, **mkorder** looks up the storage class in the shipping.conf file on UNIX or in the MultiSite Control Panel on Windows to determine the location of the storage bay to use.

If you omit this option but use **–ship** or **–fship**, **mkorder** places the shipping order in the storage bay location specified for the **–default** class in the shipping.conf file or the MultiSite Control Panel.

–shi·p –cop·y

–fsh·ip [–cop·y]

Creates a shipping order for *packet-pname*. Using **–fship** invokes **shipping_server** to send the packet. Using **–ship** places the shipping order in a storage bay. To send the packet, run **shipping_server**.**–copy** is required with **–ship**, and optional with **–fship**:

- With **–copy**, **mkorder** copies the *packet-pname* file to one of the store-and-forward facility's storage bays and places the shipping order in the bay. The copy is deleted after it is delivered successfully to all the destinations specified in the shipping order.
- Without **–copy**, **mkorder** does not copy *packet-pname*; **mkorder** places the shipping order in the directory where the file is located. *packet-pname* is deleted after it is delivered successfully to all the destinations specified in the shipping order.

–out *order-pname*

Places the shipping order in the specified file instead of in a storage bay. An error occurs if the file already exists.

Handling Packet-Delivery Failures

Default

If a packet cannot be delivered, it is sent through the store-and-forward facility to the administrator at the site of the originating replica. A mail message is sent to the store-and-forward administrator. This occurs after repeated attempts to deliver the packet have failed and the allotted time has expired; it can also occur when the destination host is unknown or a data file does not exist. The store-and-forward configuration settings specify the expiration period, the e-mail address of the administrator, and the notification program.

–pex·pire *date-time*

Specifies the time at which the store-and-forward facility stops trying to deliver the packet and generates a failure mail message instead. This option overrides the expiration period specified for the storage class in the shipping.conf file (UNIX) or MultiSite Control Panel (Windows).

The *date-time* argument can have any of the following formats:

*date***.***time* | *date* | *time* | **now**
where:

date	:=	*day-of-week* | *long-date*
time	:=	*h*[*h*]**:***m*[*m*][**:***s*[*s*]] [**UTC** [[**+** | **-**]*h*[*h*][**:***m*[*m*]]]]
day-of-week	:=	**today** | **yesterday** | **Sunday** | ... | **Saturday** | **Sun** | ... | **Sat**
long-date	:=	*d*[*d*]**–***month*[**–**[*yy*]*yy*]
month	:=	**January** | ... | **December** | **Jan** | ... | **Dec**

Specify the *time* in 24-hour format, relative to the local time zone. If you omit the time, the default value is **00:00:00**. If you omit the *date*, the default value is **today**. If you omit the century, year, or a specific date, the most recent one is used. Specify **UTC** if you want the time to be resolved to the same moment in time regardless of time zone. Use the plus (+) or minus (-) operator to specify a positive or negative offset to the UTC time. If you specify **UTC** without hour or minute offsets, the default setting is Greenwich Mean Time (GMT). (Dates before January 1, 1970 Universal Coordinated Time (UTC) are invalid.)

Examples:

```
22-November-2002
sunday
yesterday.16:00
8-jun
13:00
today
9-Aug.10:00UTC
```

–not·ify *e-mail-address*

The delivery-failure message is sent to the specified e-mail address.

If a failure occurs on a Windows host that does not have e-mail notification enabled, a message appears in the Windows Event Viewer. The message includes the *e-mail-address* value specified with this option and a note requesting that this user be informed of the status of the operation. For information about enabling e-mail notification, see the **MultiSite Control Panel** reference page.

Event Records and Comments

Default

–nc (no comment).

–c *comment* | **–cq** | **–cqe** | **–nc**

Specifies a comment to be placed in the shipping order. With **–c**, the comment string must be a single command-line token; typically, you must enclose it in double quotes. With **–cq** and **–cqe**, the command prompts you for a comment. With **–nc**, no comment is placed in the shipping order.

Specifying the Destination

Default

None.

destination ...

One or more host names (which must be usable by hosts in different domains) or IP addresses. When sending a MultiSite packet, you must specify the synchronization server for the replica.

Examples

In these examples, the lines are broken for readability. You must enter each command on a single physical line.

- Create a shipping order for file p1, which is located in the default storage bay. Store the shipping order in the same storage bay as p1, and specify that the file is to be sent to host **goldengate**.

  mkorder –data "c:\Program Files\Rational\ClearCase\var\shipping\cq_default\outgoing\p1" –sclass cq_default
  –out "c:\Program Files\Rational\ClearCase\var\shipping\cq_default\outgoing\p1_order" goldengate

  ```
  Shipping order "c:\Program Files\Rational\ClearCase\var
  \shipping\cq_default\outgoing\p1_order" generated.
  ```

- Create a shipping order in the default storage bay for a specified file that is to be delivered to host **goldengate**. Specify that **admin** must be notified if the file is not delivered successfully.

  /opt/rational/clearcase/etc/mkorder –data /usr/tmp/to_goldengate –sclass cq_default
  –ship –copy –notify admin goldengate

  ```
  Shipping order
  "/var/adm/rational/clearcase/shipping/cq_default/outgoing/sh_o_to_g
  oldengate" generated.
  ```

- Create a shipping order for the same file, but place it in the storage bay for a particular storage class. Attempt immediate delivery (**–fship**), and allow delivery attempts to continue until the beginning of May 18.

  mkorder –data c:\tmp\to_goldengate –fship –copy –sclass ClassA –pexpire 18-May goldengate

  ```
  Shipping order "c:\tmp\sclass\ClassA\sh_o_to_goldengate" generated.
  Attempting to forward/deliver generated packets...
  -- Forwarded/delivered packet
  c:\tmp\sclass\ClassA\sh_o_to_goldengate
  ```

Files

ccase-home-dir/config/services/shipping.conf

See Also

mkreplica, **MultiSite Control Panel**, **shipping.conf**, **shipping_server**, **syncreplica**
Chapter 10, *Troubleshooting MultiSite Operations*

mkreplica

Creates a replica

Applicability

Product	Command type
MultiSite	multiutil subcommand

Platform
UNIX
Windows

Synopsis

- Duplicate an existing database, generating a new replica object and a replica-creation packet:

  mkrep·lica –exp·ort [**–cl·an** *clan-name*] [**–site** *site-name*] **–fam·ily** *family-name*
  –u·ser *username* [**–p·assword**] *password* [**–max·size** *size*] [**–c·omments** *comments*]
  [**–size** *id-block-size*] [**–thres·hold** *id-block-threshold*]
  {
  { **–sh·ip** | **–fsh·ip** } **-wor·kdir** *temp-dir-pname* [**–sc·lass** *storage-class*]
  [**–pex·pire** *date-time*] [**–not·ify** *e-mail-addr*]
  | **–out** *packet-file-pname*
  } *hostname*:*site-name* ...

- Import a replica-creation packet to create a new user database replica and a new schema repository replica:

  mkrep·lica –imp·ort
  { **–site** *site-name* **–repo·sitory** *db-info* [**–vendor** *vendor-type*] *db-params*
  }
  { [**–data·base** *db-info* [**–vendor** *vendor-type*] *db-params*
  [**–c·omments** *comments*] { *packet-file-pname* | *packet-dir-path* }...

- Import a replica-creation packet to create a new replica in the same clan as the existing schema repository in the current site:

  mkrep·lica –imp·ort { [**–cl·an** *clan-name*] [**-site** *site-name*]
  –u·ser *username* [**–p·assword**] *password*
  { **–data·base** *db-info* [**–vendor** *vendor-type*] *db-params*
  [**–c·omments** *comments*] { *packet-file-pname* **|** *packet-dir-path* }...

Description

Note: Before replicating the first database of a clan, you must first activate the database set to which it belongs. You should also upgrade the databases you want to replicate to use the most recent version of the schema.

The **mkreplica –export** command may take a long time. The database and the schema repository are locked while an export is in progress. Make sure that all users are logged out before you run **mkreplica –export**. For more information, see Chapter 6, *Creating Database Replicas*.

The creation of a new replica is a three-phase process:

1. The **mkreplica –export** command duplicates the contents of the specified user database and its associated schema repository. This generates a single logical replica-creation packet for transmission to one or more other sites. A logical packet can be divided into multiple physical packets. (If you use **–fship** or **–ship**, **mkreplica** also generates a shipping order file for each physical packet.)

   Note: Creating multiple replicas in one **mkreplica –export** command is more efficient than using multiple **mkreplica –export** commands.

2. The packet is sent to one or more other sites.
3. At each receiving site, a **mkreplica –import** command uses the replica-creation packet to create a new replica. The new replica consists of two replicated databases, a schema repository and a user database. This command varies if you are adding a user database replica to a family within the same clan of an existing schema repository.

Creating Empty Vendor Databases

At each new site, the administrator must create empty vendor databases for the replica data. If this is the first replica in the new site, you need at least two empty vendor databases, one for the schema repository replica and one for the user database replica.

Note: If you are adding a new user database replica to an existing site, you don’t need to create a vendor database for the schema repository. You can associate the new user database replica with the existing schema repository in your site.

Oplog Information

When a database is replicated for the first time, the database's operation log (oplog) is enabled. All operations to be replicated are recorded in the oplog. Logging of operations continues until all replicas are deleted, leaving only the original database set. Note that creation of additional replicas is recorded in oplog entries. Existing replicas learn about a new replica through the standard synchronization mechanism. (See the **syncreplica** reference page).

Note: Before entering a **mkreplica –export** command, verify that MultiSite licenses are installed at the original site. After you activate the original database set, developers cannot access the database set without a MultiSite license (in addition to a ClearQuest license). A MultiSite license is also required to run **mkreplica –export**.

Allocating ID Blocks to a Replica

MultiSite controls how many record ID numbers are allocated to each replica. This allocation is done by using ID blocks (groups of IDs).

By default, each replica is given an ID block of 4096 IDs when it is created. When a replica reaches a threshold of 1024 IDs left to use, it is allocated another ID block of 4096 IDs to ensure that all IDs are unique. ID block allocation is handled internally by the working schema repository during synchronization.

Depending on the activity level of a replica family, it may be helpful to increase the size of the ID blocks that are allocated to a replica. For example, with the default settings, if you try to submit a large number of defects, the first 4096 are submitted successfully, but submissions after that fail.

To control how many IDs a replica is allocated, you can use the **–size** option combined with the **–threshold** option when you create a replica with the **mkreplica –export** command. You can modify these settings with the **chreplica** command.

Replica-Creation Packets

Each invocation of **mkreplica –export** creates a single logical replica-creation packet. (This is true even if you create several new replicas with one **mkreplica** command.) Each packet includes one or more replica specifications, each of which indicates the new replica's name and the synchronization server associated with the new replica.

The user database and schema repository are locked during the export phase.

The **–maxsize** option divides the single logical packet into multiple physical packets to conform to the limitations of the transfer medium.

Recovering from Failed Imports

If a replica import is interrupted or fails for any reason (a power outage, for example), you must delete the vendor databases, create new vendor database for the failed import operation, and rerun **mkreplica –import**.

It is possible to have a successful import of the schema repository, but a failed import of the user database replica. In this case, you must delete and re-create the vendor database that was intended for the user database replica. For more information, see *Recovering from a Failed Import* on page 60.

Cleaning Up Used Packets

Replica-creation packets are not deleted after import. After you import a replica-creation packet with **mkreplica –import**, you must delete the packet.

Error Handling for Packet Delivery Failures

If a packet cannot be delivered, it is sent through the store-and-forward facility back to the administrator at the site of the originating replica. A mail message is sent to the store-and-forward administrator. This occurs after repeated attempts to deliver the packet have failed and the allotted time has expired; it can also occur when the destination host is unknown or a data file does not exist. The store-and-forward configuration settings specify the expiration period, the e-mail address of the administrator, and the notification program.

Restrictions

Locks: This command fails if the database is locked (for example, during the upgrade process) or while another ClearQuest MultiSite operation is being performed.

Other: You cannot replicate a database to a host running a different version of MultiSite. You can run **mkreplica –export** at any site, but we recommend that you always run it at the working schema repository site to avoid the creation of multiple sites with the same name.

Options and Arguments — Export Phase

Specifying the Clan, Site, and Family

Default

Clan: First clan replicated at this site. If there is more than one clan at the site, **–clan** is required.
Site: Current site. If there is more than one site on this host, **–site** is required.
Family: No default; you must specify a family.

–cl·an *clan-name*

Name of the replica's clan.

–site *site-name*

Name of the replica's site.

–fam·ily *family-name*

User database family: Database name given to the user database when it was created.

Schema repository family: Not applicable. When you run **mkreplica**, the associated schema repository of the user database family you specify is included in the replica-creation packet.

Default: None.

Specifying a User Name and Password

Default

You must specify a user name and password.

–u·ser *user*

Name of a user with Super User privileges.

–p·assword *password*

Password associated with the specified user.

Specifying the Replica-Creation Packet Size

Default

When you do not specify **–maxsize**, the default packet size depends on the shipping method you use:

- Packets created with **–ship** or **–fship** are no larger than the maximum packet size specified in the MultiSite Control Panel.
- Packets created with **–out** are no larger than 2 GB.

The **mkreplica** command fails if it tries to create a packet larger than the size supported by your system.

–max·size *size*

The maximum size for a physical packet, expressed as a number followed by a single letter; for example:

500k	500 kilobytes
20m	20 megabytes
1.5g	1.5 gigabytes

Specifying a Comment

Default

None.

–c·omments *comments*

Comments you want to store with this replica's information.

Specifying ID Block Allocation

Default

ID block size: 4096. ID block threshold: 25 percent.

–size *id-block-size*

Size of ID block. You can enter any number from 1 to 1023. The value of *id-block-size* is multiplied by 100 to obtain the actual ID block size. For example, to specify an ID block of 30,000, use the number 300; to specify an ID block of 25,000, use the number 250.

–thres·hold *id-block-threshold*

The number of record ID numbers allocated to the replica. *id-block-threshold* is specified as a integer, representing a percentage. You can enter any number from 1 to 63. When the number of remaining record IDs to be used drops below the specified percentage of the current ID block size, an additional block is allocated.

Disposition of the Replica-Creation Packet

Default

None. You must specify how the replica-creation packet created by **mkreplica –export** is to be stored and/or transmitted to other sites.

–shi·p
–fsh·ip

Stores the replica-creation packet in one or more files in a store-and-forward storage bay. A separate shipping order file accompanies each physical packet, indicating how and where it is to be delivered.

–fship (force ship) invokes **shipping_server** to send the replica-creation packet. **–ship** places the packet in a storage bay. To send the packet, invoke **shipping_server**.

The disk partition where the storage bay is located (on the sending host and the receiving host) must have available space equal to or greater than the size of the replica-creation packet.

–wor·kdir *temp-dir-name*

A directory for use by **mkreplica** as a temporary workspace; it is deleted when **mkreplica** finishes. This directory must not already exist.

–sc·lass *storage-class*

Specifies the storage class of the packet and shipping order. **mkreplica** looks up the storage class in the MultiSite Control Panel (Windows) or the shipping.conf file (UNIX) to determine the location of the storage bay to use.

Default: mkreplica places the packet in the storage bay location specified for the **cq_default** class.

–out *packet-file-pname*

The name of the first physical replica-creation packet. Additional packets are placed in files named *packet-file-pname*_2, *packet-file-pname*_3, and so on.

The replica-creation packets are not delivered automatically; use an appropriate method to deliver them. You can create a packet using **–out**, and subsequently deliver it using the store-and-forward facility. See the **mkorder** reference page.

Handling Packet-Delivery Failures

Default

If a packet cannot be delivered, it is sent through the store-and-forward facility to the administrator at the site of the originating replica. A mail message is sent to the store-and-forward administrator. This occurs after repeated attempts to deliver the packet have all failed and the allotted time has expired; it can also occur when the destination host is unknown or a data file does not exist. The store-and-forward configuration settings specify the expiration period, the e-mail address of the administrator, and the notification program.

–pex·pire *date-time*

Specifies the time at which the store-and-forward facility stops trying to deliver the packet and generates a failure mail message instead. This option overrides the expiration period specified for the storage class in the shipping.conf file (UNIX) or MultiSite Control Panel (Windows).

The *date-time* argument can have any of the following formats:

*date***.***time* | *date* | *time* | **now**
where:

date := *day-of-week* | *long-date*

time	:=	*h*[*h*]**:***m*[*m*][**:***s*[*s*]] [**UTC** [[**+** | **-**]*h*[*h*][**:***m*[*m*]]]]
day-of-week	:=	**today** | **yesterday** | **Sunday** | ... | **Saturday** | **Sun** | ... | **Sat**
long-date	:=	*d*[*d*]**–***month*[**–**[*yy*]*yy*]
month	:=	**January** | ... | **December** | **Jan** | ... | **Dec**

Specify the *time* in 24-hour format, relative to the local time zone. If you omit the time, the default value is **00:00:00**. If you omit the *date*, the default value is **today**. If you omit the century, year, or a specific date, the most recent one is used. Specify **UTC** if you want the time to be resolved to the same moment in time regardless of time zone. Use the plus (+) or minus (-) operator to specify a positive or negative offset to the UTC time. If you specify **UTC** without hour or minute offsets, the default setting is Greenwich Mean Time (GMT). (Dates before January 1, 1970 Universal Coordinated Time (UTC) are invalid.)

Examples:

```
22-November-2002
sunday
yesterday.16:00
8-jun
13:00
today
9-Aug.10:00UTC
```

–not·ify *e-mail-address*

The delivery-failure message is sent to the specified e-mail address.

If a failure occurs on a Windows host that does not have e-mail notification enabled, a message appears in the Windows Event Viewer. The message includes the *e-mail-address* value specified with this option and a note requesting that this user be informed of the status of the operation. For information about enabling e-mail notification, see the **MultiSite Control Panel** reference page.

Replica Specifications

Default

None.

hostname:site-name...

One or more arguments, each of which indicates one new replica to be created from this packet in another site.

hostname	The synchronization server for the new replica. *hostname* must be usable by hosts in different domains. It is used by the store-and-forward mechanism to determine how to route update packets to the replica. However, keep this information accurate even if your site does not use store-and-forward. (See the **chreplica** reference page.) *hostname* can be either the IP address of the host, or the computer name, for example, **minuteman**. You may have to append an IP domain name, for example, **minuteman.purpledoc.com**. On UNIX, use the **uname –n** command to display the computer name. On Windows NT, the computer name is displayed in the Network Settings dialog box, which is accessible from the **Network** icon in the Control Panel. On Windows 2000, the computer name is displayed on the **Network Identification** tab in the System Properties dialog box, which is accessible from the **System** icon in the Control Panel.
site-name	Name by which the replica will be identified in **multiutil** commands. The site name must be an identifier and can be up to 50 characters long. This name must be unique within the respective clan: there cannot be two sites with the same name participating in the same clan.

Options and Arguments — Import Phase for Schema Repository and User Database Import

Specifying the Site and Database Information

Default

None.

–site *site-name*

The name of the site where the replica will be imported. The site name was given to the replica when it was exported. If you don't know the site name, contact the administrator at the exporting site.

–repo·sitory *db-info*

The database information for the vendor database you are using.

Vendor Database	dbinfo Value
DB2	*Database Alias*
Oracle	*SQL*Net Alias*
SQL Server	*Physical Database Name*

–vendor *vendor-type*

The database vendor you are using. Supported vendor types are DB2, ORACLE, and SQL_SERVER.

db-params

The required database parameters are the same parameters needed to connect to any ClearQuest database. Note these when you create the vendor database to which you import the replica. For more information about how to create empty vendor databases and the parameters you will need, see the *Installation Guide* for Rational ClearQuest.

When you import a replica, you must specify the database parameters of the vendor database for the schema repository replica and the vendor database for the user database replica. You must create these databases before importing a replica packet.

Vendor database	db-params value
DB2	**–dbologin** *dbo-name* [*dbo-pwd*]
Oracle	**–dbologin** *dbo-name dbo-pwd* [**–connectopts** *connect-options*]
SQL Server	**–server** *server-name* **–dbologin** *dbo-name* [*dbo-pwd*] **–rwlogin** *rw-login* [*rw-pwd*] [**–rologin** *ro-login* [*ro-pwd*]]

Note: Use **–rologin** only when creating a schema repository replica.

–data·base *db-info*

The user database information for the vendor database you are using.

Vendor database	dbinfo value
DB2	*Database Alias*
Oracle	*SQL*Net Alias*
SQL Server	*Physical Database Name*

–c·omments *comments*

Comments you want to store with the replica's information.

Specifying the Location of the Replica-Creation Packet

Default

None.

packet-file-pname | *packet-dir-path ...*

Specifies a pathname of a replica-creation packet. For a logical packet that spans multiple disk files, **mkreplica** scans the directory containing *packet-file-pname* for related physical packets.

If you also specify one or more *packet-dir-path* arguments, **mkreplica** searches for additional packets in these directories.

Options and Arguments — Import Phase for User Database Import Only

If you are adding a user database family to an existing clan, you need to create a vendor database for the user database replica only.

Specifying the Clan and Site

Default

Clan: First clan replicated at this site. If there is more than one clan at the site, **–clan** is required.
Site: Current site. If there is more than one site on this host, **–site** is required.

–cl·an *clan-name*

Name of the replica's clan.

–site *site-name*

Name of the replica's site.

Specifying a User Name and Password

Default

You must specify a user name and password.

–u·ser *user*

Name of a user with Super User privileges.

–p·assword *password*

Password associated with the specified user.

Specifying the Database Information

–data·base *db-info*

The user database information for the vendor database you are using.

Vendor database	dbinfo value
DB2	*Database Alias*
Oracle	*SQL*Net Alias*
SQL Server	*Physical Database Name*

–vendor *vendor-type*

Enter the database vendor you are using. Supported vendor types are DB2, ORACLE, and SQL_SERVER.

db-params

The required database parameters are the same parameters needed to connect to any ClearQuest database. Note these when you create the vendor database to which you import the replica. For more information about how to create empty vendor databases and the parameters you will need, see the *Installation Guide* for Rational ClearQuest.

When you import a replica, you must specify the database parameters of the vendor database for the schema repository replica and the vendor database for the user database replica. You must create these databases before importing a replica packet.

Vendor database	db-params value
DB2	**–dbologin** *dbo-name* [*dbo-pwd*]
Oracle	**–dbologin** *dbo-name* [*dbo-pwd*] [**–connectopts** *connect-options*]
SQL Server	**–server** *server-name* **–dbologin** *dbo-name dbo-pwd* **–rwlogin** *rw-login* [*rw-pwd*] [**–rologin** *ro-login* [*ro-pwd*]]

Note: Use **–rologin** only when creating a schema repository replica.

–c·omments *comments*

Comments you want to store with this replica's information. These comments are stored in the schema repository database at the importing site and are displayed in the Database Property dialog box in the ClearQuest Designer.

Specifying the Location of the Replica-Creation Packet

packet-file-pname **|** *packet-dir-path ...*

Specifies a pathname of a replica-creation packet. For a logical packet that spans multiple disk files, **mkreplica** scans the directory containing *packet-file-pname* for related physical packets.

If you also specify one or more *packet-dir-path* arguments, **mkreplica** searches for additional packets in these directories.

Default: None.

Examples

In these examples, the lines are broken for readability. You must enter each command on a single physical line.

Exports

- At the **boston_hub** replica, generate a replica-creation packet for the **DEV** family to create a new replica named **sanfran_hub**. The synchronization server for the new replica is **goldengate**.

  multiutil mkreplica -export -clan telecomm -site boston_hub -family DEV -u susan -p passwd -out c:\cqms\boston_hub.xml goldengate:sanfran_hub

  ```
  Multiutil: Packet file 'c:\cqms\boston_hub.xml' generated
  ```

- At the **boston_hub** replica, generate a packet that will create a replica of the **LAB** family database when imported at the **sanfran_hub** replica.

  multiutil mkreplica -export -clan telecomm -site boston_hub -family LAB -user susan -p passwd -out c:\cqms\lab.xml goldengate:sanfran_hub

  ```
  Multiutil: Packet file 'c:\cqms\lab.xml' generated
  ```

- At the **tokyo** replica, generate a replica-creation packet for the **sydney** replica and use **–fship** to forward the packet immediately.

  multiutil mkreplica -export -clan testing -site tokyo -family TEST -user masako -p passwd -fship -workdir c:\cqms\working -sclass cq_default taronga:sydney

  ```
  Multiutil: Packet file
  'c:\cqms\working\mk_TOKYO_29-January-02_09-47-27.xml' generated

  multiutil: Shipping order
  "C:\temp\cqms\ms_ship\outgoing\sh_o_mk_TOKYO_29-January-02_09-47-27
  .xml" generated.

  multiutil: Attempting to forward/deliver generated packets...

  multiutil:   -- Forwarded/delivered packet
  C:\temp\cqms\ms_ship\outgoing\mk_TOKYO_29-January-02_09-4
  ```

- Similar to the preceding example, but place the packet file in a storage bay for shipping at some later time by the store-and-forward facility.

  multiutil mkreplia -export -clan telecomm -site boston_hub -family DEV -user susan -password passwd -c "make a new replica for sanfran_hub" -ship -workdir c:\temp\working -sclass cq_default -pexpire 22-November-2003 goldengate:sanfran_hub

Imports

- Import a new database replica **sanfran_hub** and its associated schema repository replica into SQL Server databases.

  multiutil mkreplica -import -site sanfran_hub -repository sanfran_schemarepo -vendor SQL_SERVER -server sb_server -dbologin jcole passwd -rwlogin jcole passwd -rologin jcole passwd -database sanfran_userdb -vendor SQL_SERVER -dbologin jcole passwd -rwlogin jcole passwd

- Import a new user database replica that is part of the **sydney** site in the **testing** clan. The new user database replica is being imported into a SQL Server database.

  multiutil mkreplica -import -clan testing -site sydney -user bfife -p passwd -database syd_userdb -vendor SQL_SERVER -dbologin bfife passwd -rwlogin bfife passwd

See Also

activate

MultiSite Control Panel

Configures store-and-forward facility

Applicability

Product	Command type
MultiSite	Administrative tool

Platform
Windows

Synopsis

%SystemRoot%\System32\ms.cpl

To open the MultiSite Control Panel, double-click the **MultiSite** icon in Control Panel.

Description

The MultiSite Control Panel controls operation of the store-and-forward facility on each host. It provides controls for setting the configuration parameters described in the following sections. In some cases, the corresponding operation fails if a parameter is not defined, and in other cases there is a hard-coded default.

Maximum Packet Size

Default: 2097151 KB

Controls the splitting of logical packets into multiple physical packets. This value specifies the maximum size for a physical packet file. Limiting the size of physical packets can improve the reliability of packet delivery in some networks. To specify no limit, use **0** (zero).

This value is used by the following commands (unless you also specify **–maxsize**):

- **mkreplica –fship**
- **mkreplica –ship**
- **syncreplica –fship**
- **syncreplica –ship**

When you invoke **mkreplica** or **syncreplica** with **–out**, this value is not used, and you must use **–maxsize** to limit the packet size.

Administrator E-mail

Default: None.

Specifies the electronic mail address of the user to be notified when any of these events occur:

- A packet (on the local host) that has expired is returned to its sending host.
- A packet that was not delivered to its next hop is returned to its sending host.
- **syncreplica –import** finds a replica-creation packet.

Note: If you use ClearQuest MultiSite and you do not use ClearCase MultiSite, use the **control_panel** command to specify the SMTP host and the administrator e-mail.

To enable e-mail notification:

1. Specify the SMTP host to use:
    - If you use ClearCase MultiSite, verify that the **SMTP Host** box in the ClearCase Control Panel specifies a valid host. (This box is located on the **Options** tab.)
    - If you do not use ClearCase MultiSite, use the **control_panel** command to specify the SMTP host.
2. Enter an e-mail address in the **Administrator Email** box in the MultiSite Control Panel. You can specify only one address.
3. (Optional) Enter a different value in the **Email Notification Program Path** field.

Email Notification Program Path

Default: *ccase-home-dir*\bin\notify.exe

Specifies the electronic mail program to be invoked in the circumstances listed in *Administrator E-mail*.

Timeout for Unreachable Host (minutes)

Default: None.

Specifies the number of minutes for the shipping server to wait before trying to contact a target host that was previously identified as unreachable.

If the shipping server tries to send a packet to a target host and determines that the host is unreachable, it creates a file in the *ccase-home-dir*\var\shipping\ms_downhost directory. The name of the file is the name of the unreachable host. If the **Timeout for Unreachable**

Host setting is set, the shipping server checks the directory for target hosts during future shipping operations.

If the target host is found in the ms_downhost directory, and the difference between the current time and the last modification time of the file is less than the timeout setting on the shipping server host, the shipping server does not try to send packets to the target host. If the difference is equal to or greater than the timeout setting, the shipping server tries to send packets to the target host. If the **Timeout for Unreachable Host** setting is not set, the shipping server attempts to send the packet to the target host. (Each attempt to send a packet to an unreachable host takes about 30 seconds.)

Storage Classes

Storage Class Name

Default: **multiutil** commands that use the **–sclass** option use the **cq_default** storage class for packets that are not assigned to any storage class, and for packets whose storage class is not configured. The **cq_default** storage class is not created when MultiSite is installed. The **mkorder** and **shipping_server** commands use the **–default** storage class for packets that are not assigned to any storage class and for packets whose storage class is not configured. You can create additional storage classes for ClearQuest MultiSite packets, but you must use different storage classes for ClearQuest MultiSite packets and ClearCase MultiSite packets.

Specifies the name of a storage class. For each storage class, you can specify values for packet expiration, the storage bay, the return bay, and the receipt handler.

Note: Storage class names are case sensitive.

Packet Expiration

Default: When the **Use Default Expiration** check box is selected, the storage class uses the packet expiration value associated with the **–default** class. (This value is not shown in the **Packet Expiration** box; you must display the **–default** class to determine the value.) When MultiSite is installed for the first time, the Packet Expiration value for the **–default** class is set to 14 days.

Specifies the expiration period (in days) for shipping orders associated with the specified storage class. This period begins at the time the shipping order is generated. If a packet cannot be delivered to all its destinations in the specified number of days, the packet is returned to the original sending host and a message is sent to the address specified in the **Administrator Email** box. If e-mail notification is not enabled, a message is written to the Windows Event Viewer.

A value of **0** (zero) specifies no expiration and delivery is reattempted indefinitely.

This setting is overridden by the **–pexpire** option to **syncreplica** or **mkreplica**.

The **shipping_server** program does not retry delivery of packets. The Packet Expiration specification is useful only if you set up a host to periodically attempt delivery of any undelivered packets.

Storage Bay Path

Default: **multiutil** commands that use the **–sclass** option use the **cq_default** storage class. The **mkorder** and **shipping_server** commands use the **–default** storage class. You must create the **cq_default** storage class. If you use both ClearCase MultiSite and ClearQuest MultiSite, you must use different storage bays for VOB replica packets and database packets.

Defines the location of the directory that holds the outgoing and incoming update packets and shipping orders of a particular storage class.

Packets placed in a storage bay on an NTFS file system inherit the Windows ACL on the bay. Define ACLs on the storage bays to enable successful execution of MultiSite commands to process the packets and to guard against unauthorized access. Packets stored on FAT file systems have no protections.

Before using the store-and-forward facility, verify that the disk partition where the *ccase-home-dir*\var\shipping directory is created has sufficient free space for anticipated replica-creation and update packets. For more information, see *MultiSite Installation* on page 15.

Note: When you create a new storage class, the storage bay and return bay that you specify are created. The incoming and outgoing directories in the bays are also created.

Return Bay Path

Default: **multiutil** commands that use the **–sclass** option use the **cq_default** storage class. The **mkorder** and **shipping_server** commands use the **–default** storage class. You must create the **cq_default** storage class. If you use both ClearCase MultiSite and ClearQuest MultiSite, you must use different return bays for VOB replica packets and database packets.

Defines the location of the directory that holds the incoming and outgoing packets in the process of being returned to their origin because they could not be delivered to all specified destinations.

Packets placed in a return bay on an NTFS file system inherit the Windows ACL on the bay. Define ACLs on the return bays to enable successful execution of MultiSite commands to process the packets and to guard against unauthorized access. Packets stored on FAT file systems have no protections.

Receipt Handler Path

Default: None.

Specifies a batch file or program for the shipping server to run when a packet is received for the storage class. By default, no file is specified.

For each packet that is received, **shipping_server** does the following:

1. Reads the entries in the MultiSite Control Panel to find the appropriate **Receipt Handler** value for the packet.
   - If the packet is associated with a storage class and there is a **Receipt Handler** value for that storage class, **shipping_server** uses the specified batch file or program
   - If the packet is not associated with a storage class and there is a **Receipt Handler** value for the **–default** storage class, **shipping_server** uses that value
2. Invokes the receipt handler, as follows:

   script-pname [**–d·ata** *packet-file-pname*] [**–a·ctual** *shipping-order-pname*]
   [**–s·class** *storage-class*] **–o·rigin** *hostname*

   where

script-pname	Script specified in the **RECEIPT-HANDLER** entry.
–d·ata *packet-file-pname*	Location of the packet. This parameter is used only when the packet is destined for this host.
–a·ctual *shipping-order-pname*	Location of the shipping order. This parameter is used only when the packet is destined for another host.
–s·class *storage-class*	Storage class associated with the packet. This parameter is used only if the packet was associated with a storage class when it was created.
–o·rigin *hostname*	Name of the host from which the packet was first sent.

   Note: If a packet is destined for both the local host and another host, both the **–data** and **–actual** parameters are used. The packet is imported at the replica on the host, and forwarded to its next destination.

Routing Information

The **Routing Information** settings control the network routing of packets.

Next Routing Hop

Default: None.

Specifies the next destination for packets whose final destination is any of the host names specified in the **Destination Hostnames** list. This host is responsible for delivery of the packet to its destinations. You can specify a host using either its host name (which must be usable by hosts in different domains) or its numeric IP address.

Destination Host Names

Default: None.

Packets destined for any host listed in this field are sent to the host specified in the **Next Routing Hop** box. You can specify a host using either its host name (which must be usable by hosts in different domains) or its numeric IP address. The value **–default** as the **Destination Hostname** accommodates all hosts that are not associated with a routing hop.

multiutil

MultiSite user-level commands

Applicability

Product	Command type
MultiSite	MultiSite command

Platform
UNIX
Windows

Synopsis

- Single-command mode:

  multiutil *subcommand* [*options/args*]

- Interactive mode:

  multiutil

  `multiutil>` *subcommand* `[` *options/args* `]`

  `. . .`

  `multiutil>` **quit**

Description

multiutil is the principal program in MultiSite. The different **multiutil** subcommands are described in *Descriptions of Subcommands* on page 36.

When entered without an option, **multiutil** enters interactive mode. It exits if an error is returned by a command.

If you use the ClearCase/ClearQuest UCM integration, you must run **multiutil** from a machine that does not require the integration. **multiutil** requires special database set names that are not supported by the UCM integration.

recoverpacket

Resets epoch number matrix so that changes in lost packets are resent

Applicability

Product	Command type
MultiSite	multiutil subcommand

Platform
UNIX
Windows

Synopsis

recoverpacket [**–cl·an** *clan-name*] [**–site** *site-name*] **–fam·ily** *family-name*
–u·ser *username* [**–p·assword**] *password*
[**–sin·ce** *date-time*] *replica* ...

Description

The **recoverpacket** command resets the epoch row at a sending replica to reflect the last synchronization sent to a replica before a particular time. It scans through a list of epoch rows saved at the time of each export, looking for an entry prior to the time specified. When it finds an entry, it uses the associated row to reset the epoch row for the specified receiving replica. The next packet that is exported includes the changes that were in the lost packet.

Resetting Epoch Numbers Automatically

When you send an update packet to another replica, success of the transport and import phases is assumed. Therefore, the sending replica's epoch number matrix is updated to reflect that the changes are made at the receiving replica. However, if the packet is lost before reaching the receiving replica, the sending replica's assumption that the receiving replica is up to date is incorrect.

The epoch numbers at the sending replica must be returned to the values they had before the packet was sent. Making these corrections to the sending replica's epoch number matrix causes it to include the same changes in the next update packet it sends to the receiving replica.

The administrator at the receiving replica must run a **dumpoplog** command to determine the time of the last successful import. The administrator at the sending replica uses this time in the **recoverpacket** command.

Note: If the two replicas are not in the same time zone or you do not send packets at the same time you generate them (for example, you generate packets at midnight and send them at 6:00 A.M.), you must adjust for the time difference.

Resetting Epoch Numbers Manually

If there are no saved epoch rows that are as old as the specified time, the **recoverpacket** command fails. In this case, the administrator at the receiving replica must use the **lsepoch** command to determine the correct epoch number, and the administrator at the sending replica must run **chepoch** on the sending replica to reset the epoch row. See the **chepoch** reference page.

Restrictions

You must have Super User privileges.

Options and Arguments

Specifying the Clan, Site, and Family

Default

Clan: First clan replicated at this site. If there is more than one clan at the site, **–clan** is required.
Site: Current site. If there is more than one site on this host, **–site** is required.
Family: No default; you must specify a family.

–cl·an *clan-name*

Name of the replica's clan.

–site *site-name*

Name of the replica's site.

–fam·ily *family-name*

User database family: Database name given to the user database when it was created.

Schema repository family: Use the **MASTR** family if you want to recover information about the working schema repository. If you've lost a packet, run **recoverpacket** on both the **MASTR** and user database families then run **syncreplica** again.

Specifying a User Name and Password

Default

You must specify a user name and password.

–u·ser *user*

Name of a user with Super User privileges.

–p·assword *password*

Password associated with the specified user.

Specifying the Time

Default

If the time is not specified, **recoverpacket** uses the current time (and, therefore, resets the epoch row so that the changes in the most recent update packet are resent).

–since *date-time*

Specifies the time of the last successful processing of a packet at the receiving replica. The *date-time* argument can have any of the following formats:

*date***.***time* | *date* | *time*
where:

date	:=	*day-of-week* | *long-date*
time	:=	*h*[*h*]**:***m*[*m*][**:***s*[*s*]] [**UTC** [[**+** | **-**]*h*[*h*][**:***m*[*m*]]]]
day-of-week	:=	**today** | **yesterday** | **Sunday** | ... | **Saturday** | **Sun** | ... | **Sat**
long-date	:=	*d*[*d*]**–***month*[**–**[*yy*]*yy*]
month	:=	**January** | ... | **December** | **Jan** | ... | **Dec**

Specify the *time* in 24-hour format, relative to the local time zone. If you omit the time, the default value is **00:00:00**. If you omit the *date*, the default value is **today**. If you omit the century, year, or a specific date, the most recent one is used. Specify **UTC** if you want the time to be resolved to the same moment in time regardless of time zone. Use the plus (+) or minus (-) operator to specify a positive or negative offset to the UTC time. If you specify **UTC** without hour or minute offsets, the default setting is Greenwich Mean Time (GMT). (Dates before January 1, 1970 Universal Coordinated Time (UTC) are invalid.)

Examples:

```
22-November-2002
sunday
yesterday.16:00
8-jun
13:00
today
9-Aug.10:00UTC
```

replica ...

Site name of the replica for which the epoch row is reset.

Examples

In these examples, the lines are broken for readability. You must enter each command on a single physical line.

- At the **boston_hub** replica, reset the epoch row for the **sanfran_hub** replica so that changes sent since January 22, 2002 are included in the next update packet.

  multiutil recoverpacket -clan telecomm -site boston_hub -family DEV -user susan -p passwd -since 22-January-2002 sanfran_hub

  ```
  Multiutil: Using epoch information from 22-Jan-2002.10:06:52.

  Multiutil: Epoch estimates for replica 'sanfran_hub' successfully
  reset.

  SANFRAN_HUB: 3
  ```

See Also

chepoch, **lsepoch**, **restorereplica**

restorereplica

Replaces missing operations in a replica that has been restored from backup

Applicability

Product	Command type
MultiSite	multiutil subcommand

Platform
UNIX
Windows

Synopsis

restorereplica [**–cl·an** *clan-name*] [**–site** *site-name*] **–fam·ily** *family-name*
–u·ser *username* [**–p·assword**] *password* [**–force**] [**–completed**]
[**–replace**] [*replica...*]

Description

Warning: Execute this command immediately after you restore a replica from backup. Proceeding with normal development at a restored replica before executing this command causes irreparable inconsistencies among the replicas in a family.

restorereplica replaces missing changes in a replica that has been restored from backup, as follows:

1. It causes the current replica to create special update packets that contain update requests to other replicas.
2. It locks the current replica and marks the replica as being in the process of restoration.
3. It causes **lsreplica –long** to indicate which replicas must send restoration updates to the current replica.

The current replica remains in the restoration state until you have received and applied (using **syncreplica –import**) all the restoration updates needed to bring the replica up to date with the state of the family. Collectively, these updates include all the changes to the family since the backup was made, including changes made in the current replica before its failure.

You cannot recover changes that were made after the last synchronization export from your current replica. For example, if your replica was backed up on Wednesday at 12:30 P.M. and your last synchronization export was Thursday at 3:00 P.M., you can recover all changes made until Thursday at 3:00 P.M. All changes made after that time are lost.

For a description of the replica restoration procedure, see *Restoring Database Replicas* on page 102.

Locking the Replica

restorereplica locks the current replica. Locking it ensures that while restoration proceeds through execution of **syncreplica –export** and **syncreplica –import** commands, no other changes are made to the current replica.

When **syncreplica** applies the final required update, it displays a message indicating that the restoration process is complete and unlocks the replica.

Optimizing the Restoration Process

By default, **restorereplica** requires that the replica receive restoration updates from all other replicas in its family (either directly or indirectly). Only after all the updates are imported does the **syncreplica** command display the message indicating that restoration is complete.

In some cases, you can relax this requirement without compromising the correctness of the restoration process. The replica will be brought up to date if it receives a restoration update from only one replica—the last one to which the replica sent an update before it was restored from the backup version. You can specify the name of that last-updated replica (or a list of replicas, one of which must be the last-updated one) to **restorereplica**. **syncreplica** displays the restoration-completed message after receiving restoration updates from all the specified replicas.

Warning: If you use this optimization incorrectly, you can make the restored replica irreparably inconsistent with other replicas.

Restrictions

You must have Super User privileges.

Options and Arguments

Specifying the Clan, Site, and Family

Default

Clan: First clan replicated at this site. If there is more than one clan at the site, **–clan** is required.
Site: Current site. If there is more than one site on this host, **–site** is required.
Family: No default; you must specify a family.

–cl·an *clan-name*

Name of the replica's clan.

–site *site-name*

Name of the replica's site.

–fam·ily *family-name*

User database family: Database name given to the user database when it was created.

Schema repository family: Not applicable. Restoring a member of a user database family automatically requests updates for its associated schema repository replica, if necessary.

Specifying a User Name and Password

Default

You must specify a user name and password.

–u·ser *user*

Name of a user with Super User privileges.

–p·assword *password*

Password associated with the specified user.

Suppressing Interactive Prompts

Default

restorereplica prompts you for confirmation.

–f·orce

Suppresses the confirmation step.

Reducing the Number of Required Updates

Default

The replica requires restoration updates from all other members of its family. The **syncreplica** command declares the replica to be restored completely only after all the updates have been processed.

Warning: Incorrect use of these options allows new changes to be made to the replica before all missing changes are received from other replicas. This may place the entire family in an irreparably inconsistent state.

–completed
Overrides normal restoration processing: marks the replica as restored and unlocks the database. If this option is used, no more restoration packet requests can be sent and no more restoration packets can be replayed at this replica.

–rep·lace *replica*...
Changes the subset of replicas from which restoration updates are required. Specify *replica* as a site name.

Examples

For an example of restoring a replica, see *Restoring Database Replicas* on page 102.

See Also

chepoch, **lsepoch**, **lsreplica**, **syncreplica**

rmreplica

Deletes a replica

Applicability

Product	**Command type**
MultiSite	multiutil subcommand

Platform
UNIX
Windows

Synopsis

rmreplica [**–cl**·**an** *clan-name*] [**–site** *site-name*] **–fam**·**ily** *family-name*
–u·**ser** *username* [**–p**·**assword**] *password*
[**–dbset** *new-name*] *replica*

Description

Warning: To delete a replica, you must complete all steps described in *Deleting a Replica* on page 73. If you do not complete all steps in the correct order, synchronization and mastership problems can occur in other replicas in the database family.

This command deletes from the current replica's database the database-replica record that records the existence and identity of another replica. Typically, you use this command to record the fact that another replica has been decommissioned and deleted.

Restrictions

Identities: You must have Super User privileges.

Other: You must run the **rmreplica** command at the working schema repository site.

Options and Arguments

Specifying the Clan, Site, and Family

Default

Clan: First clan replicated at this site. If there is more than one clan at the site, **–clan** is required.
Site: Current site. If there is more than one site on this host, **–site** is required.
Family: No default; you must specify a family.

–cl·an *clan-name*

Name of the replica's clan.

–site *site-name*

Name of the replica's site.

–fam·ily *family-name*

User database family: Database name given to the user database when it was created.

Schema repository family: Not applicable. If there is only one user database family in the specified site, this command removes the schema repository as well. If there is more than one user database family, the schema repository is not removed.

Default: None.

Renaming the Database Set

Default

None.

–dbset *new-name*

This option is used only when removing the last replica of a clan. When you remove the last replica of a clan, you must rename the database set so it does not include any ClearQuest MultiSite flags.

Specifying the Replica

Default

When you run the command at the location of the replica to be removed, the default is the current replica. When you run the command at any other location, you must specify a replica.

replica

Site name of the replica to be deleted from the current replica's database.

Examples

For an example of using the **rmreplica** command, see *Deleting a Replica* on page 73.

See Also

chmaster, **mkreplica**

scruboplog

Deletes oplog entries for a replica

Applicability

Product	Command type
MultiSite	multiutil subcommand

Platform
UNIX
Windows

Synopsis

scruboplog [**–cl·an** *clan-name*] [**–site** *site-name*] **–fam·ily** *family-name*
[**–u·ser** *username*] [**–p·assword**] *password*
–before { *date-time* | *oplog-ID* }

Description

Operation log (oplog) entries must be kept in the replica for a significant period of time. They are used when the replica generates update packets to send to all other replicas. Oplog entries also may be required to help other replicas recover from failures.

However, you may want to delete (scrub) oplog entries occasionally to optimize hard drive space where the replica resides. You can also use the **scruboplog** command to delete the oplog of a replica that will no longer be used.

Although oplog entries record only the changes that have taken place in your replica, over time this information could require as much space as the data itself.

Before scrubbing oplog entries for a replica, you must be sure that they are no longer needed and that the other replicas in the family have the information that you want to delete from the replica's oplog. Also, you must synchronize the replicas in a family before scrubbing oplogs.

Restrictions

You must have Super User privileges.

Options and Arguments

Specifying the Clan, Site, and Family

Default

Clan: First clan replicated at this site. If there is more than one clan at the site, **–clan** is required.
Site: Current site. If there is more than one site on this host, **–site** is required.
Family: No default; you must specify a family.

–cl·an *clan-name*

Name of the replica’s clan.

–site *site-name*

Name of the replica’s site.

–fam·ily *family-name*

User database family: Database name given to the user database when it was created.

Schema repository family: The family name is **MASTR**.

Specifying a User Name and Password

Default

You must specify a user name and password.

–u·ser *user*

Name of a user with Super User privileges.

–p·assword *password*

Password associated with the specified user.

Specifying the Entries to Delete

Default

You must specify the entries to be deleted.

–before { *date-time* | *oplog-ID* }

Deletes all oplog entries before the specified *date-time* or *oplog-ID*. The argument is not inclusive: oplog entries created on the specified date or at the specified time are not deleted, and the oplog entry with the specified ID is not deleted.

The *oplog-ID* argument must be an integer. To display a replica’s operation log, use **dumpoplog**. The *date-time* argument can have any of the following formats:

*date***.***time* | *date* | *time*
where:

date	:=	*day-of-week* | *long-date*
time	:=	*h*[*h*]**:***m*[*m*][**:***s*[*s*]] [**UTC** [[**+** | **-**]*h*[*h*][**:***m*[*m*]]]]
day-of-week	:=	**today** | **yesterday** | **Sunday** | ... | **Saturday** | **Sun** | ... | **Sat**
long-date	:=	*d*[*d*]**–***month*[**–**[*yy*]*yy*]
month	:=	**January** | ... | **December** | **Jan** | ... | **Dec**

Specify the *time* in 24-hour format, relative to the local time zone. If you omit the time, the default value is **00:00:00**. If you omit the *date*, the default value is **today**. If you omit the century, year, or a specific date, the most recent one is used. Specify **UTC** if you want the time to be resolved to the same moment in time regardless of time zone. Use the plus (+) or minus (-) operator to specify a positive or negative offset to the UTC time. If you specify **UTC** without hour or minute offsets, the default setting is Greenwich Mean Time (GMT). (Dates before January 1, 1970 Universal Coordinated Time (UTC) are invalid.)

Examples:

```
22-November-2002
sunday
yesterday.16:00
8-jun
13:00
today
9-Aug.10:00UTC
```

Examples

In these examples, the lines are broken for readability. You must enter each command on a single physical line.

- Delete oplog entries before January 21, 2002 for the **DEV** family of the **boston_hub** replica.

  multiutil scruboplog -clan telecomm -site boston_hub -family DEV -user susan -p passwd -before 21-January-2002

- Delete oplog entries 1 through 300 for the **TEST** family of the **tokyo** replica

  multiutil scruboplog -clan testing -site tokyo -family TEST -user masako -p passwd -before 301

See Also

dumpoplog, **syncreplica**

shipping.conf

Store-and-forward configuration file

Applicability

Product	Command type
MultiSite	MultiSite data structure

Platform
UNIX

Synopsis

/var/adm/rational/clearcase/config/shipping.conf

Description

This file controls the operation of the store-and-forward facility on each host. The file consists of comment lines (starting with #) and one or more configuration entries, and it can contain the configuration entries described below. In some cases, the corresponding store-and-forward operation fails if an entry is missing; in other cases, there is a hard-coded default.

MultiSite installation creates the file *ccase-home-dir*/config/services/shipping.conf.template, in which all these entries are defined. If /var/adm/rational/clearcase/config/shipping.conf does not exist, the installation creates it by copying the template file. If /var/adm/rational/clearcase/config/shipping.conf exists, the installation advises you to compare the existing file to the template and make any necessary changes.

Note: If you do not install MultiSite or the Rational Shipping Server in the default installation directory (/opt/rational/clearcase), you must edit the shipping.conf file and change /opt/rational/clearcase to the pathname of your installation directory.

Packet Size

MAX-DATA-SIZE *size* [**k** | **m** | **g**]

Default: 2097151 KB

Controls the splitting of individual logical packets into multiple physical packets. Limiting the size of physical packets can improve the reliability of packet delivery in some networks. The *size* integer (with the optional **k**, **m**, or **g** suffix) specifies the maximum size for a physical packet file. **k** specifies KB (kilobytes); **m** specifies MB (megabytes); **g** specifies GB (gigabytes). Omitting the suffix specifies KB. To specify no limit, use **0** (zero).

This value is used by the following commands (unless you also specify **–maxsize**):

- **mkreplica –fship**
- **mkreplica –ship**
- **syncreplica –fship**
- **syncreplica –ship**

When you invoke **mkreplica** or **syncreplica** with **–out**, this value is not used and you must use **–maxsize** to limit the packet size.

Notification

NOTIFICATION-PROGRAM *e-mail-program-pathname*

Default: /opt/rational/clearcase/bin/notify. This program is also used if no **NOTIFICATION-PROGRAM** entry exists.

The electronic mail program to be invoked in these circumstances:

- When **shipping_server** finds that a shipping order it is about to process has expired
- When an undeliverable packet is returned to the original sending host by another host's **shipping_server** (see the description of **EXPIRATION**)
- When **syncreplica –import** finds a replica-creation packet, which must be processed with a **mkreplica** command

The mail program is invoked as follows:

e-mail-program-pathname **–s** *subject* **–f** *message-file addr* . . .

Administrator Address

ADMINISTRATOR *e-mail-address*

Default: **root**

The electronic mail address of the administrator who administers the store-and-forward facility on the local host.

A mail message is sent to the specified address in the circumstances listed in *Notification*. The configuration file can contain multiple **ADMINISTRATOR** entries; messages are sent to all the specified mail addresses.

Storage Bay and Return Bay

STORAGE-BAY *storage-class directory-pathname*
RETURN-BAY *storage-class directory-pathname*

Default: **multiutil** commands that use the **–sclass** option use the **cq_default** storage class for packets that are not assigned to any storage class, and for packets whose storage class is not configured. The **mkorder** and **shipping_server** commands use the **–default** storage class for packets that are not assigned to any storage class, and for packets whose storage class is not configured.

These lines define storage bay and return bay directories. A storage bay holds the outgoing and incoming update packets and shipping orders for a storage class. A return bay holds incoming or outgoing packets in the process of being returned to their origin because they could not be delivered to all specified destinations.

You can use multiple **STORAGE-BAY** and **RETURN-BAY** entries to define multiple bays for a storage class. **shipping_server** selects one of the bays for each packet based on the available disk space in the bays' disk partitions. The order in which you specify bays does not matter.

Note: Storage class names are case sensitive.

MultiSite installation creates a default storage class named **–default**. The storage bay and return bay for this class are created on the local host in the /var/adm/rational/clearcase/shipping directory. Each bay contains subdirectories named incoming and outgoing, which hold incoming and outgoing packets. Shipping operations look for packets in these subdirectories. Before using the store-and-forward facility, make sure that the disk partition where the shipping directory is created has sufficient free space for anticipated replica-creation and update packets.

multiutil commands that use the **–sclass** option use the **cq_default** storage class for packets that are not assigned to any storage class, and for packets whose storage class is not configured. The **cq_default** storage class is not created when MultiSite is installed. The **mkorder** and **shipping_server** commands use the **–default** storage class for packets that are not assigned to any storage class and for packets whose storage class is not configured. You can create additional storage classes for ClearQuest MultiSite packets, but you must use different storage classes for ClearQuest MultiSite packets and ClearCase MultiSite packets.

You must create *directory-pathname* with a standard UNIX **mkdir** command. You must also create the incoming and outgoing directories in the new bay. Packets placed in a bay are assigned the same owner, groups, and read-write permissions as the bay itself. (Execute permission and any special permissions on the bay are ignored.) Be sure to adjust these permissions (if necessary) to enable successful execution of MultiSite commands to process the packets and to guard against unauthorized access.

Note: The incoming and outgoing directories must be on the same file system.

Expiration Period

EXPIRATION *storage-class number-of-days*
EXPIRATION –default *number-of-days*

Default: 14 days for **–default**; none for **cq_default** (you must specify an expiration period).

Specifies the expiration period (in days) for shipping orders associated with the specified storage class. This period begins at the time the shipping order is generated. If a packet cannot be delivered to all of its destinations in the specified number of days, the packet is returned to the original sending host and one or more electronic mail messages are sent (see the descriptions in the sections *Administrator Address* and *Notification*).

Specifying **cq_default** as the storage class sets the expiration period for shipping orders that are not assigned to any storage class and for shipping orders whose storage class is not configured. Exception: When you generate a shipping order with the **mkorder** command and do not specify a storage class, the shipping order has the expiration period associated with the **–default** storage class.

A value of **0** (zero) specifies no expiration and delivery is reattempted indefinitely.

This setting is overridden by the **–pexpire** option to **syncreplica** or **mkreplica**.

The **shipping_server** program does not retry delivery of a packet. The **EXPIRATION** specification is useful only if you schedule periodic invocations of the shipping server to attempt delivery of any undelivered packets.

Packet Routing

ROUTE *next-hop host ...*
ROUTE *next-hop* **–default**

Default: None.

Controls the network routing of packets. Packets whose final destination is any of the *host* arguments are sent to the host named *next-hop*. This host is responsible for final delivery of the packet to its destinations (or additional forwarding). *next-hop* and *host* can be host names (which must be usable by hosts in different domains) or numeric IP addresses.

You can include multiple **ROUTE** entries in the configuration file. The special keyword **–default** accommodates all hosts that are not specified in another **ROUTE** entry.

Receipt Handler

RECEIPT-HANDLER *storage-class script-pathname*

Default: None.

Specifies a script for the shipping server to run for each packet received in a shipping bay.

shipping_server handles each packet that is received as follows:

1. Reads the shipping.conf file to find the appropriate **RECEIPT-HANDLER** entry for the packet.
   - If the packet is associated with a storage class and there is a **RECEIPT-HANDLER** entry for that storage class, **shipping_server** uses the *script-pathname* specified in that entry.
   - If the packet is not associated with a storage class and there is a **RECEIPT-HANDLER** value for the **–default** storage class, **shipping_server** uses that value.
2. Invokes the receipt handler as follows:

   script-pname [**–d**·**ata** *packet-file-pname*] [**–a**·**ctual** *shipping-order-pname*]
   [**–s**·**class** *storage-class*] **–o**·**rigin** *hostname*

   where

script-pname	Script specified in the **RECEIPT-HANDLER** entry.
–d·**ata** *packet-file-pname*	Location of the packet. This option is used only when the packet is destined for this host.
–a·**ctual** *shipping-order-pname*	Location of the shipping order. This option is used only when the packet is destined for another host.
–s·**class** *storage-class*	Storage class associated with the packet. This option is used only if the packet was associated with a storage class when it was created.
–o·**rigin** *hostname*	Name of the host from which the packet was first sent.

Note: If a packet is destined for both the local host and another host, both the **–data** and **–actual** parameters are used. The packet is imported at the replica on the host and then forwarded to its next destination.

Port Numbers

CLEARCASE_MIN_PORT *port-number*
CLEARCASE_MAX_PORT *port-number*

Default: None.

Caution: Set these entries only on hosts that can communicate through the firewall and have been installed with the MultiSite shipping-server-only option. To use the shipping server on a firewall system, you must also set the CLEARCASE_MIN_PORT and CLEARCASE_MAX_PORT environment variables in the **clearcase** script. For more information, see *Specifying Port Values* on page 52.

These entries specify the range of ports for the shipping server to use on a firewall system, and they are set as environment variables in the shipping server environment.

Guidelines for setting the values:

- The value range for **CLEARCASE_MIN_PORT** is 1024 through 65534.
- The value range for **CLEARCASE_MAX_PORT** is 1025 through 65535.
- The value of **CLEARCASE_MAX_PORT** must be greater than the value of **CLEARCASE_MIN_PORT**.
- We recommend that you use the range 49152 through 65535, which is the Dynamic/Private Port Range.

Timeout Period for Unreachable Hosts

DOWNHOST-TIMEOUT *minutes*

Default: None.

Specifies the number of minutes for the shipping server to wait before trying to contact a target host that was previously identified as unreachable.

If the shipping server tries to send a packet to a target host and determines that the host is unreachable, it creates a file in the /var/adm/rational/clearcase/shipping/ms_downhost directory. The name of the file is the name of the unreachable host. If one of the following parameters is set, the shipping server checks the directory for target hosts during future shipping operations:

- **DOWNHOST-TIMEOUT** setting in the shipping.conf file
- SHP_DOWNHOST_TIMEOUT_RETRY environment variable

If both parameters are set, the shipping server uses **DOWNHOST-TIMEOUT**.

If the target host is found in the ms_downhost directory, and the difference between the current time and the last modification time of the file is less than the timeout setting on the shipping server host, the shipping server does not try to send packets to the target host. If the difference is equal to or greater than the timeout setting, the shipping server tries to send packets to the target host. If neither this setting nor the environment variable SHP_DOWNHOST_TIMEOUT_RETRY is set, the shipping server attempts to send the packet to the target host. (Each attempt to send a packet to an unreachable host takes about 30 seconds.)

shipping_server

Store-and-forward packet transport server

Applicability

Product	Command type
MultiSite	MultiSite command

Platform
UNIX
Windows

Synopsis

shipping_server [**–scl·ass** *storage-class-name*] { **–pol·l** | *sources* ... }

This command is located in *ccase-home-dir*/etc on UNIX and *ccase-home-dir*\bin on Windows.

Description

This command processes one or more shipping orders on the local host and sends the associated packets or files to remote sites. After it delivers a file to all its destinations, **shipping_server** deletes the file unless one of the destinations is the local host.

Note: When **shipping_server** starts processing a shipping order, it locks the order. The lock prevents subsequent invocations of **shipping_server** from processing the order.

TCP/IP Connection

To transmit a file, **shipping_server** uses UDP to contact the **albd_server** process on the receiving host, and **albd_server** invokes **shipping_server** in receive mode on the receiving host.

If you are sending packets through a firewall (that is, the CLEARCASE_MIN_PORT and CLEARCASE_MAX_PORT environment variables are set), **shipping_server** tries to use TCP to contact the remote **albd_server**. If that connection fails, **shipping_server** uses UDP. For more information, see *Using Store-and-Forward Through a Firewall* on page 49.

On UNIX, **shipping_server** forks one subprocess for each packet that it sends. As many as 10 **shipping_server** subprocesses, each trying to send a single packet, can be started for each invocation of **shipping_server**. The same number of subprocesses are forked on the receiving machine. As a subprocess finishes, another can be started, but only 10 can run simultaneously.

After a TCP connection is established between the two **shipping_server** processes, they transfer the file. The receiving **shipping_server** selects a storage bay using the configuration settings in the shipping.conf file (UNIX) or MultiSite Control Panel (Windows). If a storage class is assigned multiple storage bays, available disk space determines the selection of a bay.

On UNIX, the packet file is created with the same owner and group as the storage bay directory, and its access mode is taken from the directory's read and write permissions. (The execute permission and special permissions, if any, are ignored.)

On Windows, the packet file inherits permissions from the Windows ACL on the storage bay directory.

Colon Characters in Packet Names

If a packet name contains a colon (**:**), **shipping_server** changes the colon to a period (**.**) during processing. This change allows packets to be delivered to Windows machines, which do not allow colons in file names.

Handling of File Name Conflicts

You can use the **mkorder** and **shipping_server** commands to transmit non packet files if the files are in the same directory as their associated shipping orders. If a file with the same name already exists on the receiving host, the new file is renamed to *filename*_1 (if you send another file with the same name, it is renamed to *filename*_2, and so on).

Setting a Timeout Period for Unreachable Hosts

You can set a timeout period during which the shipping server will not try to send packets to hosts that it previously identified as unreachable. For more information, see the **shipping.conf** (UNIX) or **MultiSite Control Panel** (Windows) reference page.

Log

On UNIX, **shipping_server** writes records of all packets sent and received, along with all errors, to file /var/adm/rational/clearcase/log/shipping_server_log.

On Windows, **shipping_server** writes records of all packets sent and received, notification messages, log messages, and all errors to the Windows Event Viewer.

Restrictions

Identities: You must have write and execute permissions on the directory containing the shipping order. On UNIX, you must own the data file or be **root**.

Locks: No locks apply.

Mastership: No mastership restrictions.

Other: The shipping order and the data file it specifies must be located in the same directory.

Options and Arguments

Restricting Processing to a Storage Class

Default
With **–poll**, processes all shipping orders in all outgoing storage bays and return bays on this host. With *sources*, processes all specified shipping orders.

–scl·ass *storage-class-name*
Processes shipping orders for the specified storage class only.

Specifying the Shipping Orders

Default
None.

–pol·l
Processes shipping orders located in some (if you use **–sclass**) or all storage and return bays defined in the shipping.conf file on UNIX or the MultiSite Control Panel on Windows.

Note: **shipping_server** processes only shipping orders whose file names start with the characters `sh_o_`. If you create shipping orders, name them according to this convention, or omit the **–poll** option and specify the shipping order pathnames.

On UNIX, only shipping order files that you own are processed. However, when **root** runs this program, shipping order files are processed regardless of ownership.

sources ...
One or more pathnames of files and/or directories. Each file you specify is processed if it contains a valid shipping order. For each directory you specify, **shipping_server** processes some (if you use **–sclass**) or all shipping orders stored in that directory.

Examples

In these examples, the lines are broken for readability. You must enter each command on a single physical line.

- Process all shipping orders in all MultiSite storage bays.

  shipping_server –poll
  <*no output means command succeeded or did not find any shipping orders*>

- Process a particular shipping order. Note that the pathname argument specifies the shipping order file, not the data file to be transmitted.

  /opt/rational/clearcase/etc/shipping_server
  /var/adm/rational/clearcase/shipping/ms_ship/sh_o_sync_sydney_19-May-02.09:
  48:45_7660_1
  <*no output means command succeeded*>

- Process all shipping order files in a specified directory.

  shipping_server "c:\Program
  Files\Rational\ClearCase\var\shipping\ms_ship\outgoing"
  <*no output means command succeeded or did not find any shipping orders*>

- Process all shipping orders in the storage bays of a specified storage class.

  /opt/rational/clearcase/etc/shipping_server –poll –sclass daily
  <*no output means command succeeded or did not find any shipping orders*>

See Also

mkorder, **MultiSite Control Panel**, **shipping.conf**, **syncreplica**
Chapter 10, *Troubleshooting MultiSite Operations*

syncreplica

Exports or imports update packets

Applicability

Product	Command type
MultiSite	multiutil subcommand

Platform
UNIX
Windows

Synopsis

- Export an update packet:

  sync·replica –exp·ort [**–cl·an** *clan-name*] [**–site** *site-name*] **–fam·ily** *family-name*
  –u·ser *username* [**–p·assword**] *password*
  [**–max·size** *max-packet-size* [**–lim·it** *num-packets*]]
  [**–nblocking**] [**–compress**]
  { { **–sh·ip** | **–fsh·ip** } **–wor·kdir** *directory*
  [**–sc·lass** *storage-class*] [**–pex·pire** *date*] [**–not·ify** *email*]
  | **–out** { *packet-file-pname* | *staging-area-pname* } }
  replica ...

- Import an update packet:

  sync·replica –imp·ort [**–cl·an** *clan-name*] [**–site** *site-name*] **–fam·ily** *family-name*
  –u·ser *username* [**–p·assword**] *password*
  [**–nblocking**]
  { **–rec·eive** [**–sc·lass** *storage-class*]
  | { *packet-file-pname* | *staging-area-pname* } ... }

Description

Synchronization of a replica with one or more sibling replicas is a three-phase process:

1. At one site, a **syncreplica –export** command creates an update packet that contains changes that have occurred in the replica at that site (and perhaps other replicas, as well).
2. The packet is sent to one or more other sites.
3. At another site, a **syncreplica –import** command applies the changes in the update packet to its replica of the same database. This step occurs at all sites that receive the packet.

Contents of an update packet:

- All changes that have occurred in the current replica since the last update generated for the destination replicas. (Changes already sent to the destination replicas are excluded from the packet).
- Changes that have occurred in other replicas, which the current replica has received in previous update packets from those replicas, but has not already passed on to the destination replicas.

In all cases, **syncreplica –export** creates a single logical update packet for use at all the specified destinations; the packet can be used to update those particular replicas only.

Notes on the Export Phase

MultiSite is designed for efficient updating of replicas. **syncreplica –export** attempts to exclude operations that have been sent previously. (However, there is no harm in sending an operation multiple times to the same replica; the first operation is imported and subsequent identical operations are ignored.)

syncreplica –export stores temporary files in the directory you specify with the **–workdir** option. This directory must not already exist and is deleted after the export packet is created.

Notes on the Import Phase

An update packet is applied to the appropriate replicas associated with the synchronization server that received the packet. You do not have to specify particular replicas or storage locations.

The import process applies update packets in the correct order. Therefore, you can specify packets in any order on the command line.

The database replica is not locked for normal database operations during the import phase, but it is locked for all other MultiSite operations.

Skipping Packets

syncreplica –import does not process an update packet in the following situations:

- The update packet contains changes that depend on other changes that have not yet been imported to this replica. This usually means that an update packet destined for this replica has not been sent or was lost during transport.
- Problems were encountered processing an earlier physical packet in a multiple-part logical packet.

In these cases, **syncreplica –import** displays an explanatory message.

Update Failures / Replaying Packets

In some cases, **syncreplica –import** begins to apply operations to a replica, but fails with an error message. For example, another process may have locked the database, causing the import to fail. After the database is unlocked, you can run **syncreplica –import** to process the entire update packet again.

There is no harm in importing update packets that have already been processed successfully; the same change will not be made twice.

For more information about update failures, see *Recovering from Lost Packets* on page 96.

Deletion of Update Packets

If a single invocation of **syncreplica –import** applies a packet successfully to all target replicas associated with the synchronization server, the update packet is deleted when the command completes its work. If the packet is processed with multiple **syncreplica –import** commands, it is not deleted.

Hooks Firing

ClearQuest hooks do not fire in response to changes made during packet import.

Handling Naming Conflicts

syncreplica resolves naming conflicts among objects created at different replicas. For more information, see *Resolving Naming Conflicts* on page 98.

Delayed Updates

syncreplica does not inform ClearQuest users of the updates to replicas. All active users see updates within a few seconds, through ClearQuest's normal database-polling routines.

Error Handling for Packet-Delivery Failures

If a packet cannot be delivered, it is sent through the store-and-forward facility to the synchronization server for the originating replica. A mail message is sent to the store-and-forward administrator. This occurs after repeated attempts to deliver the packet have all failed, and the allotted time has expired; it can also occur when the destination host is unknown or a data file does not exist. The store-and-forward configuration settings specify the expiration period, the e-mail address of the administrator, and the notification program.

Restrictions

You must have Super User privileges.

Options and Arguments — Export Phase

Specifying the Clan, Site, and Family

Default

Clan: First clan replicated at this site. If there is more than one clan at the site, **–clan** is required.
Site: Current site. If there is more than one site on this host, **–site** is required.
Family: No default; you must specify a family.

–cl·an *clan-name*

Name of the replica's clan.

–site *site-name*

Name of the replica's site.

–fam·ily *family-name*

User database family: Database name given to the user database when it was created.

Schema repository family: The family name is **MASTR**.

Specifying a User Name and Password

Default

You must specify a user name and password.

–u·ser *user*

Name of a user with Super User privileges.

–p·assword *password*

Password associated with the specified user.

Specifying the Update Packet Size

Default

When you do not specify **–maxsize**, the default packet size depends on the shipping method you use:

- Packets created with **–ship** or **–fship** are no larger than the maximum packet size specified in the shipping.conf file (UNIX) or the MultiSite Control Panel (Windows).
- Packets created with **–out** are no larger than 2 GB.

–max·size *max-packet-size* [**–lim·it** *num-packets*]

The maximum size for a physical packet, expressed as a number followed by a single letter. For example:

500k	500 kilobytes
20m	20 megabytes
1.5g	1.5 gigabytes

The **–limit** option limits the number of packets **syncreplica** generates; each packet is no larger than *max-packet-size*. Use this option when the disk space for your shipping bay or staging area is limited.

Disposition of the Update Packet

Default

None. You must specify how the update packets created by **syncreplica –export** are to be stored and/or transmitted to other sites. If you use **–ship** or **–fship** and omit the **–sclass** option, **syncreplica** places the packet in the storage bay location specified for the **cq_default** class in the shipping.conf file (UNIX) or the MultiSite Control Panel (Windows).

–shi·p
–fsh·ip

Stores the update packet in one or more files in a store-and-forward storage bay; **syncreplica** creates a separate shipping order for each physical packet, indicating how and where it is to be delivered. The destinations are the synchronization servers associated in the replica database with the *replica-name* arguments. (Synchronization server associations are created with **mkreplica –export** and can be changed with **chreplica**.)

Using **–fship** (force ship) invokes the shipping server to send the update packet immediately. Using **–ship** does not invoke this server.

–wor·kdir *directory*

A temporary working directory for **syncreplica** to use. This directory must not already exist and is deleted after the **syncreplica** export process finishes.

–sc·lass *class-name*

Specifies the storage class of the packet and shipping order. **syncreplica** looks up the storage class in the shipping.conf file on UNIX or the MultiSite Control Panel on Windows to determine the location of the storage bay to use.

–out *packet-file-pname*

The name of the first update packet. Additional physical packets, if any, are placed in files named *packet-file-pname_2*, *packet-file-pname_3*, and so on.

The update packets are not delivered automatically; use an appropriate method to deliver them. You can create a packet using **–out**, and deliver it using the store-and-forward facility. See the **mkorder** reference page.

staging-area-pname

The directory where packet files are stored.

Handling Packet-Delivery Failures

Default

If a packet cannot be delivered, it is sent through the store-and-forward facility to the synchronization server for the originating replica. A mail message is sent to the store-and-forward administrator. This occurs after repeated attempts to deliver the packet have failed, and the allotted time has expired; it can also occur when the destination host is unknown or a data file does not exist. The store-and-forward configuration settings specify the expiration period, the e-mail address of the administrator, and the notification program.

–pex·pire *date-time*

Specifies the time at which the store-and-forward facility stops attempting to deliver the packet and generates a failure mail message instead. This option overrides the expiration period specified for the storage class in the shipping.conf file (UNIX) or MultiSite Control Panel (Windows).

The *date-time* argument can have any of the following formats:

*date***.***time* | *date* | *time*
where:

date	:=	*day-of-week* | *long-date*
time	:=	*h*[*h*]**:***m*[*m*][**:***s*[*s*]] **[UTC [[+ | -]***h*[*h*][**:***m*[*m*]]]]
day-of-week	:=	**today** | **yesterday** | **Sunday** | ... | **Saturday** | **Sun** | ... | **Sat**
long-date	:=	*d*[*d*]**–***month*[**–**[*yy*]*yy*]

month := **January** | ... | **December** | **Jan** | ... | **Dec**

Specify the *time* in 24-hour format, relative to the local time zone. If you omit the time, the default value is **00:00:00**. If you omit the *date*, the default value is **today**. If you omit the century, year, or a specific date, the most recent one is used. Specify **UTC** if you want the time to be resolved to the same moment in time regardless of time zone. Use the plus (+) or minus (-) operator to specify a positive or negative offset to the UTC time. If you specify **UTC** without hour or minute offsets, the default setting is Greenwich Mean Time (GMT). (Dates before January 1, 1970 Universal Coordinated Time (UTC) are invalid.)

Examples:

```
22-November-2002
sunday
yesterday.16:00
8-jun
13:00
today
9-Aug.10:00UTC
```

–not·ify *e-mail-address*

The delivery-failure message is sent to the specified e-mail address.

If a failure occurs on a Windows host that does not have e-mail notification enabled, a message appears in the Windows Event Viewer. The message includes the *e-mail-address* value specified with this option and a note requesting that this user be informed of the status of the operation. For information about enabling e-mail notification, see the **MultiSite Control Panel** reference page.

Specifying the Destination Replicas

Default

None.

replica ...

Site name of the destination replica. You can specify one or more destination replicas. For example, **boston_hub** indicates that **boston_hub** will receive the update packet, while **boston_hub bangalore** indicates that both **boston_hub** and **bangalore** will receive the update packet.

Options and Arguments — Import Phase

Specifying the Clan, Site, and Family

Default

Clan: First clan replicated at this site. If there is more than one clan at the site,

–clan is required.
Site: Current site. If there is more than one site on this host, **–site** is required.
Family: No default; you must specify a family.

–cl·an *clan-name*
Name of the replica's clan.

–site *site-name*
Name of the replica's site.

–fam·ily *family-name*
User database family: Database name given to the user database when it was created.

Schema repository family: The family name is **MASTR**.

Specifying a User Name and Password

Default
You must specify a user name and password.

–u·ser *user*
Name of a user with Super User privileges.

–p·assword *password*
Password associated with the specified user.

Specifying the Location of the Update Packets

Default
None.

–rec·eive [–sc·lass *storage-class* **]**
Note: This option is useful only if you run **syncreplica** on the synchronization server.

Scans the current host's storage bays. Any unprocessed update packets intended for replicas associated with this host are applied to the appropriate replicas on the host. With **–sclass**, **syncreplica** scans only the storage bays of the specified storage class.

If **syncreplica** finds any replica-creation packets, it sends mail to the store-and-forward administrator. (If the current host is a Windows host and e-mail notification is not enabled, a message appears in the Windows Event Viewer.) Use **mkreplica** to import these replica-creation packets.

packet-file-pname | *staging-area-pname* ...
Processes each *packet-file-pname* as an update packet. For each *staging-area-pname* specified, locates all previously unprocessed update packets in the directory and applies them to the appropriate replicas.

Examples

In these examples, the lines are broken for readability. You must enter each command on a single physical line.

Exports

- At the **boston_hub** replica, generate an update packet for the **sanfran_hub** replica. Store the packet in c:\cqms\sanfran_hub_sync.xml.

  multiutil syncreplica -export -clan telecomm -site boston_hub -family SAMPL -user susan -p passwd -out c:\cqms\sanfran_hub_sync.xml sanfran_hub

  ```
  Multiutil: Packet file 'c:\cqms\sanfran_hub_sync.xml' generated
  ```

- Place the packet file in a storage bay for shipping at some later time.

  multiutil syncreplica -export -clan telecomm -site boston_hub -family DEV -user susan -p passwd -maxsize 500mb -workdir c:\work -ship -sclass cq_default sanfran_hub

  ```
  Multiutil: Packet file
  'C:\work\sync_BOSTON_HUB_26-March-02_10-55-16.xml' generated
  multiutil: Shipping order
  "C:\temp\cqms\ms_ship\outgoing\sh_o_sync_BOSTON_HUB_26-March-02_10-
  55-16.xml" generated.
  ```

- Similar to the preceding example, but ship the packet immediately.

  multiutil syncreplica -export -clan telecomm -site boston_hub -family DEV -user susan -password p -maxsize 500mb -workdir c:\work -fship -sclass cq_default sanfran_hub

  ```
  Multiutil: Packet file
  'C:\work\sync_BOSTON_HUB_26-March-02_10-56-43.xml' generated
  multiutil: Shipping order
  "C:\cqms\ms_ship\outgoing\sh_o_sync_BOSTON_HUB_26-March-02_10-56-43
  .xml" generated.
  multiutil: Attempting to forward/deliver generated packets...
  multiutil:   -- Forwarded/delivered packet
  C:\cqms\ms_ship\outgoing\sync_BOSTON_HUB_26-March-02_10-
  ---- NOTE: consult the NT event log for errors.
  ```

Imports

- Import all incoming update packets in the **cq_storage** storage class.

multiutil syncreplica -import -clan telecomm -site sanfran_hub -family DEV -user jcole -p passwd -receive -sclass cq_storage

```
Multiutil: 4 transactions from boston_hub have been replayed into
the MASTR database
Multiutil: 2 transactions from boston_hub have been replayed into
the DEV database
Multiutil: Deleting packet
C:\temp\cqms\ms_ship\incoming\sync_boston_hub_22-January-02_11-10-3
4.xml
```

- Process the update packet **sanfran_hub_sync.xml** at the **sanfran_hub** replica.

  multiutil syncreplica -import -clan telecomm -site sanfran_hub -family DEV -user jcole -p passwd c:\cqms\sanfran_hub_sync.xm

```
Multiutil: 1 transactions from boston_hub have been replayed into
the MASTR database
Multiutil: 2 transactions from boston_hub have been replayed into
the DEV database
Multiutil: Deleting packet c:\cqms\sanfran_hub_sync.xml
```

- Attempt to process the update packet **sanfran_hub_sync.xml** at the **sanfran_hub** replica before the **sanfran_hub** replica has been upgraded to the latest schema version.

  multiutil syncreplica -import -clan telecomm -site sanfran_hub -family DEV -user jcole -p passwd c:\cqms\sanfran_hub_sync.xml

```
Multiutil: The UPDATE_PACKET packet sent from boston_hub at
2002-01-22 15:15:50  is destined for schema revision 2, not 1;
re-execute syncreplica after site admin has upgraded database.
Multiutil: 2 transactions from boston_hub have been replayed into
the MASTR database
Multiutil: Preserving packet c:\cqms\sanfran_hub_sync.xml.
```

- Process all update packets in the incoming storage bay.

  multiutil syncreplica -import -clan telecomm -site boston_hub -family DEV -user susan -p passwd -receive

```
Multiutil: 1 transactions from SANFRAN_HUB have been replayed into
the MASTR database
Multiutil: 2 transactions from SANFRAN_HUB have been replayed into
the DEV database
Multiutil: Deleting packet
C:\temp\cqms\ms_ship\incoming\sync_SANFRAN_HUB_07-February-02_11-24
-49.xml
```

See Also

mkorder, **mkreplica**, **MultiSite Control Panel**, **shipping.conf**

Index

O

P

R

S

T

U

W