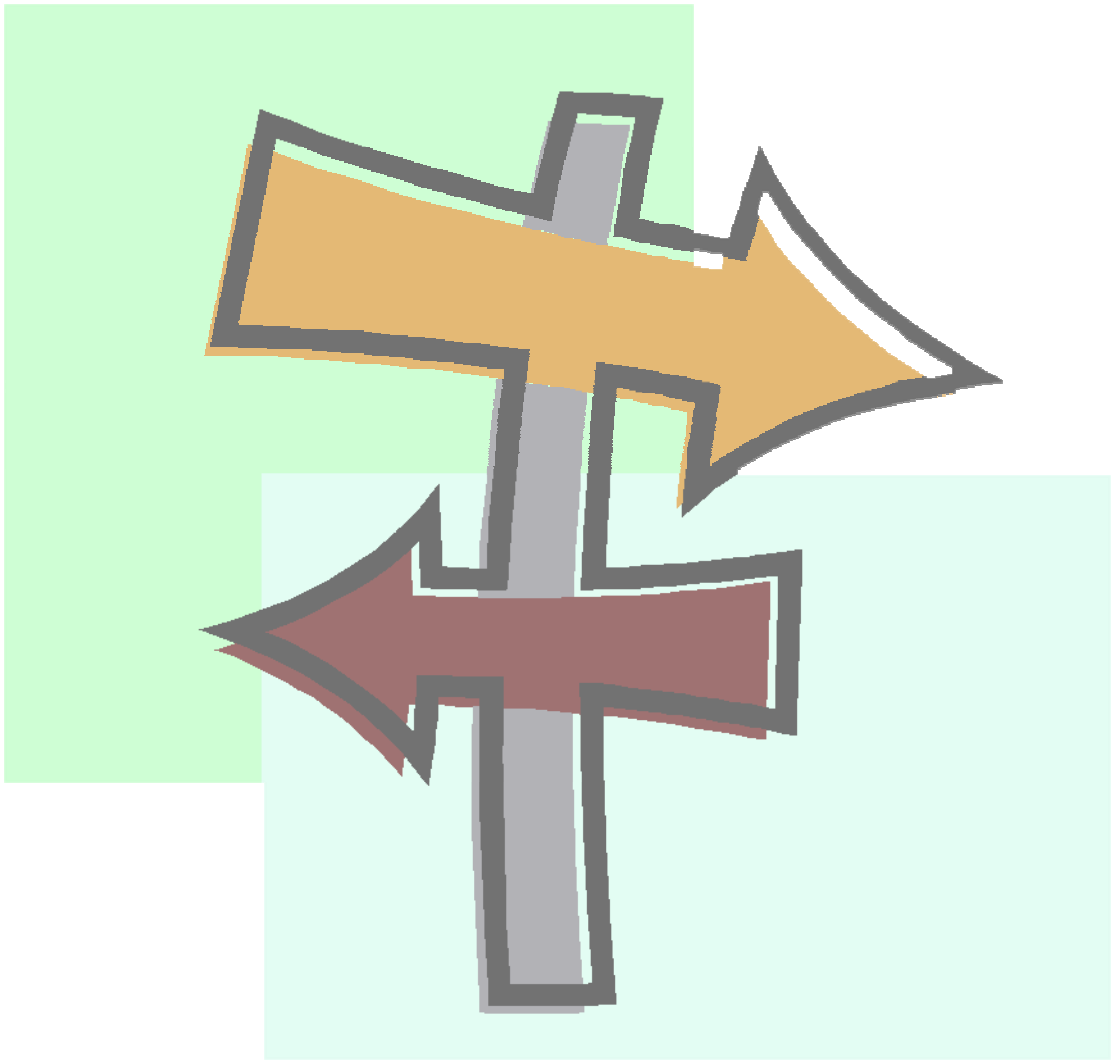
ClearCase MultiSite Divergence

Preventing and recovering from Divergence

Karl J. Weinert

April 7, 2010



Introduction

IBM® Rational® ClearCase MultiSite® allows developers at different locations to use the same versioned object base (VOB). Each location (site) has its own copy (replica) of the VOB. At any time, changes made in one replica can be sent in update packets to other replicas. Conflicting changes are prevented through the concept of mastership where only one replica has exclusive modify rights to an object at a time.

The purpose of this white paper is to provide clarification about divergence as it pertains to VOB replicas that are part of the same family. The information contained in this paper will help you distinguish between the types of divergence that may occur and provide information about various commands that can be used to help identify the presence of divergence within a ClearCase MultiSite environment.

This document does not discuss what might initially appear to be divergence in environments (such as those with secure sites) where replicas within a family are not synchronizing changes bi-directionally (one-way synchronization). It is expected that some of the replica data will differ between replicas in these environments. This document will also briefly discuss how synchronization lag can initially appear to be oplog divergence.

This document is designed to be read by ClearCase MultiSite Administrators who are responsible for the administration of their ClearCase MultiSite replications. Before proceeding you should have a good understanding of ClearCase MultiSite administration concepts such as replica synchronization and recovery as covered in the *IBM Rational ClearCase MultiSite Administrators Guide*.

Overview

The MultiSite synchronization process is such that operations performed at one replica are sent to other replicas in the family where a subsequent import replays those operations. What gets sent is controlled by means of operation logging described later in this document. Virtually all operations including changes in mastership are transferred to the other replicas in the family by means of operation logging. The few exceptions are noted in the following table.

Data propagated	Data not propagated
Elements, branches, versions (including derived object versions).	Derived objects (DOs) that have not been checked in as versions. DOs tend to be large and short-lived; transmitting them among multiple replicas is likely to be less efficient than rebuilding them at each replica.
Most kinds of type objects.	Trigger type objects. Triggers are usually used to implement local policies, and trigger type definitions often include pathnames that do not exist at other sites.
Metadata annotations: version labels, attributes, hyperlinks (including merge arrows and hyperlinks to administrative VOBs).	Individual “attached” triggers.
UCM objects: activities, baselines, components, folders, projects, streams	
Permanent locks (those created with the –obsolete option).	Temporary locks (those created without the –obsolete option).
Checkout records of elements and changes in checked-out directories. **Note:** The lscheckout –areplicas command lists checkouts in other replicas.	Contents of checked-out versions.
Event records.	
Mastership information	Mastership request settings
	Custom type managers.
	Changes to text mode property. When you create a new replica, it has the same text mode property as its parent replica, but subsequent changes are not propagated.

1. What is Divergence?

Divergence is an umbrella term used to describe any undesirable differences between the information contained at different replicas that cannot be resolved by synchronizing. It can show up as the same oplogs containing different operations, differences in the data, missing data or mastership discrepancies.

Oplog divergence

To understand oplog divergence it will be helpful to understand a basic oplog. An oplog is an entry in the database with the information needed to replay the operation that created it at other replicas in the family. When a VOB is first replicated, operation logging is enabled and all future operations are stored in the database as an oplog. These oplogs are sent along with the data required to replay the operation to the other replicas in the family through synchronization.

The following table shows the relevant information contained in an oplog.

Description	Command Output
Oplog Entry Order	25:
Operation	op= checkin
Replica OID	replica_oid= c8c84cf3.6c8e4714.9d3a.b7:4e:ee:18:ab:1b
Oplog ID	oplog_id= 25
Operation Time	op_time= 01-Jul-08.18:44:44UTC create_time= 01-Jul-08.19:21:10UTC

Oplog Entry Order is the order in which this oplog was played into the current replica whether it was created locally or imported by means of a syncreplica operation. Although not impossible, it is unlikely that the oplog entry order values will be the same at different replicas for a specific oplog.

Operation is the operation that was performed.

Replica Oid is the oid of the replica that originally performed the operation.

Oplog id is the value given to each oplog as it is created. It starts at 1 and increases sequentially by 1 for each oplog.
Note that in the dumpoplog output the oplog id and the replica oid will match at each replica in the family. Not unlike a<unique keyed pair> that can be used to distinguish <add better description\example here>

Operation Time

op_time time oplog was created at originating replica
create_time: time oplog was imported at the current replica

Note: op_time and create_time will always match at the originating replica, the op_time should be the same at every replica in the family, and the create_time will be different.

A listing of the oplogs can be seen by using the multitool dumpoplog command.

Refer to the *IBM Rational ClearCase MultiSite Administrators Guide* on the topic of dumpoplog for further information.

Note: oplogs can be scrubbed, and only oplogs that have not been previously scrubbed will show up in the dumpoplog output.

Here is an example of how to recognize the difference between two oplog_ids with the same value but created at a different replica

Boston Replica:

5:
op= chmaster
replica_oid*= 5262cfb5.d5304b77.b484.14:1a:3d:cb:ac:8d*
oplog_id*= 5*

The same **oplog** at Houston Replica:

19:
op= chmaster
replica_oid*= 5262cfb5.d5304b77.b484.14:1a:3d:cb:ac:8d*
oplog_id*= 5*

The same **oplog id** at the Houston Replica but created at a different replica (note the different replica_oid value):

25:
op= checkin
replica_oid*= c8c84cf3.6c8e4714.9d3a.b7:4e:ee:18:ab:1b*
oplog_id*= 5*

In the first two examples above the replica_oid and oplog_id both match indicating that this is the same oplog being looked at from two different replicas.

In the third example the replica_oid is different indicating that although the oplog_id is the same, this oplog was created at a different replica than the first two. Note that if we were to look at the complete oplog, the Operation Time (op_time) values will also be different in the third example compared with the first two.

How to determine if Oplog divergence is present

Oplog divergence is when two replicas have the same oplog_id from the same replica but different replicas show different information about that oplog_id. There will also be a different op_ time at each replica. Note the op_time in the dumpoplog output below.

Houston

5:
*op= **checkout***
replica_oid= 5262cfb5.d5304b77.b484.14:1a:3d:cb:ac:8d
oplog_id= 5
***op_time= 09-Oct-07.14:14:06UTC** create_time= 09-Oct-07.14:14:06UTC*

Boston

19:
*op= **chmaster***
replica_oid= 5262cfb5.d5304b77.b484.14:1a:3d:cb:ac:8d
oplog_id= 5
***op_time= 12-Oct-07.13:24:22UTC** create_time= 12-Oct-07.15:21:12UTC*

When this happens, an error will be seen during the import of a packet containing a divergent oplog and the import will fail preventing further propagation of the divergence.

multitool: Error: OPLOG DIVERGENCE DETECTED.
Please contact Rational Customer Support immediately.
replica_oid: 59dc01ea.f943417b.96c9.0d:65:cc:13:36:c1, oplog_id: 4098.
*In VOB, oplog **op_time:17-Oct-05.18:58:56UTC,** op:171.*
*In packet, oplog **op_time:17-Oct-08.19:00:15UTC**, op:127.*

Note that in the above error that the op:171 and 127 are values referencing a specific ClearCase operation such as a check in or check out.

This type of divergence is most often caused by improperly restoring a replica from backup without running restorereplica. It is also possible to cause this when moving a VOB while continuing to work in the original copy.

Refer to the IBM Rational MultiSite Administrators Guide under the topic of Restoring a replica from backup for a complete description of the steps required to properly restore a replica from a backup.

When a VOB is backed up, the backup copy is a snapshot of the VOB at the time of the backup. Since work has most likely continued in this VOB and that work will have been sent to other replicas in the family, the other replicas in the family will have more information about the restored VOB than the restored VOB has a record of. Restorereplica is the only procedure that can be used for the restored VOB to recover those changes.

There is no simple way to safely import data from a divergent replica into the healthy replicas. Recovering data from the divergent replica requires copying data from that replica to a neutral location and re-importing it once the replicas are returned to a healthy state.

Continuing to work in the divergent replica will only cause further divergence.

It is also possible by manipulating the epoch tables to skip the oplogs that are different at each site but have the same ID's and subsequently import oplogs ids that have not been imported to this replica previously. If this happens it will propagate the divergence to other replicas in the family making recovery more difficult.

It may be tempting to use this method as a workaround to this issue but there are far too many variables to successfully use this as a procedure to resolve divergence as the following example illustrates:

Example:

1. Create a new VOB
2. Add hello_world.c to source control creating version 1 in that VOB
3. Replicate the VOB (oplogs turned on)
4. Backup the VOB
5. Create version 2 of hello_world.c (oplog_id=1)
6. Synchronize the two replicas (oplog_id=1 sent to remote replica)
7. Restore VOB from backup without running restorereplica (now back to version 1 of file in original replica and epochs set back to 0)
8. Create a new version 2 of hello_world.c (Second oplog with oplog_id=1)
9. Create version 3 of hello.world.c (oplog_id=2)
10. chepoch -actual remote_replica (remote replica says I already have oplog_id 1 from you so set your epoch table to sent me oplog_id=2)
11. Synchronize the two replicas (only oplog_id=2 set)

Since version two in the example above was created once before the restore and once afterwards it will be divergent and the version already imported at the remote replica will have a different OID than the new one created after the restoration. However, because of the resetting of the epoch table, that oid will be skipped and the import will succeed with the "skipped oplog due to missing input" message. All future versions of this element on that branch will get skipped and the "*skipped oplog due to missing input*" message will be logged in the oplog.

Mastership Divergence

Mastership divergence is when replicas disagree about who masters an object. More than one replica believes it is has mastership of the same piece of data.

How can you determine if mastership divergence is present?

Unfortunately, mastership divergence isn't always easily detected until an operation is performed that cannot be replayed at another replica in the family due to a similar change being made to the same object at the other replica that masters the item.

Note the mastership discrepancy between two replicas in the following example.

Example:

*M:**view-r1\divergence-r1**\files>cleartool **describe** -l hello1.c@@\main*
branch "hello1.c@@\main"
created 09-Oct-07.10:13:53 by Karl (karl.user@myserver)
branch type: main
***master replica: rep-1@**divergence-r1 (defaulted)*

*M:**\view-r2\divergence-r2**\files>cleartool **describe** -l hello1.c@@\main*
branch "hello1.c@@\main"
created 09-Oct-07.10:13:53 by Karl (karl.user@myserver)
branch type: main
***master replica: rep-2@**divergence-r2 (defaulted)*

Since replica2 r1 and r2 both have mastership of the same branch instance, both sites will be able to create a new version on that branch. The next synchronization attempt would fail during import with the following error.

Error: Version 0x2 is not the highest on its branch

Similar errors indicating duplicate data would be seen for other objects such as applying labels and the creation of branches.

The opposite can also be true where two replicas think mastership belongs to the other replica. This situation is more likely just a timing issue where the mastership is in transit to the other replica and will simply resolve itself through normal synchronization.

Another possible cause that doesn't indicate divergence is if a packet has gotten lost in transit.

Check the current status of VOB synchronization with the multitool lsepoch command collected from all the replicas in question.

Refer to the *IBM Rational ClearCase MultiSite Administrators Guide* under the following topics for further details:

- lsepoch
- lsepoch and chepoch method
- Method2: lsepoch and chepoch method

Effects of divergence

The primary effect of divergence is discrepancies in the data between replicas. This can show up as duplicate versions, missing version(s), missing element(s), or metadata.

Duplicate but different Data

One scenario is when the same version of an element appears in both replicas but each has a different creation date, different oids and likely different data contained within the version. Since the oplogs contain two different operations (as in the "op=" entry in the example above) it follows that the data associated with that oplog will also be divergent. This is most frequently caused by a mismatch in mastership.

Below is an example where the same version has different create times and different oids at two different replicas.

Example:

At replica 1 site:
*M:**view-r1\divergence-r1***\files>cleartool ***describe*** -l hello.c@@\main\2*
created 2007-07-10T10:28:34-04 *by Karl*

*M:**view-r1\divergence-r1***\files>cleartool ***dump*** hello.c*
*hello.c (**fd2c3af0.47354216.84f6.33:b5:c3:84:09:d5**)*
files\hello.c@@\main\2
oid=fd2c3af0.47354216.84f6.33:b5:c3:84:09:d5 dbid=76 (0xec804f)

At replica 2 site:
*M:**view-r2\divergence-r2***\files>cleartool ***describe*** -l hello.c@@\main\2*
version "hello.c@@\main\2"
created 2007-07-15T09:54:14-04 *by Karl*

*M:**view-r2\divergence-r2***\files>cleartool ***dump*** hello.c*
*hello.c (**0ebe4cb7.e4374a0b.a315.d9:c2:d8:6e:56:1a**)*
files\hello.c@@\main\2
*oid=**0ebe4cb7.e4374a0b.a315.d9:c2:d8:6e:56:1a** *dbid=76 (0xec804f)*

Note: The dbid's will usually be different at each replica and are not an indicator of divergence.

This type of divergence will usually get caught with an error indicating that an import failed because the same data already exists at the importing site. Such as the Version 0x2 is not the highest on its branch error when importing a duplicate version number mentioned in the previous section.

Missing Data

Another scenario would be missing data at a site. When an operation is played into a site where, for whatever reason, the object being acted upon has been deleted at the importing site the oplog will be replayed but with the message *skipped oplog entry "<operation>" due to missing input*.

This is by design to prevent synchronization failures when, for example, an element gets removed at the replica that masters the element but a user at another replica in the family edits a version of that element that they have mastership of before the rmelem oplog is imported at that site. When they synchronize the replica to the site that has removed the element, it will simply skip whatever changes that user made to their version.

> *13:*
> *op= checkin*
> *replica_oid= 4f1ccbbc.a6954352.9014.bc:2f:03:1a:7b:44 (rep-2)*
> *oplog_id= 4*
> *op_time= 13-Oct-07.14:24:40UTC create_time= 13-Oct-07.14:24:55UTC*
> *event comment= **"Skipped oplog entry "checkin" due to missing input:** edd8162b.3e364e48.99c9.eb:c9:a2:99:58:3c"*
> *data size= 116 data= 0xecd670*
> *----------*
> *ver_oid= f76347a9.b2094665.afc0.30:69:35:6f:b9:b7 (*object not found*)*

To explain it further, the operation replayed from the oplog relies on the oid of the version being acted on. For example the data in a checkout oplog will show a reference to the previous version's oid. This can be seen with dumpoplog –long output.

Example:
pred_ver_oid= 08520713.532f46d3.88eb.f5:12:5b:ce:29:14

If the element has been removed at the importing replica, then this oid will not exist and the *skipped oplog entry "<operation>" due to missing input message* will be seen in the oplog.

Although most of the time this is expected behavior, it becomes an issue when a user is able to remove something that should be protected by mastership but is not due to mastership divergence.

This can be a little more difficult to catch since all subsequent updates to that data will continue to import with the warning message and usually is discovered only when a user at another replica needs to access one of the missing versions.

2 Causes of Divergence

Most divergence is caused by user error. Some of the most common are described below.

Oplog divergence

There are a number of things that can cause oplog divergence.

⇒ Improper restoration of a replica where a replicated VOB is restored from backup but restorereplica is not performed before putting it back into production.

Refer to the *IBM Rational ClearCase Administrators Guide* under the topic of **Restoring and replacing VOB replicas** for information about the proper procedures for restoring replicated VOBs.

⇒ Improper moving of a VOB to a new server where the original VOB is left in production during the move.

Both of these operations will cause oplog divergence. In the case of the restored replica, the oplogs contained in the database will only be as current as the latest backup. And the remote replica will know about more oplogs than the recently restored backup. When a VOB is restored from backup and work is performed in that VOB before running restorereplica, new oplogs are being created at the restored replica with the same oplog_id as those already imported at one or more remote replicas.
Note even if the same exact operations are performed in the exact same order there will still be divergence because the oid of any newly created data will be different than the data that was created before the restoration.

⇒ Improperly moving a replica will cause divergence because two copies of the same replica will be making changes, creating their own stream of oplog_id's.

Since a replica cannot update itself there will be no way to export or import the changes made at one of the duplicate replicas to the other. If this happens the first replica to send a synchronization packet to the other replicas in the family will be the healthy one. The other replica should be discarded saving any changes that may be needed at the other replicas in the family. It is possible however for both replicas to send out update packets to two separate replicas in the family. This can infect multiple replicas in a family and will be much more difficult to recover from.

Refer to the *IBM Rational ClearCase Administrators Guide* under the topic of Moving VOBs for information about the proper procedures for moving a replicated VOB.

⇒ Completing a restorereplica using the *-override* switch before importing all necessary updates from the remote replicas as well as using override in order to rerun the operation in optimized mode.

Optimized mode is when you choose a list of replicas to require updates from when initially running the restorereplica operation.

Note: It is still recommended to update all replicas with the restorereplica oplog even if an update is not required from that replica. This can prevent future incarnation errors.

Refer to the following documentation for additional information:

- *IBM Rational ClearCase MultiSite Administrators Guide*: restorereplica
- *IBM Rational ClearCase MultiSite Administrators Guide*: Replica incarnation is old
- Technote 1151039 **Replica incarnation is old**

Mastership divergence

Improper use of *chmaster –all –obsolete_replica* against a replica that is still in production.

This is a problem when the –obsolete_replica switch is used to pull mastership from an unavailable replica but the unavailable replica is still online and in use. This will mean that two replicas believe they have mastership of the same data and will be able to modify, and synchronize that data. Since mastership prevents access to data, a check of mastership is not performed when importing an oplog. An operation performed in this manner will get imported at other replicas in the family without error unless the operation conflicts with an operation performed at the true mastering replica. This will produce errors such as the following during import.

text_file_delta: Error: ***Version 0x2 is not the highest on its branch*** *<0x0,0x2> in "C:\ccstore\vobs\divergence-r1.vbs\s\sdft\30\39\0-fe0da1e5198f4f91a31c532bc16cbbde-ui"*
multitool: Error: Type manager "text_file_delta" failed create_version operation.
multitool: Error: Unable to replay oplog entry 62: error detected by ClearCase subsystem.
62:
op= checkin
replica_oid= 06d2c8e8.91474592.9b4b.e3:cf:cc:2f:0a:9d (rep-2)
oplog_id= 55
op_time= 29-Oct-07.21:32:34UTC create_time= 29-Oct-07.21:32:34UTC
*version_oid= 6e96c27d.291a4479.a4eb.5d:04:25:a0:61:36 (*object not found*)*
event comment= ""

data size= 116 data= 0xed1680
----
*ver_oid= 6e96c27d.291a4479.a4eb.5d:04:25:a0:61:36 (*object not found*)*
ver_num= 2

This type of divergence is introduced when synchronization continues despite oplog divergence being present in the VOB.

3 Prevention

Since minimal restrictions are required to run the restore procedure and the *chmaster –all –obsolete_replica* the best method of prevention is to have a good policy in place should any of the above procedures be planned.

All that is required is one of the following identities:
- VOB owner
- **root** (Linux® and the UNIX system)
- Member of the ClearCase® administrators group (Windows)

Since this is the case, it is important to have a good policy in place which all VOB owners are aware of. All VOB administrators should have a good understanding of the risks of these procedures.

4 Recovery

There is no way to recover once divergence has been introduced into a family except to replace one or more replicas. Fortunately, most times an error will be received when attempting to import a packet with oplogs that are divergent from the importing replica. This error will usually prevent the packet with the problem oplog from importing and propagating the divergence.

If an error is received, work should be stopped at the recently restored replica to prevent data loss that will either have to be re-created or manually exported from the problem VOB and imported into the healthy VOB after it has been restored properly.

The following error indicates that this packet contains an oplog with the same oplog id as one previously imported into this replica but it has a different operation and or the same operation but performed at a different time.

For vob <VOB Tag>
multitool: Error: OPLOG DIVERGENCE DETECTED. Please contact Rational Customer Support immediately.
replica_oid: ee887eda.9df211d4.bdc4.00:90:27:85:69:b4, oplog_id: 361675.

In VOB, oplog op_time:25-Feb-03.01:01:35UTC, op:134.
In packet, oplog op_time:25-Feb-03.20:07:14UTC, op:134.
multitool: Error: Cannot apply sync. packet <packet name> to <VOB Storage Location>: error detected by ClearCase subsystem

Errors indicating that the object being acted upon already exists at the importing replica suggest that two sites have mastership of the same object and both have acted upon that object independently from one another.

multitool: Error: The most recent version on branch "\main" is not the predecessor of this version.
multitool: Error: Unable to replay oplog entry 8: ClearCase object not found.

text_file_delta: Error: Version 0x2 is not the highest on its branch

multitool: Error: Element already has a branch of type <branch name>

These errors could indicate that a chmaster operation that was performed was not included in the backup copy of the VOB which allowed two replicas to have mastership of the same object at the same time. The more likely scenario is that *chmaster –all –obsolete_replica* was run against a replica that still exists or two sites ran *chmaster –all –obsolete_replica* against the same replica simultaneously.

If divergence is caught before it is imported to another replica in the family, restoring the VOB from backup and running restorereplica immediately should resolve the issue. If any data has been imported to another VOB in the family the replica will need to be replaced using the "Replacing an existing replica" procedure documented in the *IBM Rational ClearCase MultiSite Administrators Guide*.

If data is needed from the improperly restored replica, it will need to be moved to an isolated location and the data will need to be copied out manually.
Any data that had been created since the restoration can be found by dumping the oplogs (see example below), however, it will still require manual intervention to copy it to a neutral location to be copied into one of the healthy replicas.

multitool dumpoplog –long –name –vreplica <restored replica> -since <date the replica was originally restored>

Notes and warnings

The following is a list of things to rule out when divergence is suspected.

⇒ **Synchronization Delays**

If divergence is suspected but no error is present, a comparison of the epoch tables using the lsepoch command should help to clarify if the problem is related to a synchronization delay or actual divergence.

⇒ **Duplicate named metadata (such as brtype, lbtype, or others)**

When two replicas create metadata with the same name it automatically gets renamed at the importing replica. This can often cause confusion because describing the metadata using the original name will show conflicting mastership. Both replicas show that they master the object and synchronization does not resolve the issue. Comparing the oids obtained by running cleartool dump against the item should clear up any doubt that they are in fact two different objects.

Refer to the *IBM Rational ClearCase Administrators Guide* under the topic of Automatic renaming of type objects and replica objects for further information.

Note This can also happen with replicas if the same replica name is used to create a new replica from two replicas.

⇒ **Two separate elements with the same name exist in different versions of a directory**

This is known as evil twins and should not affect MultiSite operations.

Refer to Technote 1125072 **About Evil Twins** for more information

Summary

There are various causes for divergence, some of which are preventable. Be sure you have a solid backup strategy in place for your VOBs and that the proper documented procedures are followed whenever restoring a VOB that is replicated. Since there are circumstances that mistakenly appear to be divergence, be sure to investigate and eliminate all other possible causes as they may have a very simple solution. Replacing a replica is meant only to be used as a last resort when divergence has been identified.

References

The following were used in references or as other sources of information:

⇒ IBM Rational MultiSite Administrators Guide
dumpoplog
Restoring a replica from backup
lsepoch
lsepoch and chepoch method
Method 2: Lsepoch and chepoch method
Restoring and replacing VOB replicas
Restorereplica
Replica incarnation is old
Automatic renaming of type objects and replica objects
Replacing an existing replica

⇒ *IBM Rational ClearCase Administrators Guide*
Moving VOBs

⇒ Technote 1125072 **About Evil Twins**

⇒ Technote 1151039 **Replica incarnation is old**

Acknowledgements: Special thanks to Brian Cowan, David Robinson and Chris Green for their contributions to this paper.