Version 12.1 Administration Guide

Product Delineation

The phrase ClearTrigger is used throughout this document. In every instance where ClearTrigger applies and is used, the product **ClearTrigger Lite** also applies. So "ClearTrigger" can be taken to mean or interpreted as "ClearTrigger or ClearTrigger Lite" in every instance.

ClearReplica Overview

ClearReplica is a custom add-on to IBM Rational Software ClearCase product. It serves to provide a means of providing geographically disjoint development as a less expensive alternative to ClearCase Multisite. ClearReplica provides many of the replication features of ClearCase Multisite as well as additional features. With ClearReplica you can replicate whole VOBs, replicate "portions" of VOBs, replicate a "single file" in a VOB or even replicate based on branch filters; you can also replicate multiple VOBs in to single VOBs, single VOBs to multiple VOBs and replicate VOBs to other VOBs within the same ClearCase region. These options allow more flexibility to create on-line backup VOBs and staging or distribution VOBs.

ClearReplica is licensed differently than ClearCase Multisite in that ClearReplica License cost is not a factor of how many VOBs are replicated or how many users use those VOBs. ClearReplica is licensed per location, therefore is cost the same to replicate one VOB with one user from Atlanta to India as it does to replicate 100 VOBs with 300 users in Atlanta to India. **Sites can in many instances replace hundreds of Multisite licenses with one or two ClearReplica Licenses.** You simply need one license at each site if development is performed at both sites, or a single license at one site if the off site is just for backups or disaster recovery. This relaxed licensing result in huge upfront cost savings over traditional Multisite.

ClearReplica requires that ClearTrigger 12.10 or higher is installed at the sending and receiving site; again an unlicensed version of ClearTrigger will suffice at the receiving site if no development is performed on the receiving VOB there.

Some restrictions that exist in ClearCase Multisite are relaxed in ClearReplica. For instance, developers can work on the "same" branch at both locations for there is no Multisite mastership restriction. Geographically disjoint development teams can still work on separate branches and this is still the recommended method.

Use ClearReplica for geographically distributed development teams or use it for provide a hot backup to recover lost versions in a fraction of the time. Use ClearReplica to automatically distribute code or binaries located in a directory in a VOB without having to use the bandwidth of replicating the whole VOB. Replicate portions of the VOB to teams all over the world or across the hall. Simply install and configure ClearReplica on top of a ClearTrigger installation and get started!

ClearReplica Advantages over ClearCase Multisite (Changed in 12.0)

ClearReplica was designed as a low cost, but still feature-rich alternative solution to ClearCase Multisite so there are specific concerns of current and perspective Multisite users that were addressed.

Cost and licensing – cost of entry

ClearReplica is much more cost effective. Multisite is licensed per user of a replicated VOB so the cost increases when user and VOB usage grows, replicating a single VOB with Multisite that has 100 users in "Atlanta" and 30 users in "India" requires 130 licenses of Multisite. ClearReplica is licensed per "location" so two licenses of ClearReplica and ClearTrigger are needed (one of each at each site). Doubling the number of users of Multisite replicated VOBs doubles the cost of "replication", while doubling the number of users of ClearReplica replicated VOBs adds no additional cost.

Enhanced Replication Flexibility

ClearReplica allows more control over what is data is replicated. Multisite allows an organization to replicate a whole VOB from one ClearCase region to another region. ***ClearReplica allows you to replicate the whole VOB, a portion of the VOB, just a single file, or even replicate select branches or labels if desired***. No more chewing up corporate bandwidth replicating the entire development VOB when you actually desire to give the offsite location access to just a few directories and/or branches. With ClearReplica you can actually "replicate" portions of ***many*** VOBs in one or more locations and place them together on a ***single*** VOB in another location. Replicate more often knowing you are only replicating the portion of the VOB you requested. Now you can replicate to a partner/vendor the pieces from 20 VOBs and present the data to them as a single VOB. Replicate just the final product to a client from the development VOB without having to replicate "development" or create a separate VOB just to replicate the product. With ClearReplica you can even "replicate" just select branches or all branch except certain branches.

Extended "Deployment" capabilities of ClearCase

Use ClearReplica "deploy" from ClearCase VOBs to regular flat file systems on local machines or remote machines around the world. ClearReplica "deployment" only requires the license at the "shipper" site. Updates to a ClearCase VOB can be pushed as regular directories on countless machines making deployment of documents, web pages, patches, registry updates or even ClearCase triggers a snap. ClearReplica logs of what is sent and received so you know what has been deployed.

"Replication Triggers" allow further automation and notification to your process.

Use "Replication Triggers" to auto-run scripts before or after the sending or receiving of packets. Use any in-house script to perform custom logging, send email, install/execute deployed files or lock VOBs by adding a ClearReplica replication trigger. ClearReplica pre_ship, post_ship, pre_receive and post_receive triggers help manage your distributed environment. No more checking log files for certain events, have email send to you when events happen or have ClearReplica automatically

perform post deployment scripts. The destination machines do not even need to have ClearCase installed.

Automatic data encryption

All data packets sent via ClearReplica are encrypted and compressed.

Additional Peer-to-Peer Encryption

All data is by default encrypted using an internal ClearReplica Data Encryption key so that only ClearReplica can decrypt ClearReplica data packets (perfect for closed or otherwise private networks). Additional encryption can be enabled that requires the sender and receiver to agree on a generated encryption key known only to the Sender and Receiver, thus allowing for the secure transportation of packets across open, public or 3rd party networks or even thought the use of 3rd party courier services. Even those with ClearReplica cannot decrypt the packet without the Peer-to-Peer encryption key.

"Replication/Deployment" to disconnected machines or those with Dynamic IP addresses

ClearReplica facilitates the transfer of ClearReplica data between two ClearReplica servers that do not have direct access to each other, but share access to one or more machines between them that can be used as a ClearReplica **'proxy'** machines. The machines used in the transfer that are not actually the source or destination ClearReplica Server constitute the 'proxy chain' of machines. ***ClearReplica can also facilitate the transfer of data between machines that do not have a static IP address*** (e.g. salesman laptops) so that either "replication" or "deployment" data can be sent to those machines anytime they are connected to the web.

Nearly free "disaster recovery", "backup sites" or "distribution hubs"

ClearReplica "replication" to read-only VOB sites only require the license at the "shipper" site. ClearReplica requires the ClearReplica and ClearTrigger software at both the "shipper" and the "receiver", but if you are just replicating to the offsite location and no development is performed there, then ClearReplica Licensing is relaxed at the "receiver" site. ***Backup hundreds of VOBs to several locations or across the hall with only ONE ClearReplica license.*** A developer can access the "hot backup" VOB to see a version they just accidentally deleted in the "production" VOB and get the version without having to burden the corporate IT department.

Confirmation of packet delivery (automatic packet recovery)

Before ClearReplica "ships" a packet to a location it gets ***confirmation*** from that location of the last successful packet processed. ClearReplica automatically adjust and send all requested data "since" the last confirmed successful delivery. ***Dropped or lost packets are automatically recovered.*** Perform off-line maintenance on your VOB without worrying about missing packets other locations sent you, all changes will be sent automatically, the first time the location can communicate.

Relaxed Mastership

ClearReplica has relaxed mastership. Users of ClearReplica VOBs can actually work on the "same" branch in different replicas and actually see each other changes immediately. Now any of the

"replica" sites can make an element because there is no mastership to get in the way of either site changing the /main branch to make the new element.

Relaxed Replica Management

Stop "sending" or "receiving" packets at any time without performing "remove replica" commands. Sites can "opt-out" of replication at any time without having to coordinate with other replicas or worry about breaking other replicas because you had mastership of shared data. You can "opt-in" at any later time and quickly get back in sync or just choose to continue from the current point.

Parallel and distributed shippers and receivers

You can run multiple ClearReplica Shippers or Receivers in parallel, even across distributed machines from heterogeneous architectures.

Receiver-side control of replicated rmtype -lbtype

ClearReplica receiver processes can be configured accept or ignore cleartool rmtype -lbtype request from remote sites.

Receiver-side control of replicated rmtype -brtype (new to 12.0)

ClearReplica receiver processes can be configured accept or ignore cleartool rmtype -brtype request from remote sites.

Receiver-side control of replicated rmtype -attype (new to 12.0)

ClearReplica receiver processes can be configured accept or ignore cleartool rmtype -attype request from remote sites.

Receiver-side control of replicated rmelem

ClearReplica receiver processes can be configured accept or ignore cleartool rmelem request from remote sites. The receiver processes can be configured to: ignore all remote rmelem requests, accept only remote rmelem –file request, accept only rmelem –directory request or accept all remote rmelem request.

Receiver-side control of replicated rmbranch and rmver (new to 12.0)

ClearReplica receiver directories can be configured accept or ignore cleartool rmbranch or rmver request from remote sites.

Receiver-side control of replicated lock/unlock:lbtype

ClearReplica receiver directories can be configured accept or ignore cleartool lock/unlock label type request from remote sites.

Receiver-side control of replicated lock/unlock:brtype (new to 12.0)

ClearReplica receiver directories can be configured accept or ignore cleartool lock/unlock branch type request from remote sites.

Receiver-side control of replicated lock/unlock:attype (new to 12.0)

ClearReplica receiver directories can be configured accept or ignore cleartool lock/unlock attribute type request from remote sites.

Receiver-side control of replicated lock/unlock elements (new to 12.0)

ClearReplica receiver directories can be configured accept or ignore cleartool lock/unlock request for elements from remote sites.

Receiver-side control of replicated lock/unlock branch instances (new to 12.0)

ClearReplica receiver directories can be configured accept or ignore cleartool lock/unlock request for branch instances from remote sites.

Receiver-side control of replicated symbolic links

ClearReplica receiver processes can be configured accept or ignore cleartool symbolic links creation request from remote sites. The receiver processes can be configured to: ignore all remotely received symbolic link data or accept and process all received symbolic link data.

Shipper-side control of replicated symbolic links

ClearReplica shipper processes can be configured send cleartool symbolic links creation to remote sites or ignore this symbolic link data completely.

What is ClearReplica "Replication"?

ClearReplica "replication" is different than Multisite replication so it is important to know the differences. Multisite replicates all data in the VOB (minus triggers) while ClearReplica "replication" concentrates on the element "versions" and the metadata attached or required to replicate it. Rather than overview each product it is easier to explain examples.

Metadata types: In ClearCase Multisite when branch types are created in VOB A then are replicated in VOB B at the next replication regardless if they are used in either replica. In ClearReplica the type is replicated when the first element version that requires the type is replicated. On-demand creating of metadata types yields to smaller packets and eases the burden on the corporate bandwidth.

Mastership of branches: In ClearCase Multisite branch types are "mastered" by a replica, such that other replicas cannot make changes on that branch. Though in general it is a good practice not to use the same branch between sites, it does lead to situation where only one site can make new elements. In ClearReplica this restriction is relaxed. Though it is still recommended that geographically disjoint development teams still do not share a branch it is still possible when needed and any "replica" can now make an element.

Creating and Removing elements, branches or versions: In ClearCase Multisite only the "mastering" replica can create or remove elements, branches or versions. With ClearReplica any "replica" can make a new element because there is no mastership to get in the way. With ClearReplica any "replica" can also remove an element as well, initially from their "replica" but also

from remote replicas if allowed by that replica site. In ClearReplica "destructive" actions are not by default replicated, this allows ClearReplica "hot backup VOBs" to retain their usefulness (after all the reason to have a "hot backup replica" is to recover the data that you just lost in the "production replica"). However, each receiver can be separately configured to accept or ignore remote request to remove elements, branches or versions if this is desired.

Customer Support

To obtain additional information on ClearReplica or other services offered by A Better Solution, Inc., visit our web site at www.abs-consulting.com. To report problems with the ClearReplica software or documentation, please send e-mail to clearreplica_team@abs-consulting.com.

Supported Platforms

ClearReplica integrates with ClearCase installations running ClearTrigger 10.3 and higher. The system requirements for ClearReplica are consistent with those used to run ClearCase versions 3.2 and higher. ClearReplica will operate in a mixed version environment as well as a mixed OS environment. The following operating systems are supported for ClearReplica:

- **WINDOWS:**
  - **Windows 9x**
  - **Windows NT**
  - **Windows 2000 and higher**
- **UNIX:**
  - **Solaris**
  - **SunOS**
  - **HP-UX**
  - **RedHat Linux**
  - **AIX**
  - **CentOS**
  - **SUSE Linux**
  - **Solaris X86**

Installing ClearReplica

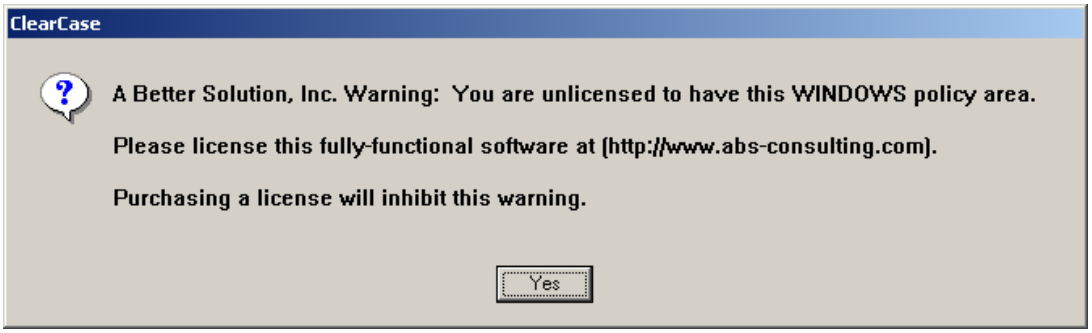
For most installation issues refer to the **ClearReplica Installation Guide**. In that document you can read about how to install ClearReplica.

License Registration

ClearReplica requires ClearTrigger 10.3 or higher in order to run. Soon, you will see this is not at all restricting, but in fact leads to a very relaxed licensing schema. A ClearReplica license is tied to a ClearTrigger licenses (actually the ClearTrigger depot) and not to how many VOBs are replicated or users of those VOBs. A single license of ClearTrigger and ClearReplica is "usually" needed at both locations though in some instances a single license of each will suffice for both the sending and receiving locations.

For example, consider Acme Inc. that has a site in Atlanta with 200 VOBs and 400 ClearCase users working in a Windows only environment could replicate all VOBs with a single license of ClearTrigger and a single license of ClearReplica. If the receiving site (for example London) was just a backup site or a read only site for Atlanta they could receive the replicas with no additional cost. If they also required the ability to develop on the replicas then it would require one (1) additional ClearTrigger license and one (1) additional ClearReplica license. Before contacting A Better Solution, Inc. to request a license, consider how many licenses you will need. You will receive a license request form when you purchase ClearReplica from A Better Solution, Inc. Return the completed form to A Better Solution, Inc. via e-mail or fax. Sites may proceed with installation and configuration of ClearReplica in the interim while a license is being obtained (ABS actually encourages this "try it first" method). ClearReplica is distributed as fully functional software with an **evaluation license key**.

If necessary, sites may also operate using an evaluation license. Once the license key is obtained, place it in the **ClearTrigger** *clearbits_file*. Use of an evaluation license will cause random ***Splash Messages*** from the vendor similar to the one below. *Splash Messages* will not change the functionality of ClearTrigger or ClearReplica.



To read more on installing your license key refer to the section entitled: **Installing your ClearReplica License String**.

Installing your ClearReplica license String

The **clearbits_file** is the configuration file used by ClearTrigger (or ClearTrigger Lite). It holds ClearTrigger licensing and ClearTrigger region configuration information used to find, enforce and execute ClearTrigger policy for the region. The **clearbits_file** is also where the ClearReplica key resides or actually the **ClearTrigger/ClearReplica** combo key.

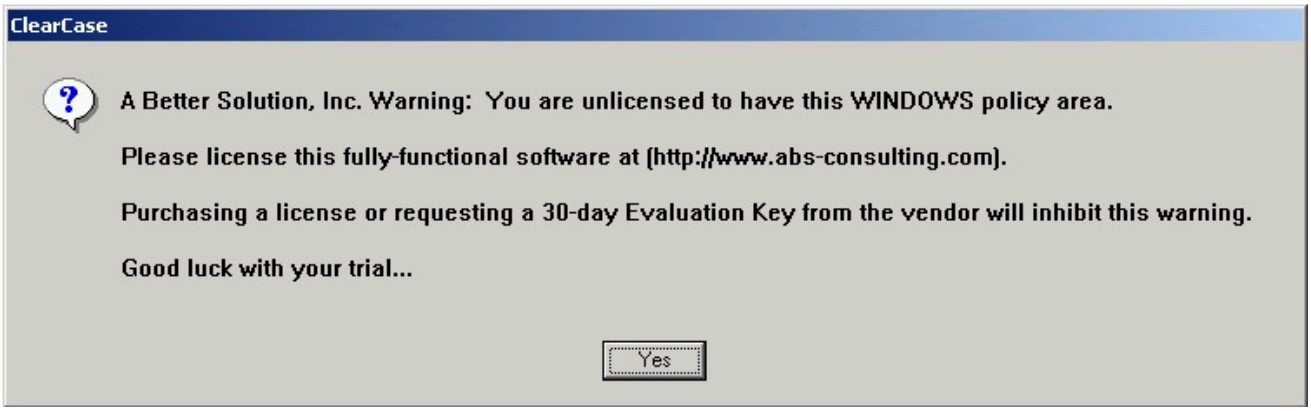
The first section of clearbits.txt is the license section. It contains the Product Name, Vendor Name, License Key Generator ID, Windows License Key, UNIX License Key, and your Organization Name. Replace the default license entry with your license key for each architecture license purchased from A Better Solution, Inc. or a licensed reseller. If an evaluation license key was acquired, place it here. For ClearTrigger, a key would look similar to the three (3) examples below:

cleartrigger ABS licman;123456789;000000000;ACME Inc – East Campus;
cleartrigger ABS licman;000000000;123456789;ACME Inc.;
cleartrigger_lite ABS licman;123456789;123456789;ACME Inc. – West Campus;

For ClearReplica enabled ClearTrigger, a key would look similar to the three (3) examples below:

cleartrigger_M ABS licman;123456789;000000000;ACME Inc – East Campus;
cleartrigger_M ABS licman;000000000;123456789;ACME Inc.;
cleartrigger_lite_M ABS licman;123456789;123456789;ACME Inc. – West Campus;

Note: Only a vendor-generated license will allow ClearTrigger to operate without randomly displaying the *ABS Splash Message* on "development" VOBs.



Note: For "One-way" replication a valid license key for ClearTrigger (or ClearTrigger Lite) and ClearReplica is not required "at the receiver" in order for ClearReplica to work.

ClearReplica Regions

Before examining the ClearReplica in detail, it is prudent to spend time defining what constitutes a ClearReplica Region and discussing the different types or "replication" offered.

A **ClearReplica Region** is the group of VOBs that are replicated within the associated ClearTrigger Region. ClearTrigger regions can contain a single VOB or contain hundreds of VOBs. For more on ClearTrigger regions refer to **ClearTrigger Administration documentation.**

ClearReplica "replication" can be either "one-way" or "two-way" replication. An example of **One-way Replication** is when site "A" pushes changes to site "B" and does not push changes back. This is usually the case when site "B" is a "backup" or "distribution only" site for site "A". **Two-way Replication** is when both site "push" their changes to the other site, as is the case with co-developed software projects (classic geographically-distributed development).

ClearReplica "replication" can be configured for "on-site" or "off-site" replication. An example of ***off-site*** replication is when site "A" (Atlanta) pushes changes to site "B" (Japan). This is useful for off-site backups or off-site development. ***On-site*** replication is when you site "A" (Atlanta) pushes changes from VOB A to another VOB B in the same region for backup, distribution or co-development purposes (i.e. /vobs/product -> /vobs/backup_product or /vobs/web_dev -> vobs/production_web).

ClearReplica **Deployment** can also be used to "deploy" directly from a **ClearCase VOBs** to a regular **flat file system** on either local machines or remote machines around the world. The machines deployed to do not even need to have ClearCase installed.

ClearReplica can also "replicate" or "deploy" the ***whole*** VOB, just a ***portion*** of the VOB, just a ***single file*** or select files, or just selected branches. The destination of the replicated VOB data can be a VOB of the same name or a VOB of a different name. Therefore ***portions*** of one or more VOBs can be "replicated" to a ***single*** VOB and a single VOB can have a portion of its data "replicated" to several VOBs in a region.

ClearReplica Matrix	One-way Replication						Two-way Replication					
	Whole	Portion	Single File	Select Branches	Select Attributes	Select Labels	Whole	Portion	Single File	Select Branches	Select Attributes	Select Labels
"off-site" Replication	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑
"on-site" Replication	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑
"off-site" Deployment	☑	☑	☑	☑	☑	☑	Deployment is by definition One-way					
"on-site" Deployment	☑	☑	☑	☑	☑	☑						

ClearCase Multisite Matrix	One-way Replication						Two-way Replication					
	Whole	Portion	Single File	Select Branches	Select Attributes	Select Labels	Whole	Portion	Single File	Select Branches	Select Attributes	Select Labels
"off-site" Replication	☑	☐	☐	☐	☐	☐	☑	☐	☐	☐	☐	☐
"on-site" Replication	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
"off-site" Deployment	☐	☐	☐	☐	☐	☐	Deployment is by definition One-way					
"on-site" Deployment	☐	☐	☐	☐	☐	☐						

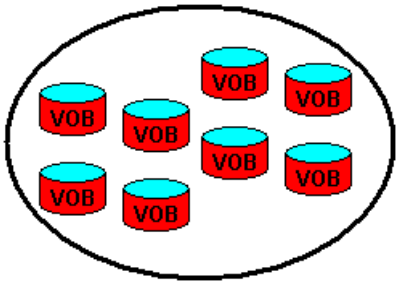
☑ **= Supported**
☐ **= NOT supported**

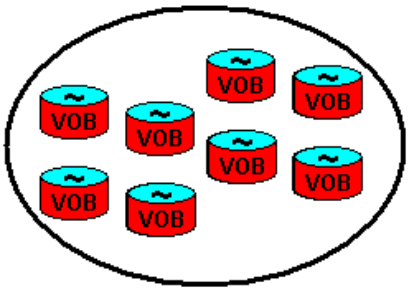
One Way Replication

ClearReplica one-way replication is when a site "A" sends ClearReplica packets to site "B" while site "B" does not sent packets to site "A". This is usually the case when site "B" is a "backup" or "distribution only" location, but can also be the case when distributed development in cases where site "A" develops code for "B" and therefore "B" needs the changes performed by "A", but "A" does not need the changes of 'B".

ClearReplica always requires a license on the "development" site (where actual checkins, checkouts, etc. are performed), but does not require one on "backup" sites. Therefore if you "replicate" 100 VOB form site "A" to site "B" and the "replicas" stored in "B" are not developed on, but exists for "backup only" or "distribution only" purposes then no license of ClearReplica is required there.

When referring to a region of VOBs where development takes place this document uses this pictorial to represent them.



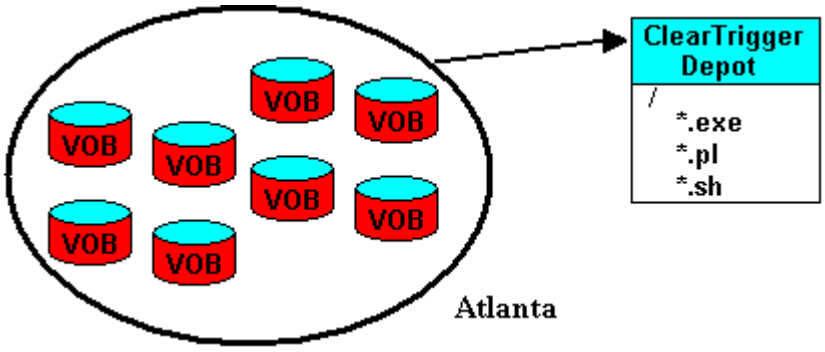


When referring to a region of VOBs where **NO** development takes place this document uses this pictorial to represent them.

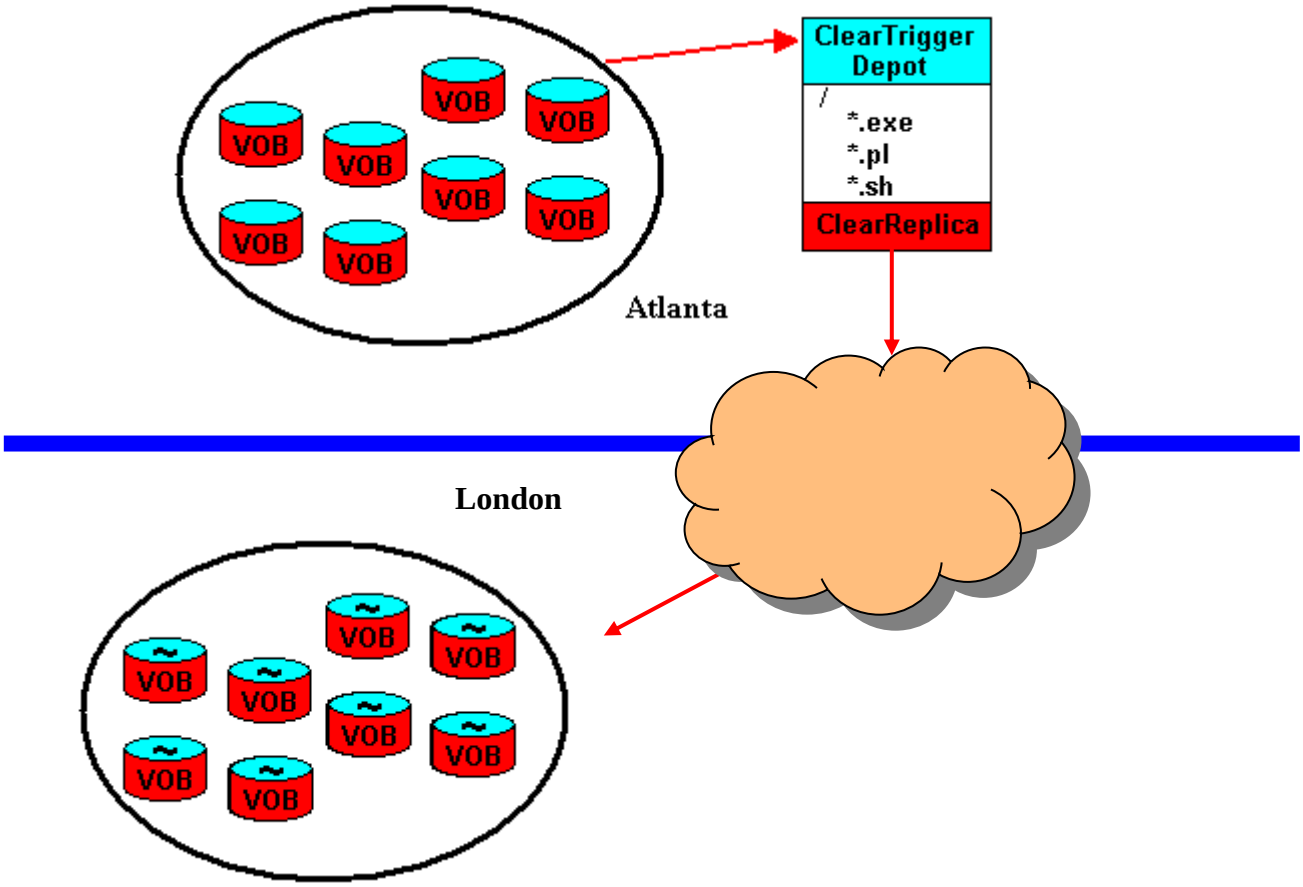
The following sub-sections describe some "***one way***" replication scenarios and how it effects ClearReplica licensing. **All require only one (1) license of ClearReplica and ClearTrigger**.

One ClearTrigger Region – One ClearReplica region - Offsite Replication

All of your VOBs use a single ***clearbits_file*** (which points to a *policy depot*) creating **one** ClearTrigger Region. All VOBs in this *ClearTrigger Region* can now have the same policy implemented. **This would require only one license of ClearTrigger.**

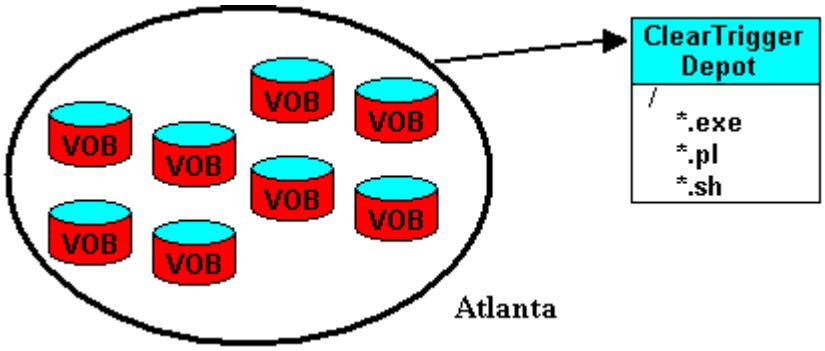


To "replicate" all of the VOBs to an offsite "disaster recovery" location or "backup" location where VOBs are accessible, but no development is performed (i.e. distribution, staging, backup VOBs), **only one additional license of ClearReplica is required**.

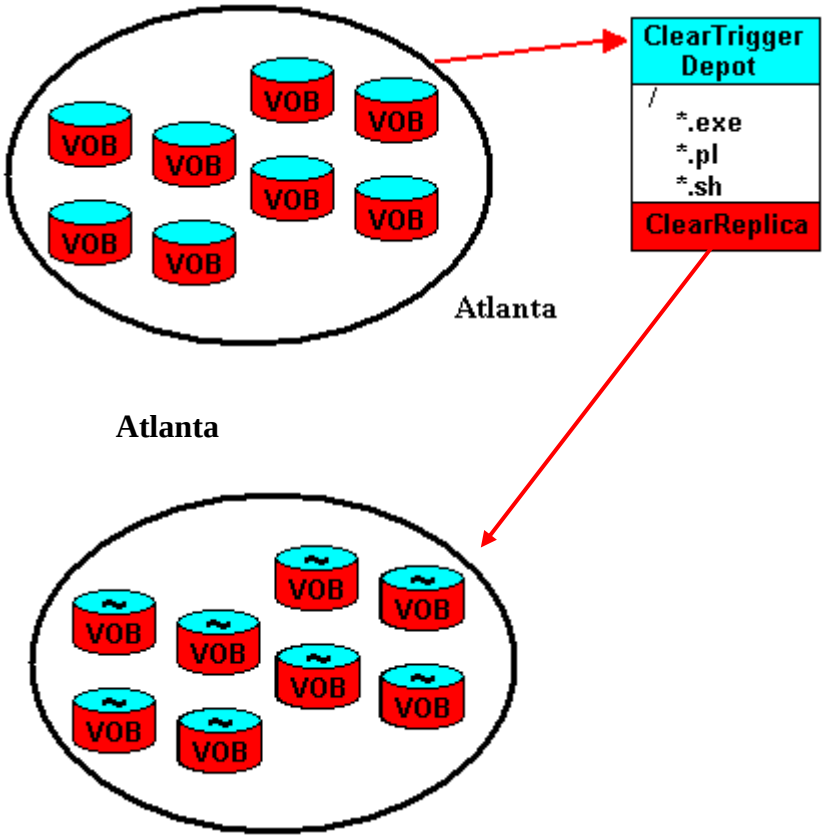


One ClearTrigger Region – One ClearReplica region - Onsite Replication

All of your VOBs use a single ***clearbits_file*** (which points to a *policy depot*) creating **one** ClearTrigger Region. All VOBs in this *ClearTrigger Region* can now have the same policy implemented. **This would require only one license of ClearTrigger.**

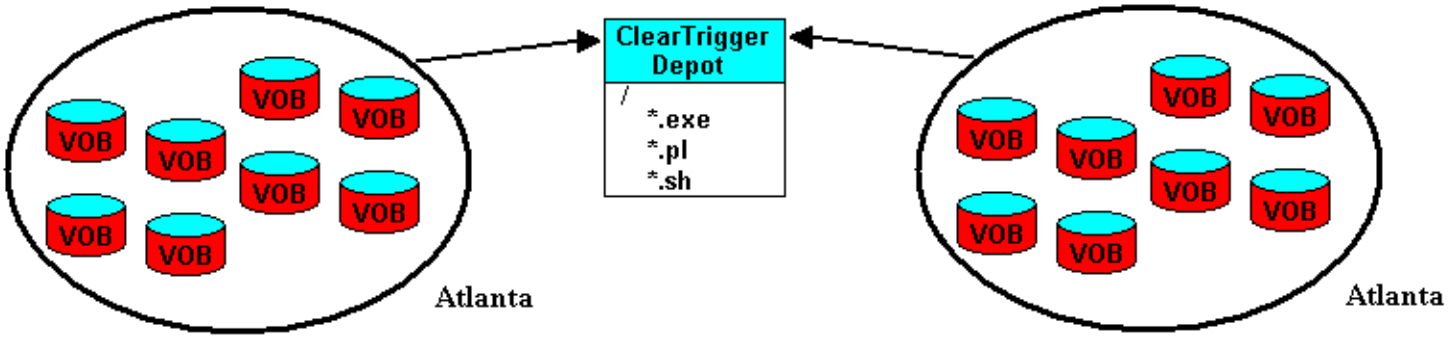


To "replicate" all of the VOBs to an offsite "disaster recovery" location or "backup" location where VOBs are accessible, but no development is performed (i.e. distribution, staging, backup VOBs), **only one additional license of ClearReplica is required**.

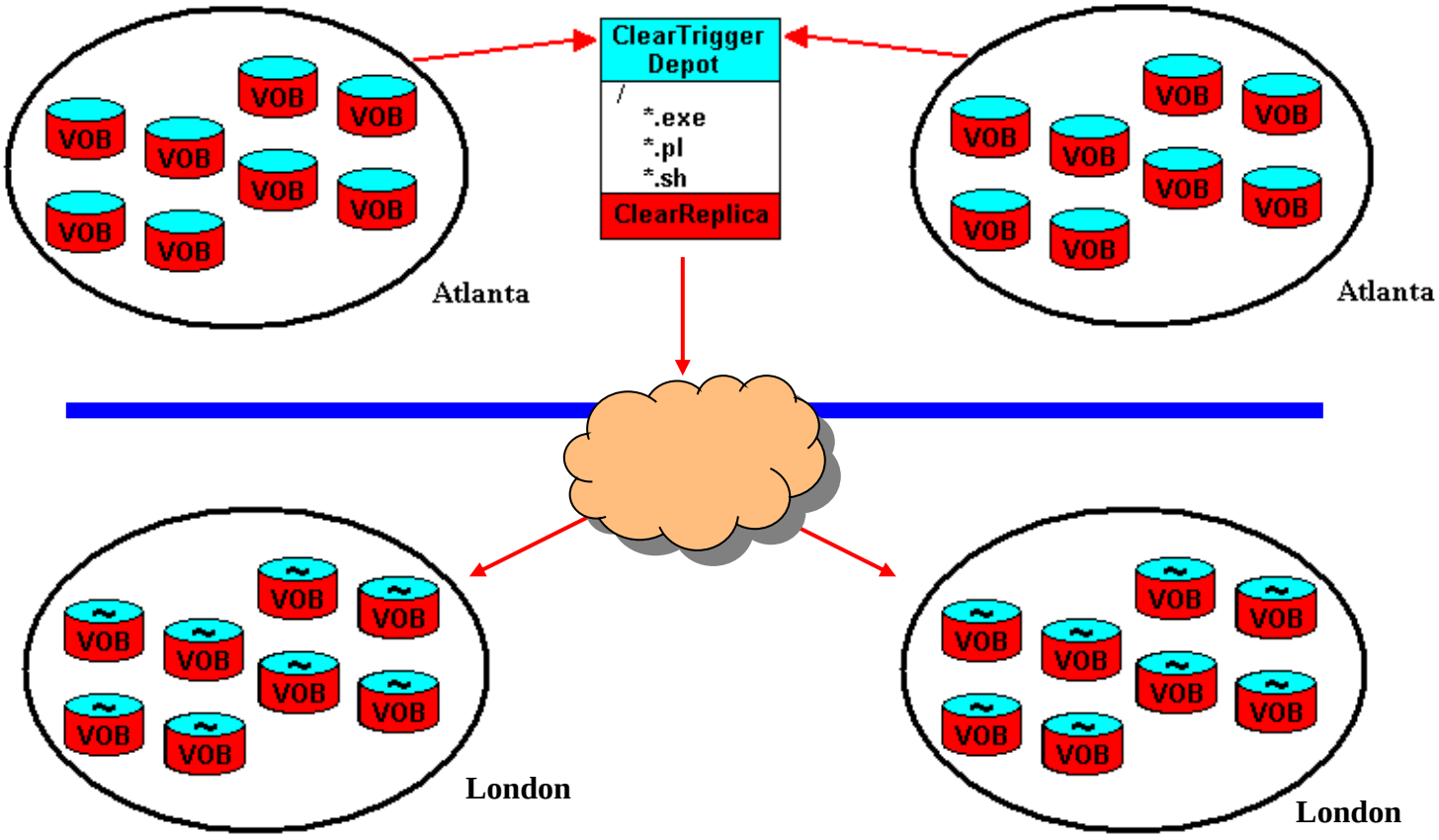


Multiple ClearTrigger Regions – One ClearReplica region – Offsite Replication

For company policy reasons you might have two (or more) ClearTrigger policy regions that use multiple ***clearbits_files*** (which points to a single *policy depot*) creating **two** ClearTrigger Regions. VOBs in the different *ClearTrigger Region* have different policy (as defined by the clearbits_file). **This would require only one license of ClearTrigger.**

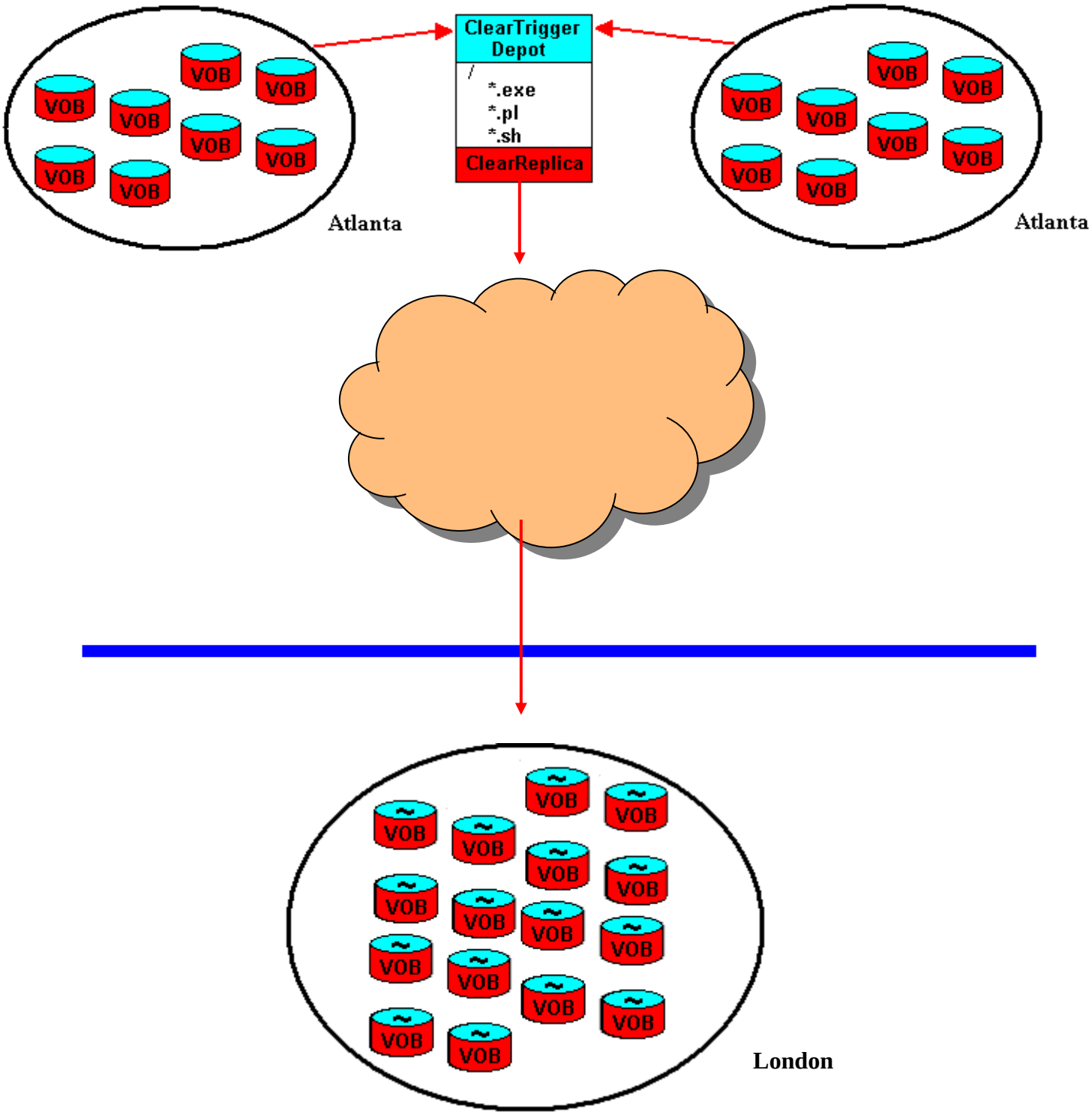


To "replicate" all of the VOBs from both regions to offsite "disaster recovery" locations or "backup" locations where VOBs are accessible, but no development is performed (i.e. distribution, staging, backup VOBs), **only one additional license of ClearReplica is required**.



With ClearReplica you can even combine the regions at the receiver.

To “replicate” all of the VOBs from both regions to an offsite “disaster recovery” location or “backup” location where VOB are accessible, but no development is performed there (i.e. distribution, staging, backup VOBs), **only one additional license of ClearReplica is required**.



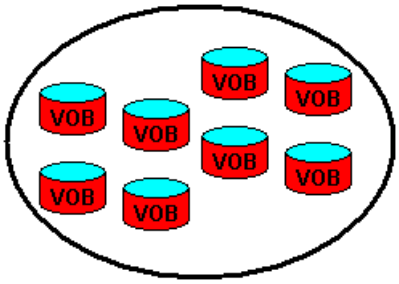
Two Way Replication

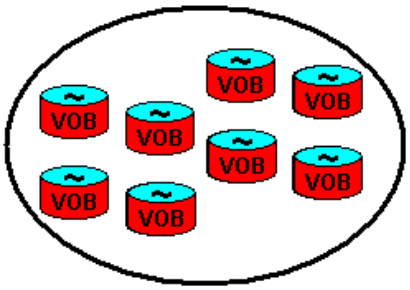
All of your VOBs use a single ***clearbits_file*** (which points to a *policy depot*) creating **one** ClearTrigger Region. All VOBs in this *ClearTrigger Region* can now have

ClearReplica two-way replication is when a site "A" sends ClearReplica packets to site "B" and site "B" sends packets to site "A". This is usually the case when site "A" and "B" are co-developing software.

ClearReplica always requires a license on the "development" site (where actual checkins, checkouts, etc. are performed), which in this case is both sites. Both sites will require a license of ClearReplica and ClearTrigger.

When referring to a region of VOBs where development takes place this document uses this pictorial to represent them.



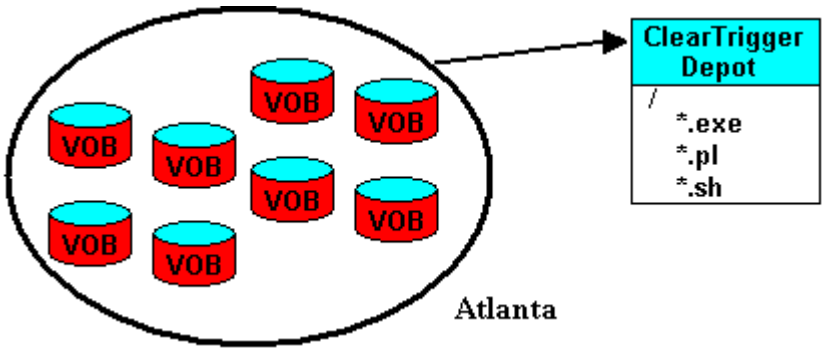


When referring to a region of VOBs where **NO** development takes place this document uses this pictorial to represent them.

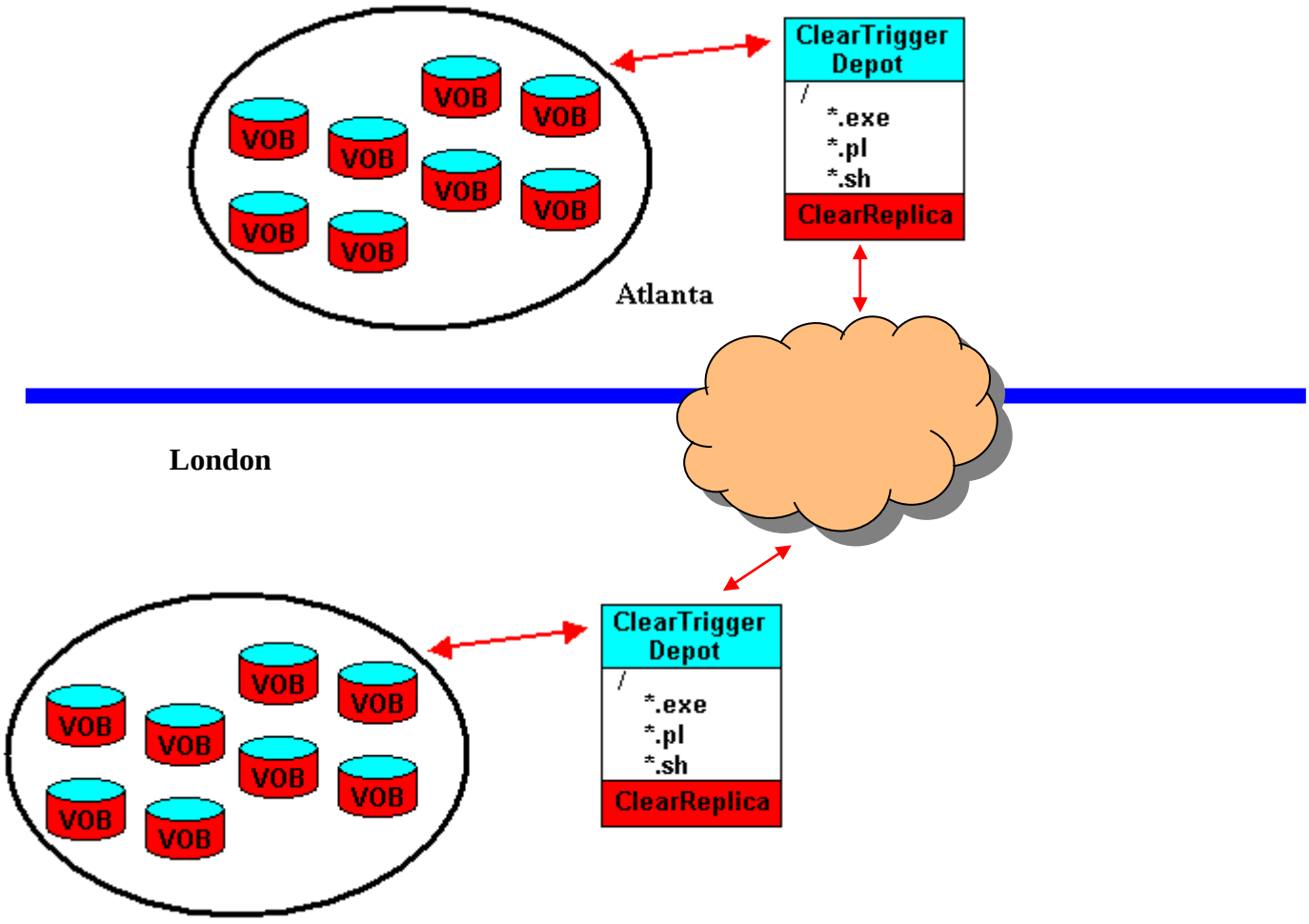
The following sub-sections describe some "***two way***" replication scenarios and how it effects ClearReplica licensing. **All require only two (2) licenses of ClearReplica and ClearTrigger**.

One ClearTrigger Region – One ClearReplica region - Offsite Replication

All of your VOBs use a single ***clearbits_file*** (which points to a *policy depot*) creating **one** ClearTrigger Region. All VOBs in this *ClearTrigger Region* can now have the same policy implemented. **This would require only one license of ClearTrigger.**

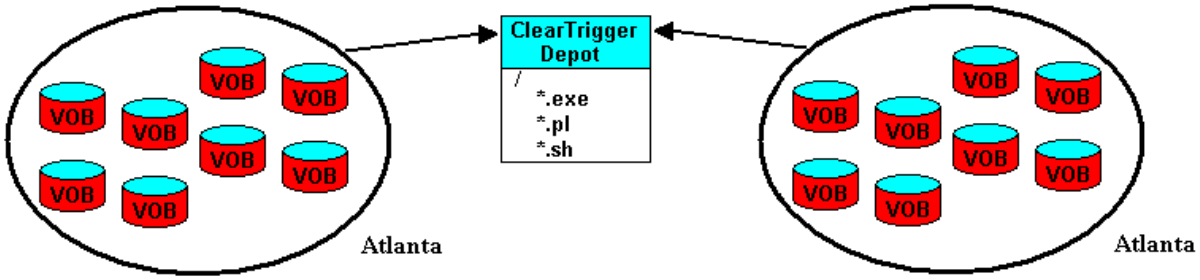


To "replicate" all of the VOBs to an offsite "development" location where VOBs are accessible and development is performed (i.e. overseas development or any geographically disjoint development), **only one additional license of ClearTrigger and one of ClearReplica is required**.

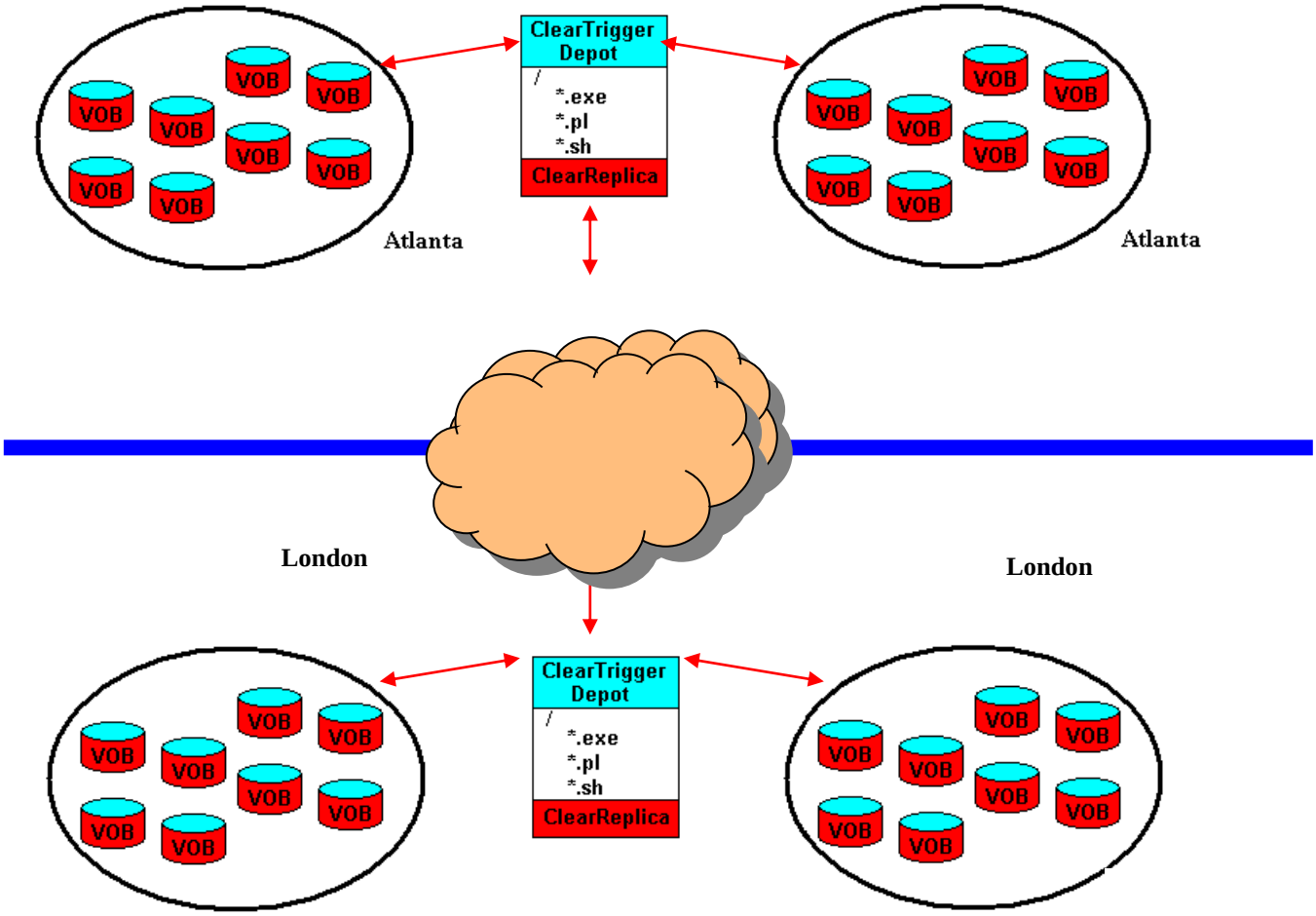


Multiple ClearTrigger Regions – One ClearReplica region – Offsite Replication

For company policy reasons you might have two (or more) ClearTrigger policy regions that use multiple ***clearbits_files*** (which points to a single *policy depot*) creating **two** ClearTrigger Regions. VOBs in the different *ClearTrigger Region* have different policy (as defined by the clearbits_file). **This would require only one license of ClearTrigger.**

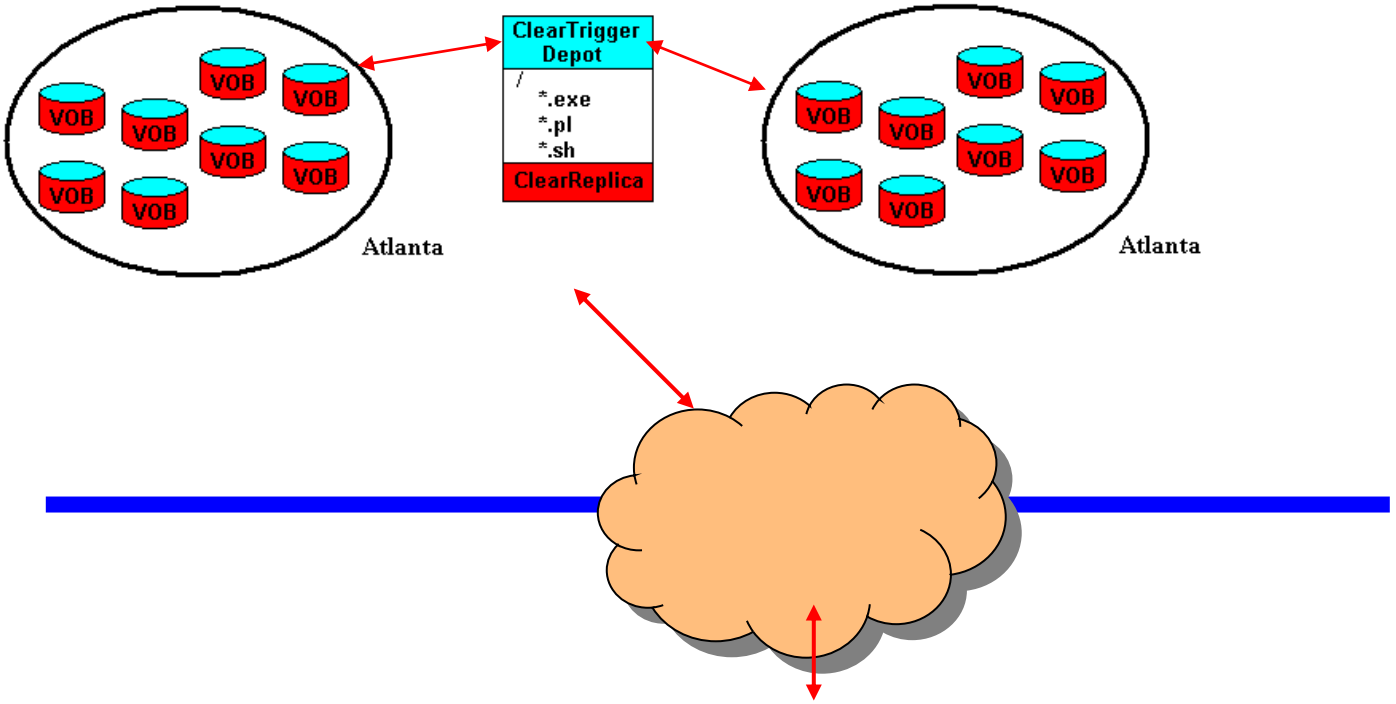


To "replicate" all of the VOBs from both regions to an offsite "development" location where VOBs are accessible and development is performed (i.e. overseas development or any geographically disjoint development), **only one additional license of ClearTrigger and two of ClearReplica are required**.

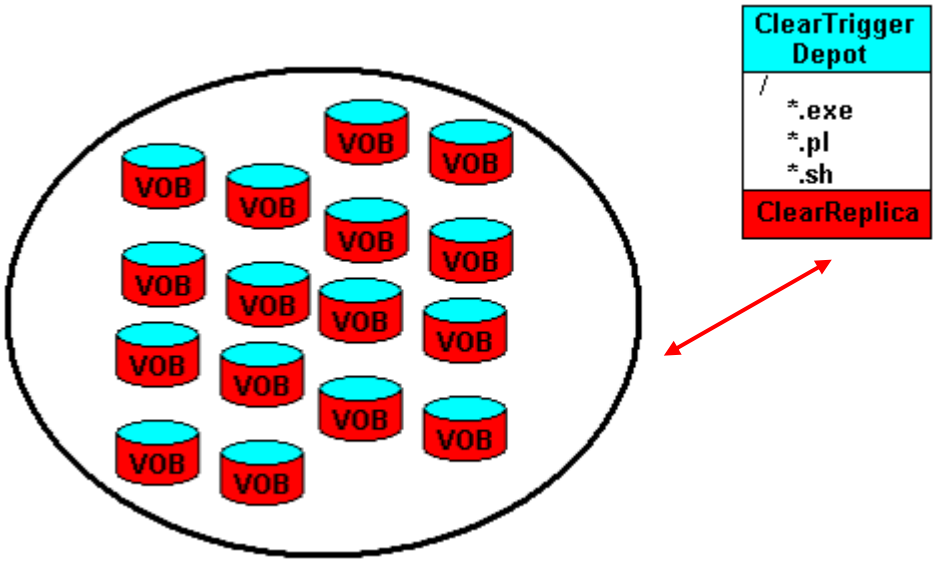


With ClearReplica you can even combine the regions at the receiver.

To "replicate" all of the VOBs from both regions to an offsite "development" location where VOBs are accessible and development is performed (i.e. overseas development or any geographically disjoint development), **only one additional license of ClearTrigger and one of ClearReplica is required**



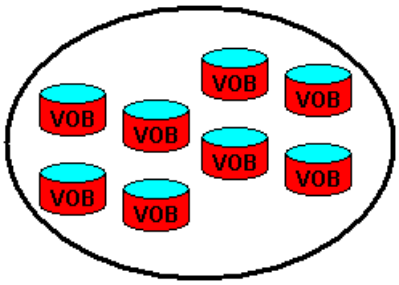
London

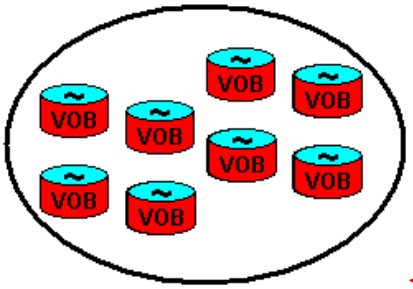


ClearReplica Deployment Regions

ClearReplica Deployment is when a site "A" sends ClearReplica packets to site "B" that are to be copied to a normal flat file system on machines in site "B". The "deployed to" sites/machines do not need to have ClearCase installed and do not require a ClearReplica license. A licensed ClearReplica site can deploy to ***unlimited*** "deployment sites/machines" both ***locally*** and ***remotely***.

When referring to a region of VOBs where development takes place this document uses this pictorial to represent them.





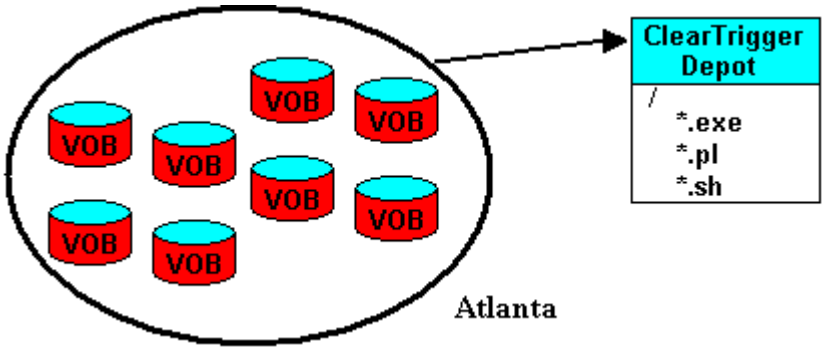
When referring to a region of VOBs where **NO** development takes place this
document uses this pictorial to represent them.

When referring to a grouping of flat file systems on machines (both ClearCase and Non-ClearCase) that are the destination of ClearReplica Deployments place this document uses pictorials like these to represent them.

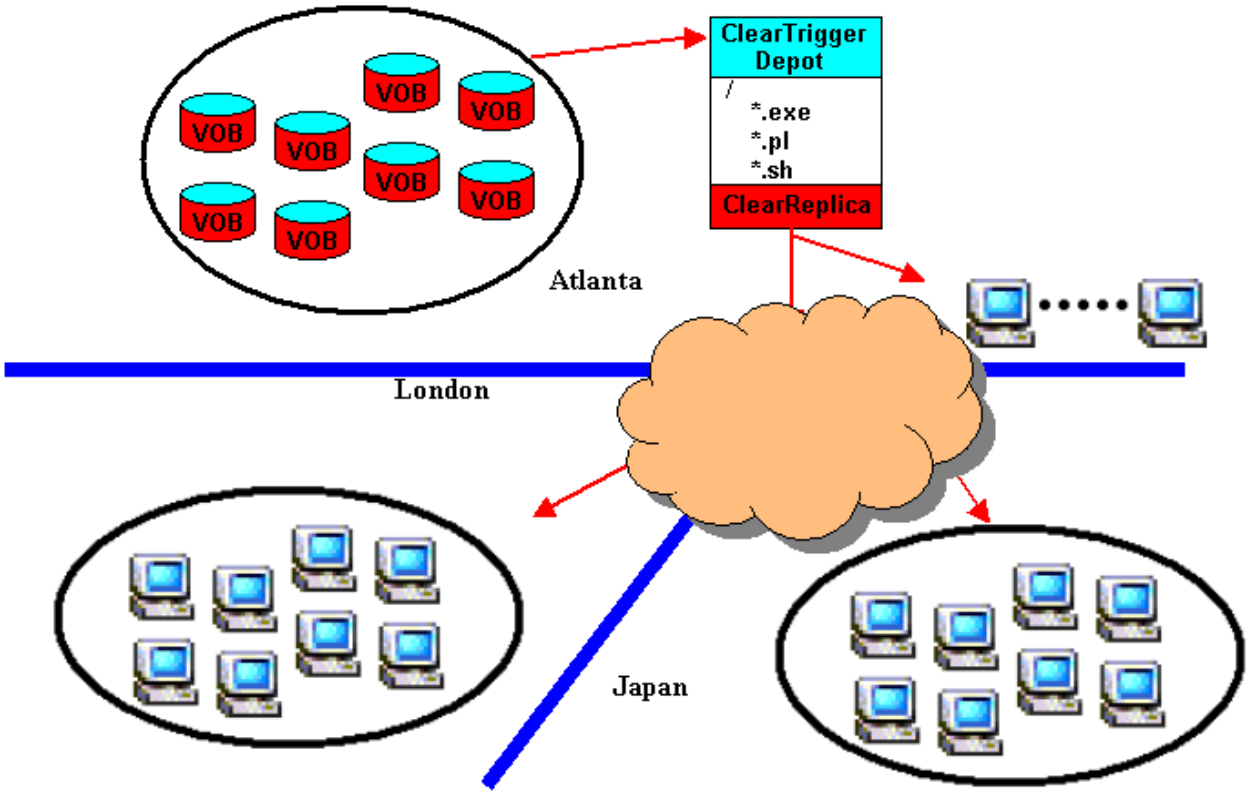
The following sub-sections describe a "***deployment***" scenario and how it effects ClearReplica licensing. **This scenario, though it deploys from many VOBs to many machines requires only one (1) license of ClearReplica and ClearTrigger.**

One ClearTrigger Region – One ClearReplica region – Onsite/Offsite Deployment

All of your VOBs use a single ***clearbits_file*** (which points to a *policy depot*) creating **one** ClearTrigger Region. All VOBs in this *ClearTrigger Region* can now have the same policy implemented. **This would require only one license of ClearTrigger.**



To "Deploy" any VOB (or portion of the VOB) to either onsite or offsite machines (even those without ClearCase installed; **only one additional license of ClearReplica is required**.



ClearReplica Configuration Files

Most ClearReplica Configuration information is contained within one of two file types; the **shipper.conf** file contains **ClearReplica_shipper** configuration data and the **receiver.conf** file(s) contain the **ClearReplica_receiver** configuration data. All instances of both files types are contained in the "**m_bay/config**" directory in the associated ClearTrigger depot directory. You only need to configure the appropriate file type if your site runs the associated service (i.e. only configure the **shipper.conf** file if you are running **ClearReplica_shipper** processes).

shipper.conf (changed in 12.1)

The **shipper.conf** file should exist in the "m_bay/config" directory of the associated ClearTrigger depot and is accessed by the **ClearReplica_shipper**. The shipper reads its **shipper configuration parameters** from the shipper.conf file as well as optional additional **class_definitions** and the **replication_definitions** that define what VOB data is actually replicated and to where. The **shipper configuration parameters** are defined below. **Items new to or changed in 12.0 or later are marked ***.

Parameters	Purpose
view	Names the outgoing replication view that should be dedicated to the shipping process
location_name	Names the location for the site
UTC_offset	Defines the difference between GMT and this time zone for <location_name>.
MVFS_DRIVE (windows only)	Names the MVFS Drive letter for the machine running the shipper process. For UNIX this variable-value pair is ignored.
duration	Number of minutes, seconds or hours that the shipper waits after processing the last replication definition in the list to again start processing the available list. If the parameter "**duration_scale**" is set to "minute" or omitted then the duration parameter can range from [0...1440] (number of minutes in 1 day). If the parameter "duration_scale is set to "hour" then the duration parameter can range from [0…24]. If the parameter "duration_scale is set to "second" then the duration parameter can range from [0…60]. A values of '0' is always interpreted as "immediately" regardless of the duration_scale parameter value.
duration_scale (optional)	From the values ["second", "minute", "hour"] with a default of "minute" if omitted. This parameter determines what the parameter "duration" refers to in its count.
transport_program **(*changed in 12.1)**	Names the program (usually ftp) that the shipper uses to transport the packet to the receiver. **This program MUST be in the PATH of the process running the "shipper".** Packets are compressed and encrypted and the passwords are encrypted. To use one of the different flavors of secure FTP (sftp) refer to the section entitled **Using Secure FTP as the transport program**.
shipper_id	Names the **userid** that is allowed to run the shipper for this associated depot. This userid should have read access to all VOBs that it will replicate/ship and read access to the associated shipper view.
pre_ship_trigger (optional)	Optional script or executable that is called **immediately before** any requested shipping packet is created. If the script exits with a zero return code then the packet is created and immediately shipped (unless it is destined to the **_hold_out** directory). If the script exits with a non-zero (failed) status then the packet is not built and the appropriate ***error*** message is written to the ClearReplica **shipper logs**. This script or program must reside in the ***m_bay/_triggers*** directory in the associated ClearTrigger depot.
post_ship_trigger (optional)	Optional script or executable that is called **immediately after** any requested shipping packet is created and shipped (or created and placed in the **_hold_out** directory), even if the packet did not process successfully (providing an opportunity to clean up any data changes made in the **pre_ship_trigger_script**). If the script exits with a non-zero (failed) status then the appropriate ***warning*** message is written to the ClearReplica **shipper logs**.. This script or program must reside in the ***m_bay/_triggers*** directory in the associated ClearTrigger depot.
startup_trigger (optional)	Optional script or executable that is called **immediately after** the successful startup of a ClearReplica shipper process. If the script exits with a zero return code then the shipper process continues. If the script exits with a non-zero (failed) status then the shipper startup is considered denied by that script and the appropriate ***error*** message is written to the ClearReplica **shipper logs**. This script or program must reside in the ***m_bay/_triggers*** directory in the associated ClearTrigger depot.

shutdown_trigger (optional)	Optional script or executable that is called **immediately after** the successful shutdown of a ClearReplica shipper process. If called, an appropriate ***error*** or ***success*** message is written to the ClearReplica **shipper logs** indicating the script's return code. This script or program must reside in the ***m_bay/_triggers*** directory in the associated ClearTrigger depot.
encryption (optional)	From the values ["standard", "peer_to_peer"] with a default of "standard" if omitted. This parameter determines if **"Standard ClearReplica Encryption"** is used or if an additional **Peer_To_Peer ClearReplica Encryption** is used. If this value is set to "peer_to_peer" then the ClearReplica shipper process uses the encryption key stored in the shippers associated **encryption key file**. The encryption key file must reside in the ***m_bay/config/keys*** directory in the associated ClearTrigger depot.
slink_scale (optional)	From the values ["0"... "3"] with a default of "1" if omitted. This optional parameter determines the amount of memory allocated to symbolic link processing when packets are created. Only add/modify if processing a VOB with an exceptional number of symbolic links (more than 10,000 symbolic links) and under the direction of ABS.
postp_scale (optional)	From the values ["1"... "3"] with a default of "1" if omitted. This optional parameter determines the amount of memory allocated to **rmelem** and **mklabel** processing link processing when packets are created. Only add/modify if suggested by ClearReplica shipper logs and under the direction of ABS.

An example of a set of Windows shipper configuration parameters follows:

```
view=shipper_view
location_name=atlanta
UTC_offset=-05:00
MVFS_DRIVE=m
duration=5
transport_program=ftp
encryption=standard
shipper_id=vobadm
pre_ship_trigger=
startup_trigger=shipper_startup.bat
shutdown_trigger=notify_shutdown.bat
```

These values are read once when the associated **ClearReplica_shipper** process is started so changes to these values require a restart of the associated "shipper" process in order to take effect. The file also contains optional additional **class_definitions** as well as one or more **replication_definitions**, which can be added at any time, the addition or modification of **class_definitions** or **replication_definitions** take effect immediately and do not require a restart.

An example of the resulting output placed in the appropriate **shipper logs** as depicted below:

```
################################################
[Mon Oct 19 13:58:45 2015 UTC -05:00] == service (clearreplica_shipper 12.0)
started (to run always) with static variables: ==
       == (required variables) ===================
          MVFS_DRIVE                        = m
          view                              = shipper_view
          location_name                     = atlanta
          UTC_offset                        = -05:00
          shipper_id                        = cclarke
          duration                          = 5
       == (optional variables) ===================
          transport_program                 = ftp
          pre_ship_trigger                  = (none)
          post_ship_trigger                 = (none)
          startup_trigger                   = shipper_startup.bat
          shutdown_trigger                  = notify_shutdown.bat
          encryption                        = standard
          duration_scale                    = minute
          slink_scale                       = 0
          postp_scale                       = 1
          shipper_class                     = (none)
          Shipper Class 0                   = (disabled)
          Shipper Class 1                   = (disabled)
          Shipper Class 2                   = (disabled)
          Shipper Class 3                   = (disabled)
          Shipper Class 4                   = (disabled)
          Shipper Class 5                   = (disabled)
          Shipper Class 6                   = (disabled)
          Shipper Class 7                   = (disabled)
          Shipper Class 8                   = (disabled)
          Shipper Class 9                   = (disabled)
       ============================================================
################################################
```


receiver.conf (changed in 12.0)

The **receiver.conf** file should exist in the "m_bay/config" directory of the associated ClearTrigger depot and is accessed by the **ClearReplica_receiver**. The receiver reads its **receiver configuration parameters** from the associated receiver.conf file as well as one or more **replication_receiver_dirs** in that designate which directories will be "watched" for received packets by the "receiver" process.

The **receiver configuration parameters** are defined below. **Items new to or changed in 12.0 or later are marked** *.

Parameters	Purpose
view	Names the receiving replication view that should be dedicated to the receiving process
location_name	Names the location for the site
UTC_offset	Defines the difference between GMT and this time zone for <location_name>.
MVFS_DRIVE	Names the MVFS Drive letter for the machine running the shipper process. For UNIX this variable-value pair is ignored.
receiver_id	Names the **userid** that is allowed to run the receiver for this associated depot. This userid should have write access to all VOBs that it will replicate/receive and write access to the associated receiver view.
pre_receive_trigger (optional)	Optional script or executable that is called **immediately before** any received packet is processed. If the script exits with a zero return code then the packet is processed. If the script exits with a non-zero (failed) status then the packet is not processed and the appropriate ***error*** message is written to the ClearReplica **receiver logs**. This script or program must reside in the ***m_bay/_triggers*** directory in the associated ClearTrigger depot.
post_receive_trigger (optional)	Optional script or executable that is called **immediately after** any received packet is processed, even if the processed packet did not complete (providing an opportunity to clean up any data changes made in the **pre_receive_trigger_script**). If the script exits with a non-zero (failed) status then the appropriate ***warning*** message is written to the ClearReplica **receiver logs**. This script or program must reside in the ***m_bay/_triggers*** directory in the associated ClearTrigger depot.
new_dir_personality (optional)	Optional parameter with the value of either **"main"** or **"branch"** that determines if newly created directory version are created on the /main branch ("main") or as per the received Config. Spec. sent by the shipping site ("branch"). **The default value is "main" if none is provided.**
startup_trigger (optional)	Optional script or executable that is called **immediately after** the successful startup of a ClearReplica receiver process. If the script exits with a zero return code then the receiver process continues. If the script exits with a non-zero (failed) status then the receiver startup is considered denied by that script and the appropriate ***error*** message is written to the ClearReplica **receiver logs**. This script or program must reside in the ***m_bay/_triggers*** directory in the associated ClearTrigger depot.
shutdown_trigger (optional)	Optional script or executable that is called **immediately after** the successful shutdown of a ClearReplica shipper process. If called, an appropriate ***error*** or ***success*** message is written to the ClearReplica **receiver logs** indicating the

	script's return code. This script or program must reside in the ***m_bay/_triggers*** directory in the associated ClearTrigger depot.
scheduled_inactivity (optional)	Defines a well-known time that the receiver will ***NOT*** process packets. During this optionally defined **scheduled inactivity duration**, packets are still received from remote locations, but processing of those packets is postponed until outside the duration. If defined, two (2) unique times must be provided of the form: **HH:MM-HH:MM** Where each HH is '00'-'23' and each MM is '00'-'59'. The first HH:MM is the "**start of the inactivity time**" designator and the second HH:MM is the "**end of the inactivity time**" designator.
encryption (optional)	From the values ["standard", "peer_to_peer"] with a default of "standard" if omitted. This parameter determines if **"Standard ClearReplica Encryption"** is used or if an additional **Peer_To_Peer ClearReplica Encryption** is used. If this value is set to "peer_to_peer" then the ClearReplica shipper process uses the encryption key stored in the shippers associated **encryption key file**. The encryption key file must reside in the ***m_bay/config/keys*** directory in the associated ClearTrigger depot.
ignore_received_slinks (optional)	Optional parameter with the value of either **"yes"** or **"no"** that determines if newly created symbolic links are created if included in received packets. **The default value is "no" if none is provided.**
ignore_received_label_data (optional)	Optional parameter with the value of either **"yes"** or **"no"** that determines if newly received label data is replicated to ignored by the receiver if included in received packets. **The default value is "no" if none is provided.**
ignore_received_branch_data (optional) **(*added in 12.0)**	Optional parameter with the value of either **"yes"** or **"no"** that determines if newly received label data is replicated to ignored by the receiver if included in received packets. **The default value is "no" if none is provided.**
ignore_received_attribute_data (optional) **(*added in 12.0)**	Optional parameter with the value of either **"yes"** or **"no"** that determines if newly received label data is replicated to ignored by the receiver if included in received packets. **The default value is "no" if none is provided.**
suppress_locked_label_checking (optional)	Optional parameter with the value of either **"yes"** or **"no"** that determines if newly received label types are tested for being locked and written to the receiver logs upon packet receipt for packets containing labels that are locked in the receiver. **The default value is "no" if none is provided.**
suppress_locked_branch_checking (optional) **(*added in 12.0)**	Optional parameter with the value of either **"yes"** or **"no"** that determines if newly received branch types are tested for being locked and written to the receiver logs upon packet receipt for packets containing branches that are locked in the receiver. **The default value is "no" if none is provided.**
suppress_locked_attribute_checking (optional) **(*added in 12.0)**	Optional parameter with the value of either **"yes"** or **"no"** that determines if newly received attribute types are tested for being locked and written to the receiver logs upon packet receipt for packets containing attributes that are locked in the receiver. **The default value is "no" if none is provided.**

An example of a set of Windows receiver configuration parameters follows:

```
view=receiver_view
location_name=atlanta
UTC_offset=-05:00
MVFS_DRIVE=m
receiver_id=cclarke
pre_receive_trigger=
post_receive_trigger=email_errors.bat
startup_trigger=receiver_startup.bat
shutdown_trigger=notify_shutdown.bat
scheduled_inactivity=02:00-04:00
new_dir_personality=main
encryption=standard
ignore_received_slinks=no
ignore_received_label_data=no
ignore_received_branch_data=no
ignore_received_attribute_data=no
suppress_locked_label_checking=no
suppress_locked_branch_checking=no
suppress_locked_attribute_checking=no
```

These values are read once when the associated **ClearReplica_receiver** process is started so changes to these values require a restart of the associated "receiver" process in order to take effect. The file also contains **replication_receiver_dirs** **which** can be added or modified at any time, the addition or modification of **replication_receiver_dirs** take effect immediately and do not require a restart.

An example of the resulting output placed in the **receiver_all** or **receiver_start_stop** logs is depicted below:

```
#####################################################
[Mon Oct 19 13:59:34 2015 UTC -05:00] == service (clearreplica_receiver 12.0)
started (to run always) with static variables: ==
          == (required variables) ===================
               MVFS_DRIVE                          = m
               view                                = receiver_view
               location_name                       = atlanta
               UTC_offset                          = -05:00
               receiver_id                         = cclarke
          == (optional variables) ===================
               pre_receive_trigger                 = (none)
               post_receive_trigger                = email_errors.bat
               startup_trigger                     = receiver_startup.bat
               shutdown_trigger                    = notify_shutdown.bat
               scheduled_inactivity                = 02:00-04:00
               new_dir_personality                 = main
               encryption                          = standard
               ignore_received_slinks              = no
               ignore_received_label_data          = no
               ignore_received_branch_data         = no
               ignore_received_attribute_data      = no
               suppress_locked_label_checking      = no
               suppress_locked_branch_checking     = no
               suppress_locked_attribute_checking = no
          ========================================
#####################################################
```

Using the _hold_out directory to *manually* transport packets

The ClearReplica receiver process does not care how packets get to their associated receiver directories it is only concerned with processing the packets that show up in this directory. This transparency allows for ***infinite manual transport mechanisms*** in addition to FTP and SFTP that are supported for automatic transport by ClearReplica. When the "hold" option is used within a **replication_definitions** the replication packet that is created is not immediately sent by the ClearReplica shipper process. This packet is instead held in the **_hold_out** directory where it is expected to be transported by some other process. The process can be:

- ftp
- sftp
- sneaker-net (disk transfer)
- direct write to receiver
- rcp
- scp
- network transfer
- email
- or any other mechanism

This allow for replication between machines that do not have a direct (or any) network connection or even those with an intermittent connection or with clients that change I.P. Addresses. There are other considerations when choosing to perform a manual transport.

- If transporting to a receiving directory with a "currently running" ClearReplica Receiver process it important to make sure to transport all associated CRC files (**epoc.*.*.*.crc** and **epoc.*.*.*.crc_*) are** completely transported before their associated pack file (**epoc.*.*.*.pack**). This is critical because any active ClearReplica receiver process that sees the pack file will immediately process associated CRC files as it assumes them to be complete. The ClearReplica Shipper itself will not transport the pack file until all associated CRC files have been transported.

- After a successful transport and processing of packets (as determined by your manual transport) it is important for the operator to also manually update the associated epoc file (**epoc.*.*.*.epoc**) so that future packets created in the future by the shipper only contain changes since the last packet was sent.

NOTE: If your **_hold_out** directory of one ClearReplica region is also the "receiver" directory of another (or the same) ClearReplica region the clearreplica_receiver is actually your transport mechanism as the shipper packets are build directly into other regions receiving directory not actual "receiver transport" processes (ftp) are ***not*** needed to "receive" the packets.

Using Secure FTP as the transport program (changed in 12.1)

As stated in the previous section, the ClearReplica receiver process does not care how packets get to their associated receiver directories it is only concerned with processing the packets that show up in this directory. This transparency allows for ***infinite manual transport mechanisms*** for ClearReplica packets. In addition to the manual transport mechanisms there are two protocols, **FTP** and **SFTP**, which ClearReplica can use to support automatic transport of ClearReplica packets. Additionally, there are several different SFTP options to support the many SFTP products available.

For automatic transport of created ClearReplica packets ClearReplica uses one (1) of the five (5) classes of automatic transport program. To select a class, adjust the **transport_program** configuration item in the appropriate **shipper.conf** file so that it is equal to the "**transport pneumonic**" (as defined in the table below) and can be found in the **clearreplica_shipper** processes **$PATH** variable or contains the full path name (no spaces allowed) to the program that ends in the **transport pneumonic**. This in many cases involves renaming or making a renaming a copy of the transport program.

Transport Class	Transport Pneumonic	Example entries in a shipper.conf file
0	**ftp**	transport_program=**ftp** transport_program=c:\dir**ftp** transport_program=/usr/bin/**ftp**
1	**sftp**	transport_program=**sftp** transport_program=c:\dir**sftp** transport_program=/usr/bin/**sftp**
2	**psftpk**	transport_program=**psftpk** transport_program=c:\dir**psftpk** transport_program=/usr/bin//**psftpk**
3	**ssftp**	transport_program=**ssftpk** transport_program=c:\dir**ssftpk** transport_program=/usr/bin/**ssftpk**
4	**psftpp**	transport_program=**psftpp** transport_program=c:\dir**psftpp** transport_program=/usr/bin/**psftpp**
5[+]	**lftp**	transport_program=**lftp** transport_program=c:\dir**lftp** transport_program=/usr/bin/**lftp**

[+] Added in 12.1.

NOTE: No spaces are allowed in **transport_program** entries.

The classes of programs differ only slightly as per vendor specification of the different programs the classes were mapped around. The default class, **class 0** (**ftp**) is common and works for all standard known **FTP** programs. The other **SFTP** classes are defined so you can determine which **Transport Pneumonic** to use in your **transport_program** definition in the appropriate **shipper.conf** file. The differences themselves are in the command line argument supported and the "in-session" commands supported.

If you wish to use SFTP instead of FTP as the automatic transport mechanism then refer to the man page of the SFTP program to determine which **Transport Pneumonic** to use.

Transport Pneumonic	Command Line arguments supported	In-session Commands supported
ftp	ftp -in file	cd, lcd, put, get, del, quit, !, user, ascii, binary
sftp	sftp -B file user@machine	cd, lcd, put, get, del, quit, !
psftpk	psftpk -b file -i keyfile -be -batch user@machine	cd, lcd, put, get, del, quit, !
ssftp	ssftp -B file user@machine	cd, lcd, put, get, rm, quit, lrm
psftpp	psftpp -b file –pw password –be –batch user@machine	cd, lcd, put, get, del, quit, !
lftp[+]	lftp -f file	cd, lcd, put, get, rm, open, quit, !

[+] Added in 12.1.

NOTE: When evaluating if a command line argument is support it is important to remember that character case is important (i.e. "-b" is not equivalent to "-B").

Class 2 (psftpk) uses a public/private key pair to authenticate. The key file must exists in the **m_bay/config/hostmaps** directory and be named {location}.pk where location is the receiving destination pneumonic defined in the associated **replication_definition**, (i.e. for "atlanta.pk").

How the ClearReplica Receiver handles locked VOBs

The ClearReplica Receiver must write to VOBs during it processing of received packets. ClearCase itself will prevent the writing to Locked VOBs. To avoid conflict with backups or other process that require a locked VOB, the ClearReplica Receiver will skip over locked VOBs and retain the packet until the VOB becomes unlocked. Additionally ClearReplica writes the event to the ClearReplica **receiver logs (**specifically the **receiver_all** log**)** and periodically rewrites to the log until the VOB is unlocked similar to that depicted below:

```
... Skipped packet (D:\receiver\epoc.500.atl.chi.pack)for locked VOB (\a_vob)...
... Packet (and any other packets for this VOB) retained for retrying later...
```

As soon as the VOB becomes unlocked processing of the packet will begin.

NOTE: ClearReplica notes a VOB is locked when it is either:

- Locked for all users
- Locked for some users with the user id of the receiver process not excluded within the lock definition.

So if the ClearReplica receiver is running as the user "**vobadm**":

ClearCase VOB is …	ClearReplica Receiver will…
Unlocked	Process the VOB
locked for ALL users accept	Skip the VOB
Locked except for the users rcarter and cclarke	Skip the VOB
Locked except for the users **vobadm** and cclarke	Process the VOB

This functionality is provided when the ClearReplica Receiver is 1.0 or higher is processing a packet from a ClearReplica Shipper 10.0 or higher.

ClearReplica Receiver Scheduled Inactive Times

The ClearReplica Receiver itself can be configured to suspend packet processing during well-known times where such processing might interfere with another process that is important to the organizations.

One can create a **scheduled_inactivity** definition in the **receiver.conf** file, doping so defines a well-known time that the receiver will ***NOT*** process packets. During this optionally defined **scheduled inactivity duration**, packets are still received from remote locations, but processing of those packets is postponed until outside the duration. Zero-filled military time is used.

If defined, two (2) unique times must be provided of the form:

```
scheduled_inactivity=HH:MM-HH:MM
```

Where each HH is '00'-'23' and each MM is '00'-'59'. The first HH:MM is the "**start of the inactivity time**" designator and the second HH:MM is the "**end of the inactivity time**" designator.

The ClearReplica Receiver will not process any newly received packets during the defined time. The packets are saved and process as soon as the current time is outside of the **scheduled inactivity duration**.

For example with a **scheduled inactivity definition** defined in the **receiver.conf** file like so:

```
scheduled_inactivity=02:00-03:30
```

The ClearReplica Receiver will delay the processing of received packets starting at 2:00am.

Additionally, ClearReplica writes the event to the ClearReplica **receiver logs** (specifically the **receiver_all** log) similar to line depicted below:

```
... Entering scheduled inactivity state.
    Packets receive, but not process during 02:00-03:30 timeframe
```

As soon as the inactivity period is over the event is written to the **receiver_all** **receiver log** and the processing on any received packets resumes.

```
... Leaving scheduled inactivity state as now outside of 02:00-03:30 timeframe.
    Received packets will now be processed
```

NOTE: The ***first*** time provide is when the inactivity period starts and the ***second*** is when the period ends. So a definition like **scheduled_inactivity=01:00-03:00** defines a 2-hour period starting at 1:00 a.m.  , while a definition like **scheduled_inactivity=03:00-01:00** defines a 22-hour period starting at 3:00 a.m.  .

Handling Cleartool Symbolic Links

The ClearReplica Shipper processes can be configured to send and package cleartool symbolic link information to remote locations or the shipper process can be configured to ignore this information completely. Additionally, the ClearReplica Receiver processes can be configured to process any symbolic link information it receives or ignore that symbolic link information altogether.

Shipper-side Symbolic Link Processing

One can configure the ClearReplica **shipper.conf** file to either package and replicate cleartool symbolic link data from the source location to the destination location or ignore this cleartool symbolic link data altogether. The **slink_scale** shipper configuration parameter is set to "1" (the default) if not defined which tells the shipper process to package cleartool symbolic link information when building packets to send to remote locations. The **slink_scale** parameter can be set to "0" which will cause the shipper process to ignore all symbolic link processing thus not replicating any symbolic link information to remote locations.

Receiver-side Symbolic Link Processing

One can configure the ClearReplica **receiver.conf** file to either process cleartool symbolic link data received from the source location or ignore this cleartool symbolic link data altogether. The **ignore_received_slinks** receiver configuration parameter is set to "no" (the default) if not defined which tells the receiver process to process any cleartool symbolic link information it encounters when processing packets it receives from remote locations. The **ignore_received_slinks** parameter can be set to "yes" which will cause the receiver process to ignore all symbolic link data it receives from remote locations.

Ignoring received Type Data (changed in 12.0)

The ClearReplica Receiver processes can be configured to process or ignore certain metadata **type** or **instance** data. Specifically, the receiver can be configured to ignore **branch** type:instance, **label** type:instance and **attribute** type:instance data.

One can request that the ClearReplica receiver ignore **label type** and **instance** data which will cause the receiver to ignore any label information passed to it in a received packet. The receiver will ignore any all requests for:

- mklbtype, modlbtype or rmtype lbtype commands
- mklabel or rmlabel commands
- lock or unlock of label types
- any label information associated with processed versions

You can read more on how to manipulate this behavior in the sub-section entitled **Receiver-side Ignoring Received Branch Data** within this section.

One can request that the ClearReplica receiver ignore **branch type** and **instance** data which will cause the receiver to ignore any branch information passed to it in a received packet. The receiver will ignore any all requests for:

- mkbrtype, modbrtype or rmtype brtype commands
- mkbranch or rmbranch commands
- lock or unlock of branch types
- any branch (other than /main) information associated with processed versions

You can read more on how to manipulate this behavior in the sub-section entitled **Receiver-side Ignoring Received Branch Data** within this section.

One can request that the ClearReplica receiver ignore **attribute type** and **instance** data which will cause the receiver to ignore any attribute information passed to it in a received packet. The receiver will ignore any all requests for:

- mkattype, modattype or rmtype attype commands
- mkattribute or rmattributecommands
- lock or unlock of attribute types
- any attribute information associated with processed versions

You can read more on how to manipulate this behavior in the sub-section entitled **Receiver-side Ignoring Received Attribute Data** within this section.

Note: For each of these options the default if not provided is to NOT ignore the associated request.

Receiver-side Ignoring Received Label Data

One can configure the ClearReplica **receiver.conf** file to either process cleartool label data received from the source location or ignore this cleartool label data altogether. The **ignore_received_label_data** receiver configuration parameter is set to "no" (the default) if not defined which tells the receiver process to process any cleartool label information it encounters when processing packets it receives from remote locations. The **ignore_received_label_data** parameter can be set to "yes" which will cause the receiver process to ignore all label data it receives from remote locations.

Receiver-side Ignoring Received Branch Data (new to 12.0)

One can configure the ClearReplica **receiver.conf** file to either process cleartool branch data received from the source location or ignore this cleartool branch data altogether. The **ignore_received_branch_data** receiver configuration parameter is set to "no" (the default) if not defined which tells the receiver process to process any cleartool branch information it encounters when processing packets it receives from remote locations. The **ignore_received_branch_data** parameter can be set to "yes" which will cause the receiver process to ignore all branch data it receives from remote locations.

Receiver-side Ignoring Received Attribute Data (new to 12.0)

One can configure the ClearReplica **receiver.conf** file to either process cleartool attribute data received from the source location or ignore this cleartool attribute data altogether. The **ignore_received_attribute_data** receiver configuration parameter is set to "no" (the default) if not defined which tells the receiver process to process any cleartool attribute information it encounters when processing packets it receives from remote locations. The **ignore_received_attribute_data** parameter can be set to "yes" which will cause the receiver process to ignore all attribute data it receives from remote locations.

Suppression of Locked Data Checking (changed in 12.0)

The ClearReplica Receiver processes can be configured to suppress certain checks it makes for locked label, branch or attribute types at the beginning of processing a packet. When the receiver checks for locked metadata types before it process element versions in the packet to write warning similar to the one below in the warning logs.

```
[Mon Oct 19 12:28:06 2015 UTC -05:00] -- Warning: Label (TESTED) locked in
receiving VOB (\SIZZLER) for all users...  No such labels will be applied
to versions in this pass...
```

So one would expect that any applications of that label to versions would cause a warning when attempted (it does) as depicted below.

```
ClearReplica Error: Lock on Label type TESTED " prevents operation "make
label" on version "bar.c@@\main\1".
```

If during your replication you would normally send thousands of label types from a location to a location where all of those types would be expected to be locked then it might not be necessary to log and review the initial warnings when the second warning (also written to the warning logs) might suffice.

The behavior for suppression of the initial lock checking for each of types is controlled by its associated **lock_supression_checking** configuration item in the **receiver.conf** file:

Metadata Type	lock_supression_checking parameter
lbtype (labels)	**suppress_locked_label_checking**
brtype (branches)	**suppress_locked_branch_checking**
attype (attributes)	**suppress_locked_attribute_checking**

Links to sections that provide more details for each parameter follow:

- **Receiver-side Suppressing Label Locked Checking**
- **Receiver-side Suppressing Branch Locked Checking**
- **Receiver-side Suppressing Attribute Locked Checking**

Note: For each of these options the default if not provided is to NOT ignore the associated request.

Receiver-side Suppressing Label Locked Checking

One can configure the ClearReplica **receiver.conf** file to either process to suppress locked label warnings displayed in the receiver logs while the receiver pre-processes the packet. This is helpful when replicas contain thousands of label types which may be locked at the receiver. Alternately, the shipper could elect to send minimal label data by taking advantage of **label filters** at that location. The **suppress_locked_label_checking** receiver configuration parameter is set to "no" (the default) if not defined which tells the receiver process to display any cleartool locked label warnings in the receiver logs. The **suppress_locked_label_checking** parameter can be set to "yes" which will cause the receiver process to suppress these locked label checks and warnings. Enabling this feature can significantly improve receiver packet performance time for shippers that send packets with thousands of label types not modified in the receiver. You should only enable this feature if it is known that the labels types sent from the shipper are ***unlocked*** at the receiver location.

Receiver-side Suppressing Branch Locked Checking (new to 12.0)

One can configure the ClearReplica **receiver.conf** file to either process to suppress locked label warnings displayed in the receiver logs while the receiver pre-processes the packet. This is helpful when replicas contain thousands of branch types which may be locked at the receiver. Alternately, the shipper could elect to send minimal branch data by taking advantage of **branch filters** at that location. The **suppress_locked_branch_checking** receiver configuration parameter is set to "no" (the default) if not defined which tells the receiver process to display any cleartool locked branch warnings in the receiver logs. The **suppress_locked_branch_checking** parameter can be set to "yes" which will cause the receiver process to suppress these locked label checks and warnings. Enabling this feature can significantly improve receiver packet performance time for shippers that send packets with thousands of branch types not modified in the receiver. You should only enable this feature if it is known that the branch types sent from the shipper are ***unlocked*** at the receiver location.

Receiver-side Suppressing Attribute Locked Checking (new to 11.0)

One can configure the ClearReplica **receiver.conf** file to either process to suppress locked label warnings displayed in the receiver logs while the receiver pre-processes the packet. This is helpful when replicas contain thousands of attribute types which may be locked at the receiver. Alternately, the shipper could elect to send minimal attribute data by taking advantage of **attribute filters** at that location. The **suppress_locked_attribute_checking** receiver configuration parameter is set to "no" (the default) if not defined which tells the receiver process to display any cleartool locked attribute warnings in the receiver logs. The **suppress_locked_attribute_checking** parameter can be set to "yes" which will cause the receiver process to suppress these locked atribute checks and warnings. Enabling this feature can significantly improve receiver packet performance time for shippers that send packets with thousands of attribute types not modified in the receiver. You should only enable this feature if it is known that the attribute types sent from the shipper are ***unlocked*** at the receiver location.

Replication of VOBs (or VOB Portions)

In addition to the increased flexibility ClearReplica provides in "**where**" one can replicate, ClearReplica provide added flexibility in "**what**" can be replicated. ClearCase Multisite allows for a whole VOB to be replicated to other ClearCase regions. ClearReplica also allows this functionality, but also adds the abilities listed below:

- ClearReplica allows for the duplication of VOB data within the "**same**" region (excellent for hot online backups or distribution areas).
- ClearReplica allows for the replication of a "**portion**" (including a single file) or many disjoint "portions" of a VOB to other VOBs in a different or same region (even the same VOB). This is critical in **saving corporate bandwidth** as now one does not have to "replicate" the "whole" VOB when just a fraction of it is really needed by the off-site development group.
- ClearReplica allows for the replication of data from "many" VOBs into a single VOB (excellent for limiting the accessibility of code to 3rd party vendors).
- ClearReplica allows for the replication of data in a VOB as limited by the "visibility" of a user defined config. spec. (i.e. branches, label, etc. can be used to limit what is "replicated").

Replication of existing VOBs

ClearReplica was designed to keep VOBs (or VOB portions) in sync. By making sure changes in a VOB (or portion) are propagated to its "replica" VOB (or portion). You can create an empty VOB (or VOB directory) at the destination location and replicate an existing VOB (or portion) to it. When replication an existing older/larger VOBs it is best to first copy the VOB to the destination location and then turn "replication" on for that VOB.

For example if you wanted to replicate the **/vobs/a_vob** from ***Atlanta*** to ***Japan***. It would be best to first ***copy*** the VOB from the Atlanta location (region) to the Japan location (region) and then create a **replication_key** to "replicate" from Atlanta to Japan. Now development from either location is duplicated in the other location.

To "copy" the VOB from one region to another refer to the ClearCase Administration Manual for copying ClearCase VOBs. The VOB should be copied at a known time or backup (i.e. today at 5:00 p.m.) and then the **replication epoch time** set back to sometime just before the copy time (today at 4:50 p.m.) to start replication.

Creating Replication_definitions

There are four (4) steps that must be executed in order to "replicate" a VOB or VOB portions. They will not have to be performed each time as some steps apply to the "location" or site.

- **Get a replication_key from the "receiver" location:** Collaborate with the "receiving" site and receive a **replication_key** (which grants permission for a *specific* sender to send packets to a *specific* directory at the receiving site. (the ClearReplica "receiver" will only process packets it receives in specific directories and only from sites it recognizes). You will only have to get a replication_key once per receiver you want to send to. (You will create one for sites you wish to receive packets from.)

- **Define the location_host_entry for the "receiver" location:** This is simply associating a location_name like "Atlanta" (easy for people to associate) with the **replication_key** (that is encrypted and not really human readable). You will use location_names in your **replication_definitions**.

- **Define the replication_definition's Config. Spec.:** Designate the Config. Spec. that defines the "visibility" during replication.

- **Define the replication_definition:** The **replication definitions** defines what is to be replicated and where it is to be sent.

Additionally and optionally, if you are replicating an older or larger VOB that you have previously copied to the destination location, you may decide to initially set the **replication epoch time** back to a time just before the copy time to start replication.

Creating a replication_key

In order for the ClearReplica "shipper" (clearreplica_shipper) to send a replication packet to a receiver it must have the proper **replication_key** provided by the "receiver" site. This key is used by the "shipper" location to send packets to the "receiver" location. A member of the "shipper" location request from the "receiver" a key which the "receiver' produces by executing the **clearreplica_hostentry** program (if the ClearReplica admin for the "shipper" site knows the appropriate information then they can produce the key). The site administrator can issue multiple keys for receiving in different **replication_receiver_directories** (i.e. if the "receiver" wanted to receiver all packets from Atlanta in the directory **/clearreplica/receiver/atlanta** and all packets from London in the directory **/clearreplica/receiver/London)**. Making replication_receiver_directories is discussed in the next section.

Making Replication Receiver Directories (changed in 12.0)

ClearReplica replication packets are received in **replication_receiver_directories**. This directory location is embedded in the encrypted **replication_key** the "receiver" provides the "shipper" administrator. The ClearReplica administrator should make sure the any replication_receiver_directory exist and is writable by the userid provided in any associated replication_key.

The **replication_receiver_directory** has **a physical_receiver_directory** that should also be inserted in the **receiver.conf** file on the **{depot}/m_bay/config** directory. For UNIX this physical_receiver_directory is usually the same as the replication_receiver_directory because the FTP root is usually the machine's root directory ("/"). For Windows this physical_receiver_directory is usually different as the FTP root is usually assigned and not the root of the file system (i.e. the FTP root might be **C:\Inetpub\ftproot**) so a Windows replication_receiver_directory of "**clearreplica_receiver**" might have a physical_receiver_directory of "**C:\Inetpub\ftproot\clearreplica_receiver**" that must be placed in the receiver.conf file.

The "receiver" administrator should make sure the **physical_receiver_directory** (and therefore the **replication_receiver_directory**) is created and writable by the userid provided for any associated **replication_keys**.

The site administrator can issue ***multiple*** keys for different "receiving" directories for example if the "receiver" wanted to receiver all packets from Atlanta in the directory **/clearreplica/receiver/atlanta** and all packets from London in the directory **/clearreplica/receiver/London**.

A **receiver directory definition** contains 1-3 parameters or parameter groups separated by the '>' character:

- A *required* **physical directory parameter**
- An *optional* **protection parameter**
- An *optional* **processing parameter group**

- Physical Directory Parameter

The **physical directory parameter** is the physical location of the actual receiver directory.

- Protection Parameter

The **protection parameter** is optional and determines how newly created elements create in the "receiver VOBs" are owner. The valid values and what they mean are listed in the table below:

Protection Parameter (values)	Meaning
receiver	**the default – ensures that newly created elements take the ownership of the currently receiving process id (the id running the ClearReplica_receiver process).**
preserve	**ensures that the newly created elements take the same ownership they had in the "original sending VOB"**
exactly	**ensures that newly created elements take on the user:group ownership that is explicitly defined in the key (both the user and group MUST be provided)**

- Processing Parameter Group

The **processing parameter group** is optional and contains zero or more comma separated **Processing Parameters** that determine the behavior of packets processed in the associated **receiver directory**.

The processing parameter group can contain a **rmelem_processing_value** from the table below:

rmelem Processing Parameter (values)	Meaning
allow_rmemlem_none	**the default if none in this group is provided – ensures that any/all rmelem requests from other replicas are ignored**
allow_rmemlem_file	**ensures that only rmelem request for "files" from other replicas are accepted/processed**
allow_rmemlem_dir	**ensures that only rmelem request for "direstories" from other replicas are accepted/processed**
allow_rmemlem_all	**ensures that all rmelem request for "files" and "directories" from other replicas are accepted/processed**

Note: Only the first one encountered is used, any others that appear within the group are ignored.

The **rmbranch processing parameter** is optional and determines how rmbranch requests from remote locations are processed locally. The valid values and what they mean are listed in the table below:

rmbranch Processing Parameter (values) (new to 12.0)	Meaning
allow_rmbranch_none	**the default if none in this group is provided – ensures that any/all rmbranch requests from other replicas are ignored**
allow_rmbranch	**ensures that any/all rmbranch requests for (e.g. cleartool rmbranch foo.c@@\main\b2) from other replicas are accepted/processed. Any local branch type branch instance or element locks are honored.**

Note: Only the first one encountered is used, any others that appear within the group are ignored.

The **rmver processing parameter** is optional and determines how rmver requests from remote locations are processed locally. The valid values and what they mean are listed in the table below:

rmver Processing Parameter (values) (new to 12.0)	Meaning
allow_rmver_none	**the default if none in this group is provided – ensures that any/all rmver requests from other replicas are ignored**
allow_rmver	**ensures that any/all rmver requests for (e.g. cleartool rmver foo.c@@\main\b2\7) from other replicas are accepted/processed. Any local branch instance or element locks are honored.**

Note: Only the first one encountered is used, any others that appear within the group are ignored.

The **rmlbtype processing parameter** is optional and determines how rmtype -lbtype requests from remote locations are processed locally. The valid values and what they mean are listed in the table below:

rmlbtype Processing Parameter (values)	Meaning
allow_rmlbtype_none	**the default if none in this group is provided – ensures any/all rmtype - lbtype request from other replicas are ignored**
allow_rmlbtype	**ensures that all rmtype - lbtype request for (e.g. cleartool rmtype lbtype:FOO) from other replicas are accepted/processed. Any local label type locks are honored.**

Note: Only the first one encountered is used, any others that appear within the group are ignored.

The **rmbrtype processing parameter** is optional and determines how rmtype -brtype requests from remote locations are processed locally. The valid values and what they mean are listed in the table below:

rmbrtype Processing Parameter (values) (new to 12.0)	Meaning
allow_rmbrtype_none	**the default if none in this group is provided – ensures any/all rmtype - brtype request from other replicas are ignored**
allow_rmbrtype	**ensures that all rmtype - brtype request for (e.g. cleartool rmtype brtype:fea_1) from other replicas are accepted/processed. Any local branch type locks are honored.**

Note: Only the first one encountered is used, any others that appear within the group are ignored.

The **rmattype processing parameter** is optional and determines how rmtype -attype requests from remote locations are processed locally. The valid values and what they mean are listed in the table below:

rmattype Processing Parameter (values) (new to 12.0)	Meaning
allow_rmattype_none	**the default if none in this group is provided – ensures any/all rmtype - attype request from other replicas are ignored**
allow_rmattype	**ensures that all rmtype - attype request for (e.g. cleartool rmtype attype:qa_state) from other replicas are accepted/processed. Any local attribute type locks are honored.**

Note: Only the first one encountered is used, any others that appear within the group are ignored.

The **rmtype processing parameter** is optional works as a **super-parameter** that encompasses the **rmlbtype processing, rmbrtype processing** and the **rmattype processing parameters** as a whole. The valid values and what they mean are listed in the table below:

rmtype Processing Parameter (values) (new to 12.0)	Meaning
allow_rmtype_none	**If provided it is equivalent to providing the 3 values:** **allow_rmlbtype_none** **allow_rmbrtype_none** **allow_rmattype_none**
allow_rmtype_all	**If provided it is equivalent to providing the 3 values:** **allow_rmlbtype** **allow_rmbrtype** **allow_rmattype**

Note: Only the first one encountered is used, any others that appear within the group are ignored.

Note: The existence of a **super-parameter** in the string will override non-super_parameters of the same type.

Given the allowed values for members of the rmlbtype, rmbrtype, rmattype and rmtype parameter groupings; each of the entries below are equivalent:

- `>allow_rmlbtype, allow_rmbrtype, allow_rmattype >`
- `>allow_rmbrtype, allow_rmattype, allow_rmlbtype >`
- `>allow_rmtype_all >`

These entries below are also equivalent:

- `>allow_rmlbtype_none, allow_rmbrtype_none, allow_rmattype_none >`
- `>allow_rmbrtype_none, allow_rmattype_none, allow_rmlbtype_none >`
- `>allow_rmtype_none >`
- `>>`

The **lbtype lock/unlock processing parameter** is optional and determines how label type lock requests from remote locations are processed locally. The valid values and what they mean are listed in the table below:

lbtype_lock Processing Parameter (values)	Meaning
allow_lbtype_locks_none	**the default if none in this group is provided – ensures that lock/unlock lbtype request from other replicas are ignored**
allow_lbtype_locks	**ensures that lock/unlock lbtype request from other replicas are honored**

Note: Only the first one encountered is used, any others that appear within the group are ignored.

The **brtype lock/unlock processing parameter** is optional and determines how branch type locksrequests from remote locations are processed locally. The valid values and what they mean are listed in the table below:

brtype_lock Processing Parameter (values) (new to 12.0)	Meaning
allow_brtype_locks_none	**the default if none in this group is provided – ensures that lock/unlock brtype request from other replicas are ignored**
allow_brtype_locks	**ensures that lock/unlock brtype request from other replicas are honored**

Note: Only the first one encountered is used, any others that appear within the group are ignored.

The **attype lock/unlock processing parameter** is optional and determines how attribute type lock requests from remote locations are processed locally. The valid values and what they mean are listed in the table below:

attype_lock Processing Parameter (values) (new to 12.0)	Meaning
allow_attype_locks_none	**the default if none in this group is provided – ensures that lock/unlock attype request from other replicas are ignored**
allow_attype_locks	**ensures that lock/unlock attype request from other replicas are honored**

Note: Only the first one encountered is used, any others that appear within the group are ignored.

The **type_lock processing parameter** is optional works as a **super-parameter** that encompasses the **lbtype_lock processing, brtype_lock processing** and the **attype_lock processing parameters** as a whole. The valid values and what they mean are listed in the table below:

type_lock Processing Parameter (values) (new to 12.0)	Meaning
allow_type_locks_none	**If provided it is equivalent to providing the 3 values:** **allow_lbtype_locks_none** **allow_brtype_locks_none** **allow_attype_locks_none**
allow_type_locks_all	**If provided it is equivalent to providing the 3 values:** **allow_lbtype_locks** **allow_brtype_locks** **allow_attype_locks**

Note: Only the first one encountered is used, any others that appear within the group are ignored.

Note: The existence of a **super-parameter** in the string will override non-super_parameters of the same type.

Given the allowed values for members of the lbtype_lock, brtype_lock, attype_lock and type_lock parameter groupings; each of the entries below are equivalent:

- >allow_lbtype_locks, allow_brtype_locks, allow_attype_locks >
- >allow_brtype_locks, allow_attype_locks, allow_lbtype_locks >
- >allow_type_locks_all >

These entries below are also equivalent:

- >allow_lbtype_locks_none, allow_brtype_locks_none, allow_attype_locks_none >
- >allow_brtype_locks_none, allow_attype_locks_none, allow_lbtype_locks_none >
- >allow_type_locks_none >
- >>

The **rnlbtype processing parameter** is optional and determines how rntype -lbtype requests from remote locations are processed locally. The valid values and what they mean are listed in the table below:

rnlbtype Processing Parameter (values) (new to 12.0)	Meaning
allow_rnlbtype_none	**the default if none in this group is provided – ensures any/all rntype -lbtype request from other replicas are ignored**
allow_rnlbtype	**ensures that all rntype - lbtype request for (e.g. cleartool rntype lbtype:FOO BAR) from other replicas are accepted/processed. Any local label type locks are honored.**

Note: Only the first one encountered is used, any others that appear within the group are ignored.

The **rnbrtype processing parameter** is optional and determines how rntype -brtype requests from remote locations are processed locally. The valid values and what they mean are listed in the table below:

rnbrtype Processing Parameter (values) (new to 12.0)	Meaning
allow_rnbrtype_none	**the default if none in this group is provided – ensures any/all rntype - brtype request from other replicas are ignored**
allow_rnbrtype	**ensures that all rntype - brtype request for (e.g. cleartool rntype brtype:fea_1 fea_one) from other replicas are accepted/processed. Any local branch type locks are honored.**

Note: Only the first one encountered is used, any others that appear within the group are ignored.

The **rnattype processing parameter** is optional and determines how rntype -attype requests from remote locations are processed locally. The valid values and what they mean are listed in the table below:

rnattype Processing Parameter (values) (new to 12.0)	Meaning
allow_rnattype_none	**the default if none in this group is provided – ensures any/all rntype - attype request from other replicas are ignored**
allow_rnattype	**ensures that all rmtype - attype request for (e.g. cleartool rntype attype:qa_state QA_state) from other replicas are accepted/processed. Any local attribute type locks are honored.**

Note: Only the first one encountered is used, any others that appear within the group are ignored.

The **rntype processing parameter** is optional works as a **super-parameter** that encompasses the **rnlbtype processing, rnbrtype processing** and the **rnattype processing parameters** as a whole. The valid values and what they mean are listed in the table below:

rntype Processing Parameter (values) (new to 12.0)	Meaning
allow_rntype_none	**If provided it is equivalent to providing the 3 values:** **allow_rnlbtype_none** **allow_rnbrtype_none** **allow_rnattype_none**
allow_rntype_all	**If provided it is equivalent to providing the 3 values:** **allow_rnlbtype** **allow_rnbrtype** **allow_rnattype**

Note: Only the first one encountered is used, any others that appear within the group are ignored.

Note: The existence of a **super-parameter** in the string will override non-super_parameters of the same type.

Given the allowed values for members of the rnlbtype, rnbrtype, rnattype and rntype parameter groupings; each of the entries below are equivalent:

- `>allow_rnlbtype, allow_rnbrtype, allow_rnattype >`
- `>allow_rnbrtype, allow_rnattype, allow_rnlbtype >`
- `>allow_rntype_all >`

These entries below are also equivalent:

- `>allow_rnlbtype_none, allow_rnbrtype_none, allow_rnattype_none >`
- `>allow_rnbrtype_none, allow_rnattype_none, allow_rnlbtype_none >`
- `>allow_rntype_none >`
- `>>`

The **element lock/unlock processing parameter** is optional and determines how element instance lock requests from remote locations are processed locally. The valid values and what they mean are listed in the table below:

element_lock Processing Parameter (values) (new to 12.0)	Meaning
allow_element_locks_none	**the default if none in this group is provided – ensures that lock/unlock element instance request from other replicas are ignored**
allow_element_locks	**ensures that lock/unlock element instance request from other replicas are honored**

Note: Only the first one encountered is used, any others that appear within the group are ignored.

The **branch lock/unlock processing parameter** is optional and determines how branch instance lock requests from remote locations are processed locally. The valid values and what they mean are listed in the table below:

branch_lock Processing Parameter (values) (new to 12.0)	Meaning
allow_branch_locks_none	**the default if none in this group is provided – ensures that lock/unlock branch instance request from other replicas are ignored**
allow_branch_locks	**ensures that lock/unlock branch instance request from other replicas are honored**

Note: Only the first one encountered is used, any others that appear within the group are ignored.

The **rmtype processing parameter** is optional works as a **super-parameter** that encompasses the **lbtype_lock processing, brtype_lock processing** and the **attype_lock processing parameters** as a whole. The valid values and what they mean are listed in the table below:

instance_lock Processing Parameter (values) (new to 12.0)	Meaning

allow_instance_locks_none	**If provided it is equivalent to providing the 2 values:**
	`allow_element_locks_none`
	`allow_branch_locks_none`

allow_instance_locks_all	**If provided it is equivalent to providing the 2 values:**
	`allow_element_locks`
	`allow_branch_locks`

Note: Only the first one encountered is used, any others that appear within the group are ignored.

Note: The existence of a **super-parameter** in the string will override non-super_parameters of the same type.

Given the allowed values for members of the element_lock, branch_lock and instance_lock parameter groupings; each of the entries below are equivalent:

- `>allow_element_locks, allow_branch_locks >`
- `>allow_branch_locks, allow_element_locks >`
- `>allow_instance_locks_all >`

These entries below are also equivalent:

- `>allow_element_locks_none, allow_branch_locks_none >`
- `>allow_branch_locks_none, allow_element_locks_none >`
- `>allow_instance_locks_none >`
- `>>`

The **label_dir processing parameter** is optional and determines how/if directories versioned are labeled during replication. By default, during replication the receiver labels all directories and their imported parents with the included versions union of all labels applied during replication of file versions contained in those directories. This parameter allows you to change this behavior.

The valid values and what they mean are listed in the table below:

Label_dirs Processing Parameter (values)	Meaning
label_dir	**the default if none in this group is provided - labels all directories and their imported parents with the versions union of all labels applied during replication of file versions contained in those directories. To defeat this behavior, use the no_label_dir option.**
no_label_dir	**ensures that directory versions are not labeled the receiving process.**

Converting pre 12.0 receiver_directory definitions to 12.0 (new to 12.0)

Prior to ClearReplica 12.0 many of the *optional* **processing parameter group** options for a **replication_receiver_dir** were singularly field delimited and thus were required in a certain order. This has been relaxed in 12.0 and higher as these parameters can be defined in any order to make it easier for the ClearReplica administrator. Additional **super-parameter** shortcuts have been added as well that allow the definitions to be much shorter and easier to create and understand. View the examples below to get a feel for changes needed to convert to 12.0. You place in just what you need and in the order that makes best sense to you.

This pre 12.0 definition:
`C:\CR\_receiver>>allow_rmelem_all>allow_rmlbtype>allow_rmbrtype>allow_lbtype_locks>label_dir>allow_rmattype>allow_brtype_locks>allow_attype_locks>`
Can now be written as any of these:
`C:\_receiver>>allow_rmelem_all,allow_rmtype_all,allow_type_locks,label_dir`
`C:\_receiver>>allow_rmelem_all,label_dir,allow_rmtype_all,allow_type_locks`
`C:\_receiver>>allow_type_locks,allow_rmelem_all,label_dir,allow_rmtype_all`

This pre 12.0 definition:
`C:\receiver_dir>preserve>>>>>no_label_dir>`
Can now be written as:
`C:\receiver_dir>preserve>no_label_dir`

This pre 12.0 definition:
`C:\_receiver>>>allow_rmlbtype>allow_rmbrtype>>>allow_rmattype>>allow_attype_locks>`
Can now be written as any of these:
`C:\receiver_dir>>allow_attype_all,allow_attype_locks`
`C:\receiver_dir>>allow_attype_locks,allow_attype_all`

Example receiver.conf file (changed in 12.0)

```
#########################################################################
# The receiver.conf file is read by clearreplica_receiver to know which #
# receiver directories to watch for incoming packets.                    #
#                                                                        #
# blank lines are ignored                                                #
# Lines that start with a '#' are comments only and also ignored.        #
#                                                                        #
# the view=<view_name> line names the incoming replication view          #
# the location_name=<location_name> line names this location             #
# the UTC_offset=<offset> line defines the difference between GMT and    #
#                         this time zone for <location_name>.            #
# the MVFS_DRIVE=<drive> line names the MVFS Drive letter for the        #
#                        machine running the receiver process. For UNIX #
#                        this variable-value pair is ignored.            #
# the receiver_id=<userid> line indicates the only userid allowed to     #
#                         run the "receiver" process. Usually the owner #
#                         of the VOBs. This ID must have write access    #
#                         to all VOBs that it receives packets for.      #
# the pre_receive_trigger=<script> line indicates script (or program)    #
#                         that can be called prior to processing any     #
#                         received packet.  (optional)                   #
# the post_receive_trigger=<script> line indicates script (or program)   #
#                         that can be called after processing any        #
#                         received packet.    (optional)                 #
# new_dir_personality=<personality> line determines if new directory     #
#                         versions are made on /main or as per the       #
#                         config. spec. send by the shipper location.    #
#                         The values of personality are "main" to        #
#                         place new version on /main or "branch" to      #
#                         place new versions as per the config. spec.    #
#                         sent by the shipper. (default is "main")       #
#                         that can be called after processing any        #
#                         received packet.    (optional)                 #
# encryption=<encryption_type> line determines if packets processed      #
#                         with this receiver will used "standard"        #
#                         ClearReplica encryption or "peer_to_peer"      #
#                         ClearReplica encryption.                       #
#                         (default is "standard") (optional)             #
# startup_trigger=<script> line indicates script (or program) that is    #
#                         called immediately after the successful        #
#                         startup of a ClearReplica receiver process.    #
#                         (optional)                                     #
# shutdown_trigger=<script> line indicates script (or program) that is   #
#                         called immediately after the successful        #
#                         shutdown of a ClearReplica receiver process.   #
#                         (optional)                                     #
# scheduled_inactivity=<HH:MM-HH:MM> Defines a well-known time that the  #
#                         receiver will NOT process packets.  During     #
#                         this optionally defined duration, packets are  #
#                         still received from remote locations, but      #
#                         processing of those packets is postponed       #
#                         until outside the duration. (optional)         #
# ignore_received_slinks=<slink_personality> Optional parameter with     #
#                         the value of either "yes" or "no" that         #
#                         determines if newly created symbolic links     #
#                         are created if included in received packets.   #
#                         The default value is "no" if no value is       #
#                         provided. (optional)                           #
```



```
#                                                                              #
# ignore_received_label_data=<label_data_personality> Optional parameter#
#                          with the value of either "yes" or "no" newly  #
#                          received label data is replicated to ignored  #
#                          by the receiver if included in received       #
#                          packets.  The default value is "no" if no     #
#                          value is provided. (optional)                 #
# suppress_locked_label_checking=<locked_label_checking_personality>    #
#                          Optional parameter with the value of either   #
#                          "yes" or "no" that determines if newly        #
#                          received label types are tested for being     #
#                          locked and written to the receiver logs upon  #
#                          packet receipt for packets containing labels  #
#                          that are locked in the receiver. The default  #
#                          value is "no" if none is provided.(optional)  #
#                                                                        #
# Additional variables explained in ClearReplica administration manual. #
#                                                                        #
# All other lines are treated as receiver directories that the           #
# ClearReplica_receiver will attempt to watch.                           #
##########################################################################

view=receiver_view
location_name=atlanta
UTC_offset=-05:00
MVFS_DRIVE=m
receiver_id=cclarke
pre_receive_trigger=
post_receive_trigger=email_errors.bat
startup_trigger=receiver_startup.bat
shutdown_trigger=notify_shutdown.bat
scheduled_inactivity=02:00-04:00
new_dir_personality=main
encryption=standard

##############################################################################
# These receiving bay definitions are read continually so changes you make to #
# them are available at the very next process time. The process time right    #
# after the after the last receiving bay definition has completed.           #
#                                                                            #
# The directories are those that can be populated via FTP from ClearReplica  #
# shipper machines in this machine or other machines in other ClearReplica   #
# regions.                                                                   #
#                                                                            #
# Each directory can have an additional (optional) parameter defined that    #
# determines how newly created elements created in the "receiver VOBs" are   #
# owned.  The valid values are:                                              #
#                                                                            #
#         "receiver" - the default - ensures that newly created elements take #
#                      the ownership of the currently receiving process id   #
#                      (the id running the ClearReplica_receiver process).   #
#         "preserve" - ensures that the newly created elements take the      #
#                      same ownership they had in the "original sending VOB" #
#         "exactly"  - ensures that newly created elements take on the same  #
#                      user:group ownership that is explicitly defined in the #
#                      key (both the user and group MUST be provided)        #
#                                                                            #
```

```
#                                                                              #
# Each directory can have an additional (optional) parameter defined that      #
# determines how remote rmelem request are processed locally.                  #
# The valid values are:                                                        #
#                                                                              #
#         "allow_rmemlem_none" - the default if not provided - ensures that    #
#                                rmelem request from other replicas are        #
#                                ignored                                       #
#         "allow_rmemlem_file" - ensures that only rmelem request for "files"  #
#                                from other replicas are accepted/processed    #
#         "allow_rmemlem_dir"  - ensures that only rmelem request for "dirs"   #
#                                from other replicas are accepted/processed    #
#         "allow_rmemlem_all"  - ensures that all rmelem request for "files"   #
#                                and "directories" from other replicas are     #
#                                accepted/processed                            #
#                                                                              #
# Each directory can have an additional (optional) parameters defined that     #
# determines how remote rmtype -lbtype -brtype or -attype request are          #
# processed locally.  The valid values are:                                    #
#                                                                              #
#         "allow_rmlbtype_none" - the default if not provided - ensures that   #
#                                 rmtype -lbtype request from other replicas   #
#                                 are ignored                                  #
#                                                                              #
#         "allow_rmlbtype"      - ensures that all rmtype -lbtype request      #
#                                 from other replicas are accepted/processed   #
#                                                                              #
#         "allow_rmbrtype_none" - the default if not provided - ensures that   #
#                                 rmtype -brtype request from other replicas   #
#                                 are ignored                                  #
#                                                                              #
#         "allow_rmbrtype"      - ensures that all rmtype -brtype request      #
#                                 from other replicas are accepted/processed   #
#                                                                              #
#         "allow_rmbrtype_none" - the default if not provided - ensures that   #
#                                 rmtype -brtype request from other replicas   #
#                                 are ignored                                  #
#                                                                              #
#         "allow_rmtype_all" - equivalent to using the 3 parameters:           #
#                              allow_rmbrtype,                                 #
#                              allow_rmbrtype,                                 #
#                              allow_rmbrtype                                  #
#                                                                              #
#         "allow_rmtype_none" - equivalent to using the 3 parameters:          #
#                               allow_rmbrtype_none,                           #
#                               allow_rmbrtype_none,                           #
#                               allow_rmbrtype_none                            #
#                                                                              #
# Each directory can have an additional (optional) parameter defined that      #
# determines how remote locking and unlocking of lbtype, brtype or attypes     #
# request are processed locally.  The valid values are:                        #
#                                                                              #
#         "allow_branch_locks_none" - the default if not provided - ensures    #
#                                     that lock:unlock branch request from other #
#                                     replicas are ignored.                    #
#                                                                              #
#         "allow_branch_locks"       - ensures that lock:unlock branch request #
#                                     from other replicas are accepted/processed #
#                                                                              #
#         "allow_element_locks_none" - the default if not provided - ensures   #
#                                     that lock:unlock element request from    #
#                                     other replicas are ignored.              #
```

```
#                                                                              #
#         "allow_element_locks"    - ensures that lock:unlock element request #
#                                    from other replicas are accepted/processed #
#                                                                              #
#         "allow_instance_locks_all" - equivalent to including/using both      #
#                                    allow_branch_locks and allow_element_locks #
#                                                                              #
#         "allow_instance_locks_none" - equivalent to including/using both     #
#                                    allow_branch_locks_none and               #
#                                    allow_element_locks_none                  #
#                                                                              #
# Each directory can have an additional (optional) parameters defined that     #
# determines how remote rmtype -lbtype -brtype or -attype request are          #
# processed locally.  The valid values are:                                    #
#                                                                              #
#         "allow_rnlbtype_none" - the default if not provided - ensures that   #
#                                  rntype -lbtype request from other replicas  #
#                                  are ignored                                 #
#                                                                              #
#                                                                              #
#         "allow_rnlbtype"      - ensures that all rntype -lbtype request      #
#                                  from other replicas are accepted/processed  #
#                                                                              #
#         "allow_rnbrtype_none" - the default if not provided - ensures that   #
#                                  rntype -brtype request from other replicas  #
#                                  are ignored                                 #
#                                                                              #
#         "allow_rnbrtype"      - ensures that all rntype -brtype request      #
#                                  from other replicas are accepted/processed  #
#                                                                              #
#         "allow_rnbrtype_none" - the default if not provided - ensures that   #
#                                  rntype -brtype request from other replicas  #
#                                  are ignored                                 #
#                                                                              #
#         "allow_rntype_all" - equivalent to using the 3 parameters:           #
#                              allow_rnbrtype,                                 #
#                              allow_rnbrtype,                                 #
#                              allow_rnbrtype                                  #
#                                                                              #
#         "allow_rntype_none" - equivalent to using the 3 parameters:          #
#                              allow_rnbrtype_none,                            #
#                              allow_rnbrtype_none,                            #
#                              allow_rnbrtype_none                             #
#                                                                              #
# Each directory can have an additional (optional) parameters defined that     #
# determines how remote rntype -lbtype -brtype or -attype request are          #
# processed locally.  The valid values are:                                    #
#                                                                              #
#         "allow_lbtype_locks_none" - the default if not provided - ensures    #
#                                    that lock:lbtype and unlock:lbtype request #
#                                    from other replicas are ignored           #
#                                                                              #
#         "allow_lbtype_locks"      - ensures that lock/unlock:lbtype request  #
#                                    from other replicas are accepted/processed #
#                                                                              #
#         "allow_brtype_locks_none" - the default if not provided - ensures    #
#                                    that lock:brtype and unlock:brtype request #
#                                    from other replicas are ignored           #
#                                                                              #
#         "allow_brtype_locks"      - ensures that lock/unlock:brtype request  #
#                                    from other replicas are accepted/processed #
#                                                                              #
#         "allow_attype_locks_none" - the default if not provided - ensures    #
```

```
#                                  that lock:attype and unlock:attype request #
#                                  from other replicas are ignored            #
#         "allow_attype_locks"      - ensures that lock/unlock:attype request #
#                                  from other replicas are accepted/processed #
#         "allow_type_locks_all" - equivalent to using the 3 parameters:      #
#                              allow_lbtype_locks,                            #
#                              allow_brtype_locks,                            #
#                              allow_attype_locks                             #
#                                                                             #
#         " allow_type_locks_none" - equivalent to using the 3 parameters:    #
#                              allow_lbtype_locks_none,                       #
#                              allow_brtype_locks_none,                       #
#                              allow_attype_locks_none                        #
#                                                                             #
#                                                                             #
# Each directory can have an additional (optional) parameter defined that     #
# determines how or if element directory versions are labeled.                #
# The valid values are:                                                       #
#         "label_dir" - the default if not provided - labels all directories  #
#                           and their imported parents with the versions      #
#                           union of all labels applied during import to      #
#                           contained in those directories.                   #
#                           To defeat this behavior, use the no_label_dir     #
#                           option.                                           #
#         "no_label_dir" - ensures that directory versions are not labeled    #
#                          during the receiving process.                      #
#                                                                             #
# Distributed with commented out examples below:                              #
#                                                                             #
# C:\Inetpub\ftproot\receiver                                                 #
# /clearreplica/receiver                                                      #
# /clearreplica/receiver_a>exactly=cclarke3:ccusers                           #
# /clearreplica/receiver_b>receiver                                           #
# /clearreplica/receiver_c>preserve                                           #
# /clearreplica/receiver_d>>allow_rmelem_all                                  #
# /clearreplica/receiver_e>preserve>allow_rmelem_file                         #
# /clearreplica/receiver_f>preserve>allow_rmlbtype,allow_rmbrtype,no_label_dir#
# /clearreplica/receiver_8>preserve>allow_rmelem_file,rmtype_all              #
# /clearreplica/receiver_h>preserve>allow_rmbrtype,allow_lbtype_locks         #
# /clearreplica/receiver_i>preserve>allow_rmbrtype,allow_attype_locks         #
# /clearreplica/receiver_j>preserve>allow_rmbranch,allow_brtype_locks         #
# /clearreplica/receiver_k>preserve>allow_rmelement,allow_type_locks_all      #
# /clearreplica/receiver_l>preserve>allow_element_locks,allow_branch_locks    #
# /clearreplica/receiver_m>preserve>allow_instance_locks,allow_branch_locks   #
# /clearreplica/receiver_n>preserve>allow_rmattype,no_label_dir               #
#                                                                             #
###############################################################################

C:\Inetpub\ftproot\receiver
C:\Inetpub\ftproot\receiver_QA>exactly=cclarke3:qa
C:\Inetpub\ftproot\receiver_Atlanta>receiver
C:\Inetpub\ftproot\receiver_onsite>preserve
C:\Inetpub\ftproot\receiver_EU>>allow_rmelem_all
C:\Inetpub\ftproot\receiver_UK>preserve>allow_rmelem_file
C:\Inetpub\ftproot\receiver_UK>preserve>allow_rmelem_file,allow_rmlbtype
C:\Inetpub\ftproot\receiver_UK>preserve>allow_rmbrtype,allow_lbtype_locks
C:\Inetpub\ftproot\receiver_UK>preserve>allow_rmbrtype,no_label_dir
```

Defining the location_host_entry

The **location_host_entry** is simply an association of a **location_name** like "Atlanta" (easy for people to associate with) with a **location_key** (that is encrypted and not really human readable). You will use **location_names** in your **replication_definitions**. The **location_key** is placed in the **hostmap** file of the name **<location>.hostmap** where <location> is the name given to the "receiver" location by the "shipper" location (i.e. "atlanta", or "london").

To create a location_host_entry for "London" create a file named "**london.hostmap"** in the **{depot}/m_bay/config/hostmaps** directory and place the location_key provided by the ClearReplica administrator of that site. Place ONLY the **location_key** in the appropriate hostmap file. **Location_keys** are generated by execution of the **clearreplica_hostentry** program.

Defining the replication_definition's Config. Spec.

Each replication_definition has an associated Config Spec. This Config. Spec. is used to provide the "visibility" of what is "replicated". ClearReplica will "replicate" any elements "visible" with that Config. Spec. in the VOB or VOB directory defined in the replication_definition.

CHECKEDOUT rules are not allowed in the Config. Spec. Config. Specs. that contain a CHECKEDOUT rule are invalid and ClearReplica will write to the error log and not process the associated replication_definition when it encounters such a rule.

Config. Specs. with the version selector rules from: **-config**, **-error**, **-time** are invalid and ClearReplica will write to the error log and not process the associated replication_definition when it encounters such a rule.

Place the associated Config. Spec. in the **config_spec** file of the name **###.cs** where ### is a 3 or 4 digit identifier (from "000"' – "9999") that corresponds with the identifier for the **replication_definition**. There is a one-to-one correspondence between replication_definitions and config_spec files. To create config_spec file for replication_definition "123" create a file named "**123.cs"** in the **{depot}/m_bay/config/config_specs** directory and place the desired Config. Spec. Place ONLY the desired Config. Spec. in the appropriate config_spec file.

Creating additional class_definitions

You should create the **class_definition** only after you have create its associated **shipper.conf** file because as soon as you create the **class_definition** in the **shipper.conf** file it is "active" and ClearTrigger will expect that the file exists for replication inspection. Creating a **class_definition** can provide two distinct benefits:

- **provide improved performance** by allowing multiple shippers to run in parallel on a single machine or even on different machines.
- **provide separation** between project groups

Each benefit and how to achieve this benefit is described in the sections that follow.

When ClearTrigger/ClearReplica enabled VOBs have changes performed in them they will interrogate the appropriate **shipper.conf** file to determine ClearReplica should mark that VOB section as "needing replication" during the next shipper process pass. If additional class_definitions are defined then this informs ClearTrigger that it should also interrogate that associated shipper.conf file for that class (i.e. "class 4" will cause the interrogation of the "shipper_**4**.conf" file) after interrogating the original shipper.conf file.

There can be up to 10 additional shipper classes defined. (0..9). The **class_definition** can only be defined in the original **shipper.conf** file (the only "class-less" shipper.conf file). The **class_definition** can exist on a line by itself in the shipper.conf file of the form:

add_class=#

Where # is ('0' – '9'). So examples of class definitions are below:

```
add_class=0
add_class=1
add_class=9
```

Please note that **class_definitions**:

- define which additional shipper_[class].conf file to interrogate.
- can only exist in the "class-less" shipper.conf file
- can not be duplicated in the shipper.conf file.

Again, creating a **class_definition** can provide two distinct benefits:

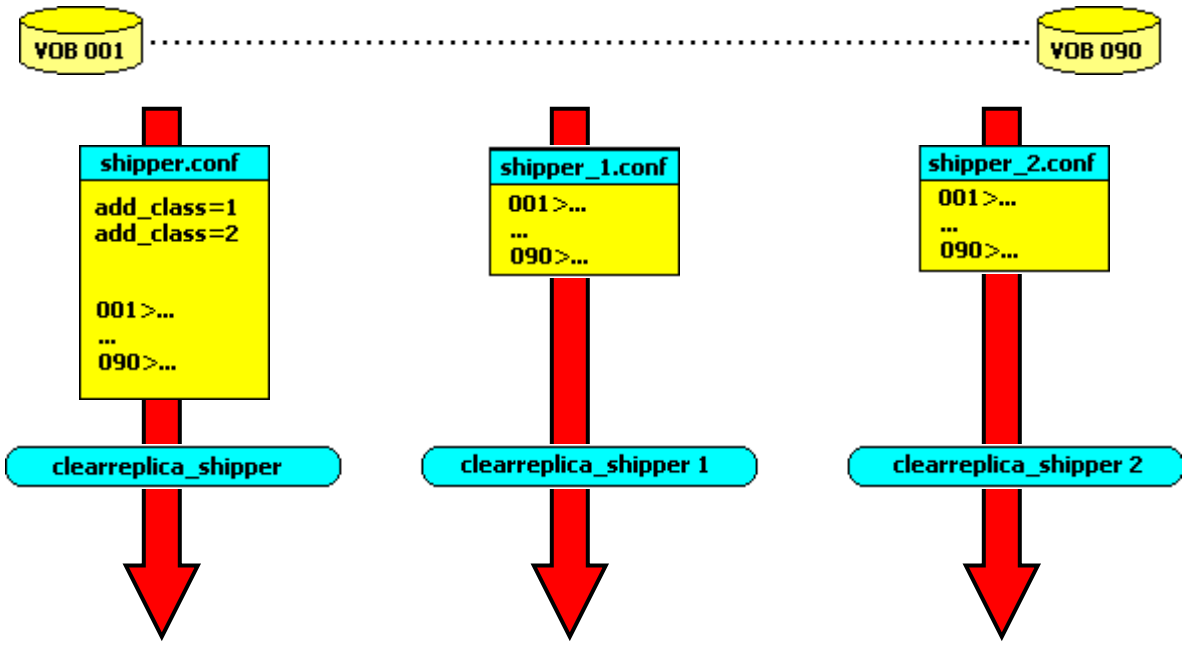
- **provide improved performance** by allowing multiple shippers to run in parallel on a single machine or even on different machines.
- **provide separation** between project groups

Each benefit and how to achieve this benefit is described in the sections that follow.

Using Class_Definitions to provide better packet building performance:

A ClearReplica Administrator can define/arrange the **shipper.conf** files so that each file has the same set of the desired **replication_keys** so that any class can process any of the keys. This allows the administrator to run all clearreplica_shipper process simultaneously allowing for better packet creation performance. different data manage a different file.

For example: A site with 90 VOBs might have all **replication_keys** related to VOBs 1-90 defined in **shipper.conf**, **shipper_1.conf** and **shipper_2.conf.** The administrator can now run all three shipper processes so that all process the data.



Now the next the first process that gets to a packet will process it while the other immediately process available packet.

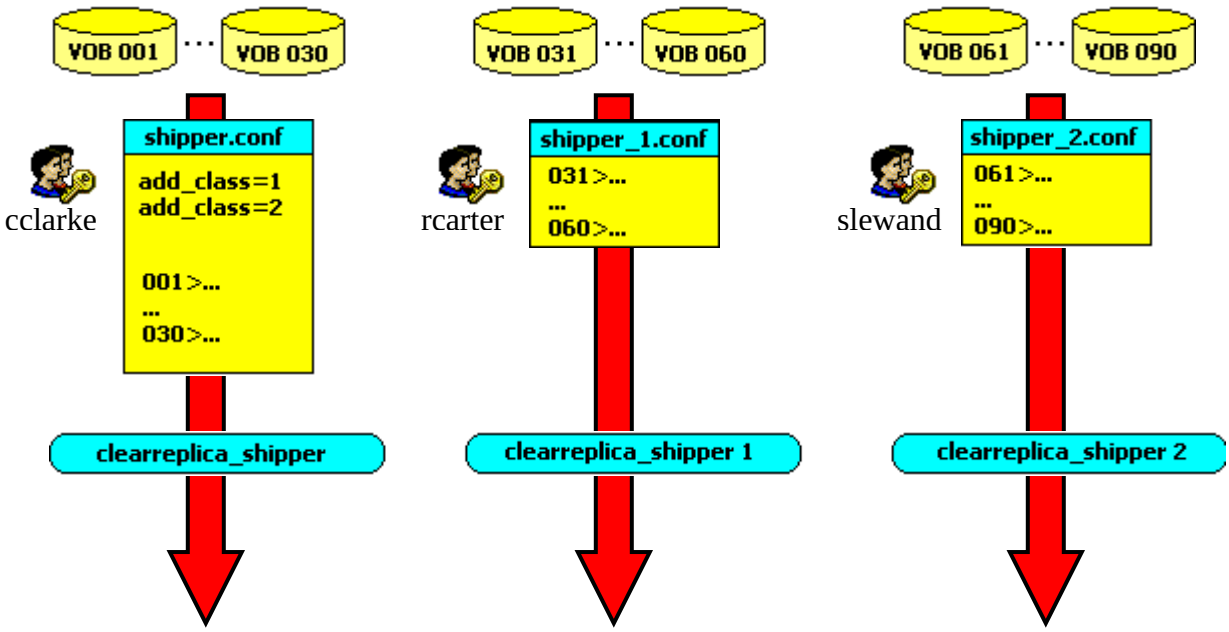
NOTE: In the instance where all keys defined are in all applicable "classed" **shipper.conf** files are also defined in the original "class-less" **shipper_conf** file is it not required to place additional **class_definitions** in the "class-less" shipper_conf file. ClearTrigger does not need to interrogate the associated "classed" shipper_conf files; it is still possible to run multiple shippers without class_definitions in the original shipper_conf. Additional class_definitions are only needed if all keys in the **replication_key** keyset are not in the original shipper_conf. file.

NOTE: The different ClearReplica shipper processes can even be run from different machines to further take advantage of available CPU and bandwidth.

Using Class_Definitions to provide project separation:

A ClearReplica Administrator can define/arrange the **shipper.conf** files so that each file has a unique set of the desired **replication_keys** so that administrator of different data manage a different file.

For example: A site with 90 VOBs might have all **replication_keys** related to VOBs 1-30 defined in **shipper.conf** managed by the administrator cclarke, all **replication_keys** related to VOBs 31-60 defined in **shipper_1.conf** managed by the administrator rcarter and all **replication_keys** related to VOBs 61-90 defined in **shipper_2.conf** managed by the administrator slewand. Administrators can now run their associated clearreplica_shipper process at times or with configuration settings that suit them.



Now replication can be managed differently by three people or groups that have different replication needs in frequency, time or access.

NOTE: In the instance where all keys defined are in all applicable "classed" **shipper.conf** files are NOT defined in the original "class-less" **shipper_conf** file is it required to place additional **class_definitions** in the "class-less" shipper_conf file. ClearTrigger does need to interrogate the associated "classed" shipper_conf files. Additional class_definitions are needed until the complete **replication_key** keyset is read.

NOTE: The different ClearReplica shipper processes can even be run from different machines to further take advantage of available CPU and bandwidth.

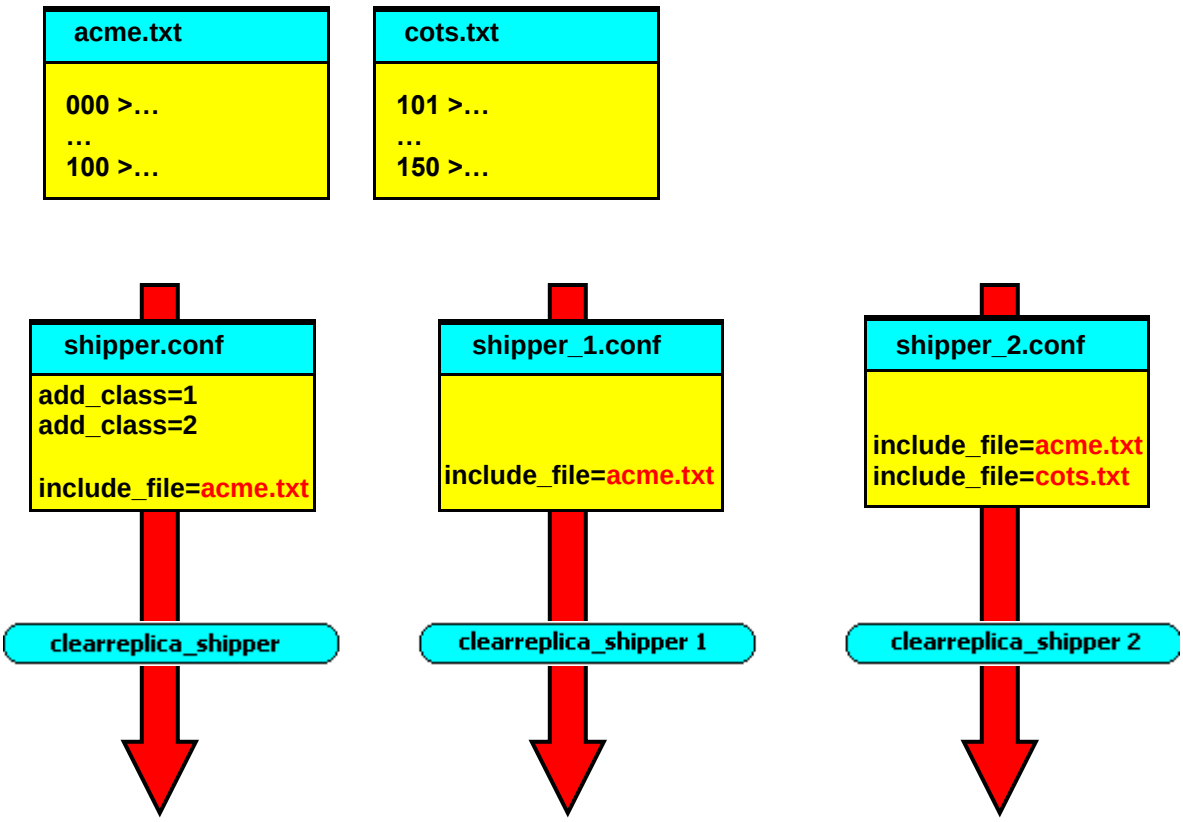
Using shipper_conf include files

A ClearReplica Administrator can define/arrange several replication keys that they wish to place in a series of **shipper.conf** files so that each file has this set of **replication_keys** without the administrator having to modify or maintain each file. The administrator can place such keys in a normal file that can then be included in the **shipper.conf** files.

Multiple **include files** are allowed in **shipper.conf** files.

For example: A site with many VOBs might have all **replication_keys** 1-100 defined in **acme.txt** and additional replication keys 101-200 defined in COTs.txt. It is perfectly OK to include one or more of these files in **shipper.conf** files. These files can only contain replication keys, all other data in the file is ignored. The contained keys are treated as if they were in the *original* **shipper.conf** file.

So, given the example depicted below, that the processing demand for keys 000-100 are spread equally over the three (3) shipper process that are associated with the shipper.conf files; ***shipper.conf***, ***shipper_1.conf*** and ***shipper_2.conf***. While the keys in the cots.txt file are only processed via the shipper process associated with ***shipper_2.conf***.



Now it is easier to maintain **replication_keys** across multiple **shipper.conf** files.

Creating the replication_definition

You should create the **replication_definition** only after you have create its associated **config_spec** file because as soon as you create the replication_definition in the **shipper.conf** file it is "active" and replication starts at the next appropriate replication time if the ClearReplica "shipper" is running.

Creating a **replication_definition** defines:

- which VOB to replicate,
- which portion of that VOB (all of the VOB or sub-directory of the VOB)
- the destination VOB for the replicated data
- where in the destination VOB to place the data
- which "location" to send the packet (where destination VOB is expected to reside). **This location MUST exist prior to replication**.
- if the new packet is immediately scheduled for shipping or marked "hold for another process."
- if the new packet will contain all branches or a subset of the existing branches
- if the new packet will contain all labels or a subset of the existing labels
- if the new packet will contain all attributes or a subset of the existing attributes

Replication_definitions are placed in the **shipper.conf** file located in the **{depot}/m_bay/config** directory and are of the form:

replication_id>[unix_vob_tag]>[windows_vob_tag]>send_portion>receive_vob_tag>receive_portion>destinations [>ship_or_hold] [>branch_filter][>label_filter][>attribute_filter]

Definition Field	Field Syntax
replication_id	["000"..."9999"]
unix_vob_tag	the Unix VOB tag (required if there are local UNIX clients for the VOB)
windows_vob_tag	the Windows VOB tag (required if there are local Windows clients for the VOB. (normalized to OS)
send_portion	the relative portion of the VOB to send or '.' for whole VOB. (normalized to OS)
receive_vob_tag or **deployment directory**	the required VOB tag for the receiving machine the format of this VOB tag should be consistent with the OS of the receiving machine and any hostmap definitions for that location as per the associated entry in the ClearReplica Hostmaps dir. If the field starts with **"/[D]"** or **"\[D]"**then the field is a **deployment directory** and the Destination directory on the "**deployed to**" machine should be indicated. If the field starts with **"/[d]"** or **"\[d]"**then this indicates a "**soft deployment**" (empty files are not packaged or sent to the destination machine). The path provided must be an absolute path on the destination machine starting with '/' for a UNIX machine (i.e. /docs/product) or a drive resident path for a Windows machine (i.e. c:\docs\product). **NOTE: UNC paths are not supported for deployment directories.**
receive_portion	the relative destination directory of the receiving VOB or '.' for top of VOB. If the previous field is a **deployment directory** then this field should be **'.'.**
destination	destination(s) to receive defined packet.
ship_or_hold	***Optional:*** "ship" or "hold". Determines if newly created packets for this key are created in the depots ***_ship_out*** directory and immediately scheduled for shipping or placed in the ***_hold_*****out** to wait for some other process or means (i.e. floppy transfer).

	NOTE: This is also useful when Replicating to the Same ClearReplica Region to avoid unneeded "receiver transport" processing.
branch_filter	***Optional:*** **"[E]"** or **"[O]"**. Determines if newly created packets for this key contain all existing branches/branch patterns or some subset thereof. If the field starts with **"[E]"** then this is an "exception" filter and **[E]** is immediately followed with a comma separated list of branches or simple branch patterns not to include in the packet ("main" cannot be "excluded"). If the field starts with **"[O]"** then this is an "only" filter and **[O]** is immediately followed with a comma separated list of the only branches or simple branch patterns to include in the packet ("main" is always assumed in the list)
label_filter	***Optional:*** **"[E]"** or **"[O]"**. Determines if newly created packets for this key contain all existing labels/label patterns or some subset thereof. If the field starts with **"[E]"** then this is an "exception" filter and **[E]** is immediately followed with a comma separated list of labels or simple label patterns not to include in the packet. If the field starts with **"[O]"** then this is an "only" filter and **[O]** is immediately followed with a comma separated list of the only labels or simple label patterns to include in the packet. This does not prevent version themselves from being replicated, just the associated label metadata.
attribute_filter	***Optional:*** **"[E]"** or **"[O]"**. Determines if newly created packets for this key contain all existing attribute/ attribute patterns or some subset thereof. If the field starts with **"[E]"** then this is an "exception" filter and **[E]** is immediately followed with a comma separated list of attributes or simple attribute patterns not to include in the packet. If the field starts with **"[O]"** then this is an "only" filter and **[O]** is immediately followed with a comma separated list of the only attributes or simple attribute patterns to include in the packet. This does not prevent version themselves from being replicated, just the associated attribute metadata.

For example: Given the "shipper" location "atlanta" has these entries in its **shipper.conf** file.

Example replication_definition	What it does
102>/vobs/m2>\m2>.>\m2>.>japan	Replicate the whole "m2" VOB from here to the location "japan"
108>/vobs/m1>\m1>.>\m1_bup>.>atlanta	Replicate the whole "m1" VOB from here to a VOB "m1_bup" in this same location (region).
109>/vobs/m1>\m1>bins>\offsite>deliver>india	Replicate the "bins" directory in the "m1" VOB to the "deliver" directory in the "offsite" VOB at the "india" location.
110>/vobs/m1>\m1>bins>\[D]c:\offsite>.>london	Deploy the "bins" directory in the "m1" VOB to the "c:\offsite" directory on a machine in the "london" location.
112>/vobs/m1>\m1>.>\offsite>.>ohio>hold	Replicate "m1" VOB to the "offsite" VOB at the "ohio" location. Do not send the packet, place it in the ***_hold_*****out** directory to be sent later.
113>/vobs/m1>\m1>.>\m1_bup>.>atlanta>hold	Replicate "m1" VOB to the "onsite" VOB at thethis location. Do not send the packet, place it in the ***_hold_*****out** directory. **Note:** When sending to the same region you can avoid unneeded "receiver transport" processes (ftp) by sending packets to the ***_hold_*****out** directory instead of the default ***_ship_*****out** directory and also making the ***_hold_*****out** a ClearReplica"receiving" directory.
114>/vobs/m1>\m1>.>\offsite>.>tampa>ship>[E]test,qa,build_*	Replicate "m1" VOB to the "offsite" VOB at the "tampa" location. Do not include element versions on the "**test**", "**qa**" or any branches that ***start*** with "**build_**" in the "replication".
115>/vobs/m1>\m1>.>\qa>.>orlando>ship>[O]test,qa,*_passed	Replicate "m1" VOB to the "qa" VOB at the "orlando" location. Only include element versions on the "test", "qa" or any branches that ***end*** in "**_passed**" in the "replication".
116>/vobs/m1>\m1>.>\qa>.>sfo>ship>>[O]QA,BETA,*_BLD_*	Replicate "m1" VOB to the "qa" VOB at the "sfo" location. Only include label metadata for the QA, BETA or any labels ***with*** the string "**_BLD_**" in its name in the "replication".
117>/vobs/m1>\m1>.>\qa>.>sfo>ship>>[E]DEV_BLD,BLD_*	Replicate "m1" VOB to the "qa" VOB at the "sfo" location. Do not include label metadata for the DEV_BLD labels or any labels that ***start*** with

	"**BLD_**" in the "replication"

There are more detailed **replication_definition examples** in the next section.

Replication/Deployment Definition Examples

This section provides some examples of ClearReplica **replication definitions**. Syntax is not discussed in this section. Example replication scenarios are described and a replication_definition that satisfies that scenario is provided. The examples are grouped into the sub-sections "**on-site replication**" and "**off-site replication**" and "**replication portions of VOBs**" with examples of other types included in each sub-section.

Onsite "Replication": Replicating within the same region

ClearReplica can replicate VOBs or VOB portions not only between different regions, but it can also "replicate" VOB data between VOBs in the "same" region, even the "same" VOB. Replication between VOBs or directories in the same "region" is referred to as "**on-site replication**".

NOTE: You can increase performance when replicating to the same ClearTrigger region by making your ***_hold_*out** directory also a "receiving" directory. When you do this not actual "receiver transport" processes (ftp) are ***not*** needed to "receive" the packets.

In the following sub-sections, examples of "on-site replication" scenarios are described and **replication definitions** satisfying that scenario is provided.

"Replicate" VOBs to the same location (making hot version backups):

Assuming that the shipping location is "atlanta" these **replication_definitions** provide "replication" of VOB within the same region. The backup VOBs could be used for element version recovery or "read only" VOBs or another development group.

102>/vobs/m1>\m1>.>/vobs/m1_backup>.>atlanta
103>/vobs/m2>\m2>.>/vobs/m2_backup>.>Atlanta

"Replicate" VOBs to the same location (combining VOBs):

Assuming that the shipping location is "atlanta" these **replication_definitions** provide "replication" of VOB within the same region and at the same time combine the VOBs to a single hot backup VOB. The backup VOB could be used for element version recovery or "read only" VOB or another development group.

102>/vobs/m1>\m1>.>/vobs/vob_backup>vob_m1>atlanta
103>/vobs/m2>\m2>.>/vobs/vob_backup>vob_m2>Atlanta

"Replicate" VOB portions to the same location (distribution areas):

Assuming that the shipping location is "atlanta" these **replication_definitions** provide "replication" of VOB portions within the same region. The backup VOB could be used for element version recovery or "read only" VOB or some distribution area.

101>/vobs/m1>\m1>web_pages>/vobs/production_site>.>atlanta
102>/vobs/m2>\m2>web_pages/asp>/vobs/support_site>help/asp>atlanta
107>/vobs/m2>\m2>FAQ_pages/asp>/vobs/support_site>support/FAQ>atlanta
109>/vobs/m3>\m3>secure_pages/html>/vobs/support_site>commercial>Atlanta

Selective Branch "Replication" to the same location (branch exclusion):

Assuming that the shipping location is "atlanta" these **replication_definitions** provide "selective branch replication" of VOB portions within the same region. The select branches are "excluded" from the replication.

101>/vobs/m1>\m1>web_pages>/vobs/qa>.>atlanta>ship>[E]prod,int,qa_*,*_failed
102>/vobs/m1>\m1>web_pages>/vobs/pre_prod>.>atlanta>hold>[E]release,int,*_cr_*

Selective Branch "Replication" to the same location (branch inclusion):

Assuming that the shipping location is "atlanta" these **replication_definitions** provide "selective branch replication" of VOB portions within the same region. The select branches are the only ones "included" in the replication ("main" is always assumed included).

101>/vobs/m1>\m1>web_pages>/vobs/qa>.>atlanta>ship>[O]qa,pre_prod,*_passed
102>/vobs/m1>\m1>web_pages>/vobs/pre_prod>.>atlanta>hold>[O]qa,int,night_build_*

Selective Label "Replication" to the same location (label exclusion):

Assuming that the shipping location is "atlanta" these **replication_definitions** provide "selective label replication" of VOB portions within the same region. The select labels are "excluded" from the replication.

101>/vobs/m1>\m1>web_pages>/vobs/qa>.>atlanta>ship>>[E]QA_PASS,PRE_PROD,*_FAILED
102>/vobs/m1>\m1>web_pages>/vobs/pre_prod>.>atlanta>hold>>[E] PROD,QA,CR_*

Selective Label "Replication" to the same location (label inclusion):

Assuming that the shipping location is "atlanta" these **replication_definitions** provide "selective label replication" of VOB portions within the same region. The select labels are the only ones "included" in the replication.

101>/vobs/m1>\m1>web_pages>/vobs/qa>.>atlanta>ship>>[O]QA_PASS, PRE_PROD,*_PASSED
102>/vobs/m1>\m1>web_pages>/vobs/pre_prod>.>atlanta>hold>>[O]PROD,QA,DEPOLY_*

Onsite "Replication": Replicating to a different region

ClearReplica can replicate VOBs or VOB portions not only within the same region, but if can also "replicate" VOB data between VOBs in "different" regions, it can even "replicate" data from a VOB in one region to a differently named VOB (or VOBs) in a different region. Replication between VOBs or directories in different "regions" is referred to as "**off-site replication**". In the following sub-sections, examples of "off-site replication" scenarios are described and **replication definitions** satisfying that scenario is provided.

"Replicate" VOBs to different locations:

Assuming that the shipping location is "atlanta" these **replication_definitions** provide "replication" of VOBs to different regions. The backup VOBs could be used for element version recovery or "read only" VOBs or another development group, the data can even be placed into a VOB of a different name on the "receiving" location.

102>/vobs/m1>\m1>.>/vobs/m1>.>china japan washington
103>/vobs/m2>\m2>.>/vobs/m2>.>india
104>/vobs/soccer>\ soccer >.>/vobs/football>.>london

"Replicate" VOBs to different locations (combining VOBs):

Assuming that the shipping location is "atlanta" these **replication_definitions** provide "replication" of VOB to different regions and at the same time combine the VOBs to a single hot backup VOB. The backup VOB could be used for element version recovery or "read only" VOB or another development group.

102>/vobs/car>\car >src>/vobs/rec_vehicle>car_src >japan
103>/vobs/boat>\boat >src>/vobs/rec_vehicle >boat_src >japan
104>/vobs/cycle>\cycle >src>/vobs/rec_vehicle >cycle_src > japan

"Replicating" portions of VOBs or single files:

ClearReplica can be used to "replicate" a portion or portions (including a single file) of a VOB and those portions can be reassembled in a different or many different VOBs.

"Replicate" a portion of a VOB to many other VOBs

Assuming that the shipping location is "atlanta" these **replication_definitions** provide "replication" of VOB portions to other VOBs in different regions.

102>/vobs/engine>\engine>src>/vobs/car>engine >japan detroit
103>/vobs/engine>\engine>src>/vobs/truck>engine >japan charlotte
104>/vobs/engine>\engine>src>/vobs/bus>engine > washington

"Replicate" portions of different VOBs to a single VOB

Assuming that the shipping location is "atlanta" these **replication_definitions** provide "replication" of VOB portions from different VOBs in one region to a single VOB in another region.

102>/vobs/car>\car >src>/vobs/rec_vehicle>land >japan
103>/vobs/boat>\boat >src>/vobs/rec_vehicle >water >japan
104>/vobs/cycle>\cycle >src>/vobs/rec_vehicle >land > japan

"Replicate" a "single file" in a VOB to many other VOBs

Assuming that the shipping location is "atlanta" these **replication_definitions** provide the "replication" of a single VOB file to other VOBs in different regions. Note that the file is identified in the "origination" portion of the key while the directory it will be "replicated" to is in the "destination" portion.

102>/vobs/engine>\engine>src/foo.c>/vobs/car>engine >japan detroit
103>/vobs/engine>\engine>src/bar.c>/vobs/truck>engine >japan charlotte
104>/vobs/engine>\engine>src/stat.doc>/vobs/bus>engine > Washington

NOTE: The "shipping" VOBs must have **ClearTrigger 12.0** or higher on them to properly create "single file" replication packets.

"Deploying" portions of VOBs or single files:

ClearReplica can be used to "deploy" a portion or portions (including a single file) of a VOB to flat files systems on both local and remote machines (even if those machines do not have ClearCase). In order to "deploy" to a non-VOB (flat filesystem) you must indicate that your **receive_vob_tag** is a **deployment directory in the replication definition.**

Deployment Directories begin with either the **Windows Deployment Directory String** ("\[D]") or the **UNIX Deployment Directory String** ("/[D]") and the **receiver portion** key should always contain a single "**.**" like the examples in the table below:

Deployment Directory examples	Meaning
>/[D]/docs/car>.>	Deploy to the **/docs/car** directory on the **Unix** machine at the destination site.
>/[D]/etc/>.>	Deploy to the **/etc/** directory on the **Unix** machine at the destination site.
>\[D]c:\docs\car>.>	Deploy to the **c:\docs\car** directory on the **Windows** machine at the destination
>\[D]c:\etc>.>	Deploy to the **c:\etc** directory on the **Windows** machine at the destination site.
>/[d]/etc/>.>	Deploy to the **/etc/** directory on the **Unix** machine at the destination site, but do package empty files.
>\[d]c:\docs\car>.>	Deploy to the **c:\docs\car** directory on the **Windows** machine at the destination site, but do not package empty files.

"Deploy" a portion of a VOB to many other machines/locations

Assuming that the shipping location is "atlanta" these **replication_definitions** "deploy" VOB portions to other machines/location.

102>/vobs/engine>\engine>src>/[D]/runtime/car>.>japan detroit
103>/vobs/engine>\engine>src>/[D]/runtime/truck>.>japan charlotte
104>/vobs/engine>\engine>src>/\[D]c:\runtime/bus>.> washington

"Deploy" a "single file" in a VOB to many other VOBs

Assuming that the shipping location is "atlanta" these **replication_definitions** "deploy" a single VOB file to other VOBs in different regions. Note that the file is identified in the "origination" portion of the key while the directory it will be "deployed" to is in the "destination" portion.

102>/vobs/engine>\engine>src/foo.c>/[D]/motors/car>.>japan detroit
103>/vobs/engine>\engine>src/bar.c>/[D]/wheels/truck>.>japan charlotte
104>/vobs/engine>\engine>src/stat.doc>\[D]c:\test\bus>engine > Washington

NOTE: The "shipping" VOBs must have **ClearTrigger 12.0** or higher on them to properly create "single file" deployment packets.

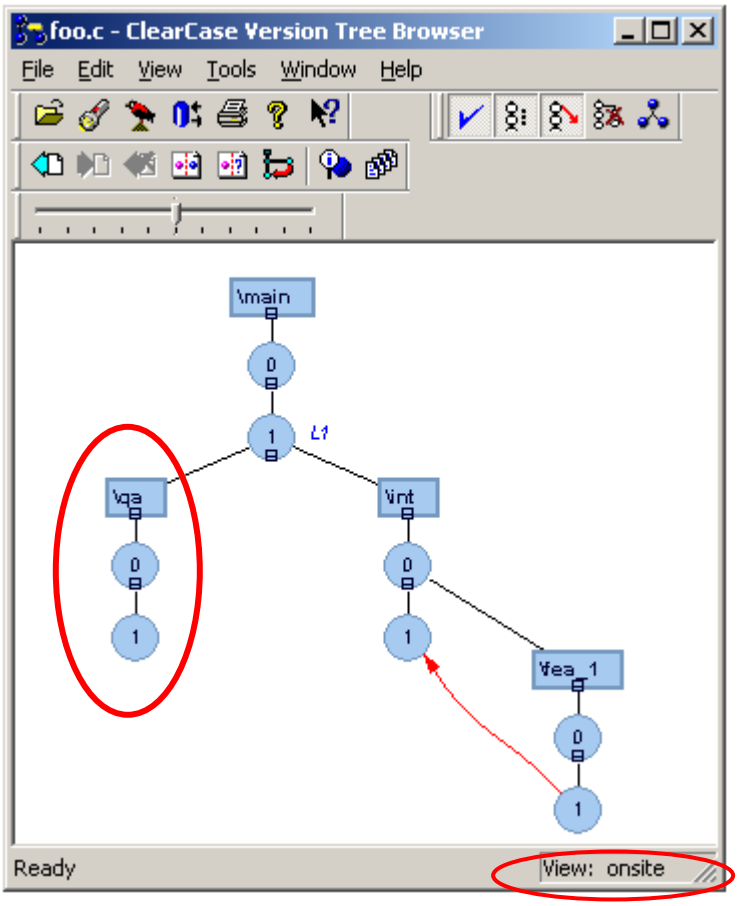
"Restricted Branch Replication"

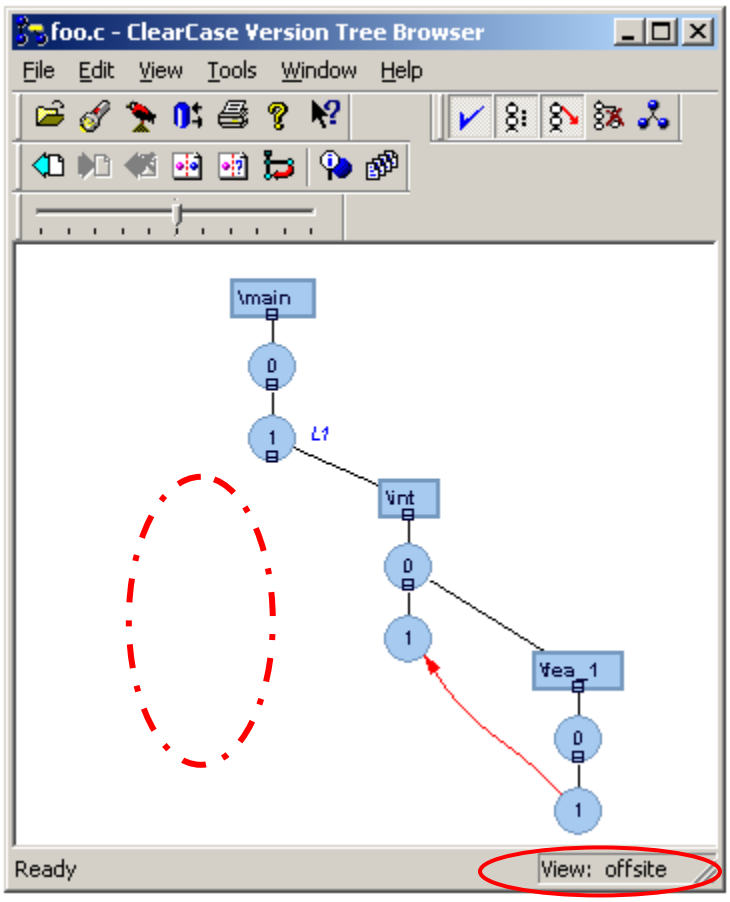
ClearReplica can be used to limit the branches that are "replicated" to other ClearReplica replicas (both locally and remotely). To limit what branches are sent in a replication packet, add an optional **Branch Filter** to a **replication definition**. This feature is extremely useful for saving corporate bandwidth as one can replicated just the branch shared between two sited (i.e. an "integration" or "release" branch) rather than all of the branches used during development.

Branch Filters begin with either the **Branch Exclusion String** ("[E]") or the **Branch Only String** ("[O]") and the comma separated list of branches like the examples in the table below:

Branch Filter Examples	Meaning
>[E]qa	***Exclude*** element versions that exist on the "qa" branch from Replication.
>[O]int,fea_1	***Only include*** versions that exist on the "int" or "fea_1" branches during Replication.

Either example above will result in element versions on the "**qa**" branch not being replicated to the **offsite** location.





NOTE: You never need to include "main" on either list. The "main" branch is always included for [O] filters while "main" cannot be "excluded" from [E] filters.

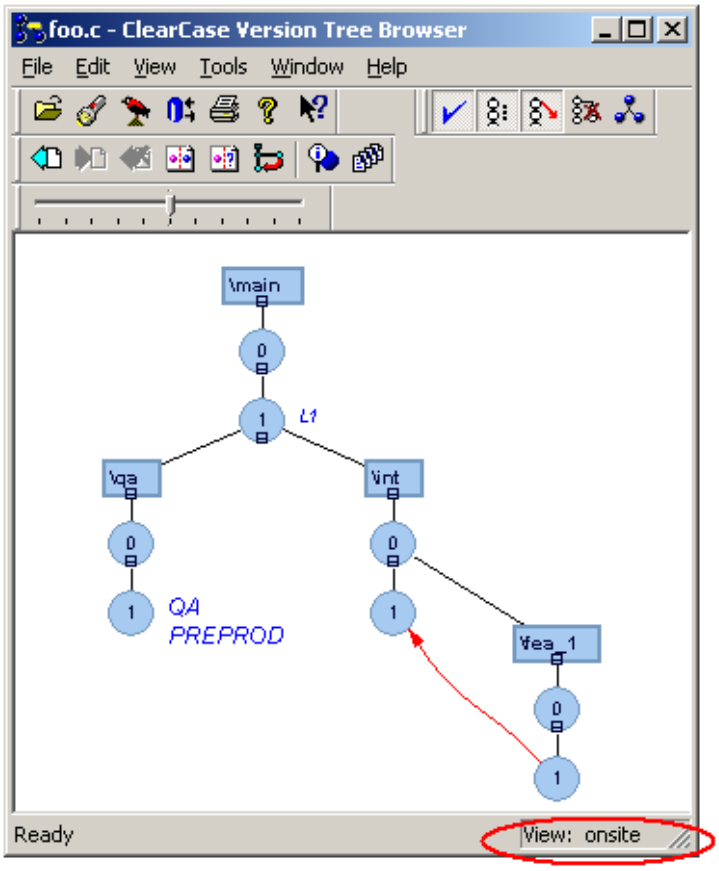
"Restricted Label Replication"

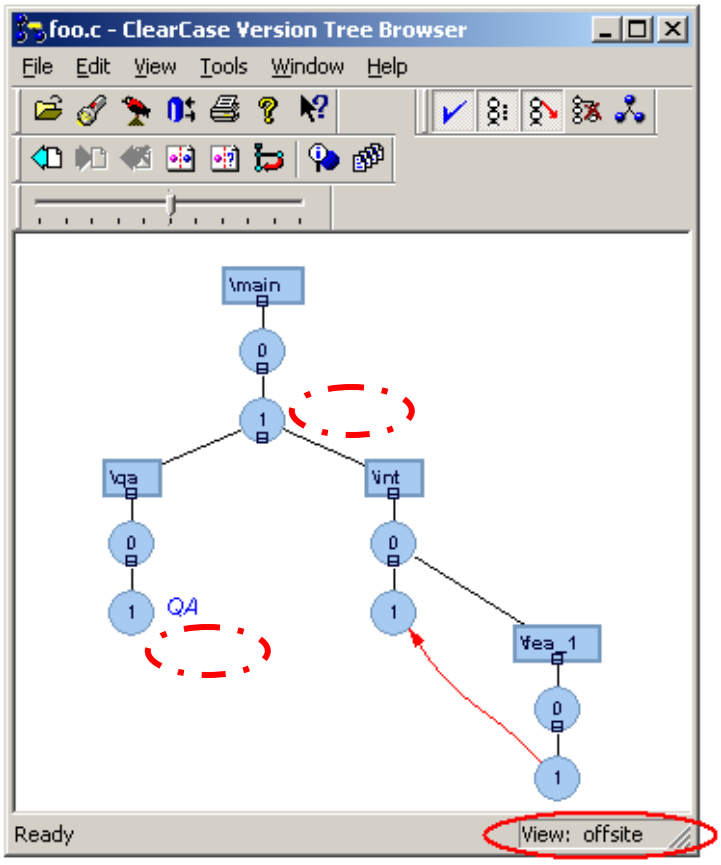
ClearReplica can be used to limit the labels that are "replicated" to other ClearReplica replicas (both locally and remotely). To limit what labels are sent in a replication packet, add an optional **Label Filter** to a **replication definition**. This feature is extremely useful in saving corporate bandwidth as one can replicated just the desired label metadata shared between two sited (i.e. an "QA" or "RELEASE" label) rather than all of the labels used during development.

Label Filters begin with either the **Label Exclusion String** ("[E]") or the **Label Only String** ("[O]") and the comma separated list of labels like the examples in the table below:

Label Filter Examples	Meaning
>[E]L1,PREPROD	***Exclude*** label metadata for the "L1" and "PREPOD" labels from Replication.
>[O]QA	***Only include*** label metadata that exist for the "L1" or "PROD" labels during Replication.

Either example above will result in the "**QA**" label being replicated, but not the **L1** or **PREPROD** labels being replicated to the **offsite** location.





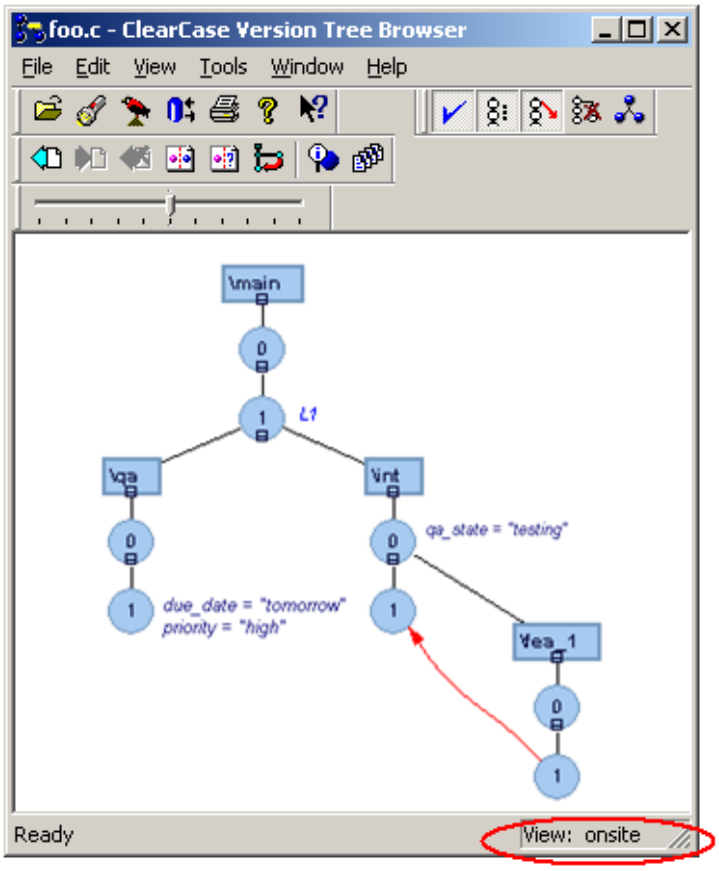
"Restricted Attribute Replication" (new to 12.0)

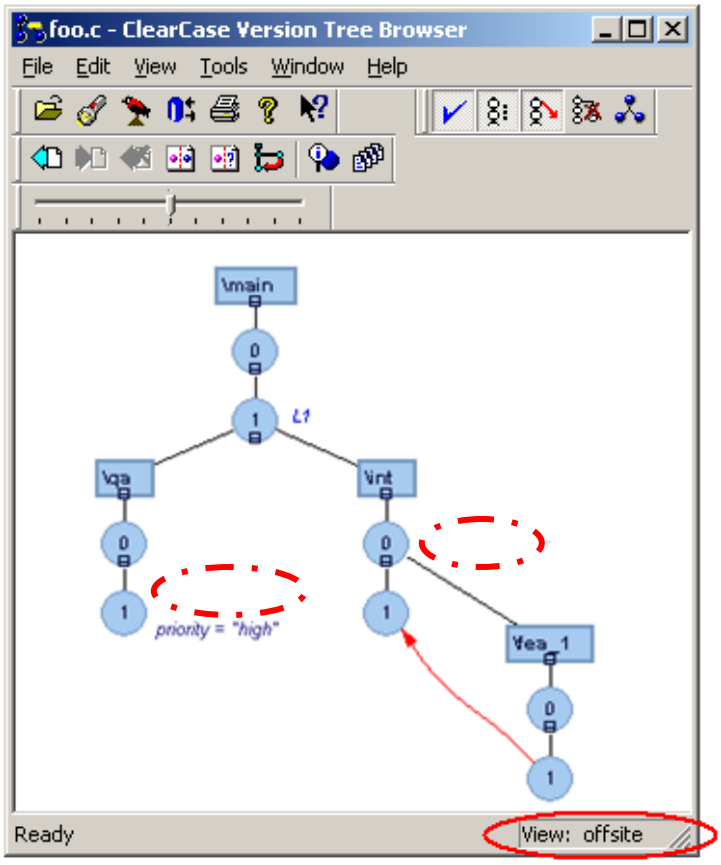
ClearReplica can be used to limit the attributes that are "replicated" to other ClearReplica replicas (both locally and remotely). To limit what attributes are sent in a replication packet, add an optional **Attribute Filter** to a **replication definition.** This feature is extremely useful in saving corporate bandwidth as one can replicated just the desired attribute metadata shared between two sited (i.e. an "qa_state" or "due_date" attribute) rather than all of the attributes used during development.

Attribute Filters begin with either the **Attribute Exclusion String** ("[E]") or the **Attribute Only String** ("[O]") and the comma separated list of attributes like the examples in the table below:

Label Filter Examples	Meaning
>[E]qa_state,due_date	***Exclude*** attribute metadata for the "qa_state" and "due_date" attributes from Replication.
>[O]priority	***Only include*** attribute metadata that exist for the "priority" attributes during Replication.

Either example above will result in the "**priority**" attribute being replicated, but not the "**qa_state**" or "**due_date**" labels being replicated to the **offsite** location.





"Exclusion" of portions of VOBs or single files:

There are 3 ways to "exclude" portions of VOBs that are replicated, by "**Key Omission**", by "**Config. Spec Omission**" and by "**Config. Spec. Exclusion**" rules. ClearReplica can be used to "replicate" a portion or portions (including a single file) of a VOB and those portions can be reassembled in a different or many different VOBs.

Exclude by "Key Omission"

If you don't ask for it to be replicated it is not replicated so you could define a replication definition that only replicated the **"src"** directory in a VOB which will not replicate the **"lib"** directory which you did not specifically ask to be replicated. There are two ways to perform file or directory exclusion. For example: assuming that the shipping location is "atlanta" this **replication_definition** provide the "replication" of the "src" directory (and it's children) in the VOB /vobs/engine, but not the "lib" directory (/vobs/engine/lib) because it is not below the "src" directory.

102>/vobs/engine>\engine>src>/vobs/engine>src >japan

Exclude by "Config. Spec. Omission"

If you choose a config. spec. that does not "see" any version of the file or directory you do not want replicated then it is not replicated so you could define the associated replication key's config. Spec. in a manner that does not see any version of the file and it will not get replicated. For example: assuming that the shipping location is "atlanta" this **replication_definition** provide the "replication" of the "src" directory (and it's children) in the VOB /vobs/engine, but not the "src/covers" directory (/vobs/engine/src/covers) because it is not below visible with this config. Spec.

102>/vobs/engine>\engine>src>/vobs/engine>src >japan

with associated **102.cs** file of:

element * /main/integration/LATEST
element * /main/LATEST –mkbranch integration

this would prevent the "src/covers" directory (initially created and being developed on the /main/integration/future branch) from being replicated until the "src" directory was merged from /main/integration/future to /main or /main/integration.

Exclude by "Config. Spec. Exclusion"

This is very similar to "Config. Spec. Omission" in that you choose a config. Spec. that does not "see" any version of the file or directory you do not want replicated, but it is more direct in that you specify that you do not want the element replicated in the config. Spec. using a "-none" rule. For example: assuming that the shipping location is "atlanta" this **replication_definition** provide the "replication" of the "src" directory (and it's children) in the VOB /vobs/engine, but not the "src/covers" directory (/vobs/engine/src/covers).

102>/vobs/engine>\engine>src>/vobs/engine>src >japan

with associated **102.cs** file of:

element /vobs/engine/src/covers –none
element * /main/LATEST

this would prevent the “src/covers” directory from being replicated because you are essentially informing ClearCase not to make this directory visible in the replication view.

Creating ClearReplica Replication Triggers

The ClearReplica replication process can be further automated and augmented by the addition of **Replication Triggers Definitions**. Optional Replication Triggers Definitions can be added to automatically run scripts before or after either sending or receiving "Replication" packets. You can use any in-house script to perform custom logging, send email notification, install/execute deployed files, lock VOBs or even redirect packets by just adding a ClearReplica **Replication Trigger**.

ClearReplica has four **(4) replication trigger types** defined in either of the ClearReplica Configuration Files; **shipper.conf** or **receiver.conf** help you manage your distributed environment:

pre_ship_trigger
post_ship_trigger
pre_receive_trigger
post_receive_trigger

The scripts or programs called by these triggers should not require user interaction in any way as they affect the replication/deployment process, which is usually a completely automated and unassisted process. Scripts that cause dialogs to appear and wait for user input will cause the replication/deployment process to want for that input. All undirected writes to **stdout** and **stderr** are redirected to **/dev/null** (UNIX) or **NUL** (Windows), however you can redirect them to files from within the scripts (including the standard ClearReplica logs if desired).

The trigger program or script itself must reside in the ***m_bay/_triggers*** directory in the associated ClearTrigger depot. Only the script or program name is included in the definition, ClearReplica process will look for that program only in the ***m_bay/_triggers*** directory.

Replication Trigger scripts (if any) are defined in the associated configuration file as described in the table below:

Trigger Definition (optional)	Located in configuration file	Details
pre_ship_trigger	**shipper.conf**	Optional script or executable that is called **immediately before** any requested shipping packet is created.
post_ship_trigger	**shipper.conf**	Optional script or executable that is called **immediately after** any requested shipping packet is created and shipped (or created and placed in the **_hold_out** directory).
pre_receive_trigger	**receiver.conf**	Optional script or executable that is called **immediately before** any received packet is processed.
post_receive_trigger	**receiver.conf**	Optional script or executable that is called **immediately after** any received packet is processed, even if the processed packet did not complete (providing an opportunity to clean up any data changes made in the **pre_receive_trigger_script**).

Additional **ClearReplica Replication Environmental Variables** are defined by the shipper or receiver processes and available within your trigger script to allow you to write customized replication/deployment process automation triggers.

Pre_ship_trigger:

The **pre_ship_trigger** definition is optionally defined in the **shipper.conf** and helps automate or augment the replication or deployment process.

The definition can designate a single trigger script (or executable) located in the ***m_bay/_triggers*** directory in the associated ClearTrigger depot that is to be called **immediately before** any requested shipping packet is created. If the script exits with a zero return code then the packet is created and immediately shipped (unless it is destined to the **_hold_out** directory). If the script exits with a non-zero (failed) status then the packet is not built and the appropriate ***error*** message is written to the ClearReplica **shipper logs**.

The scripts or programs called by these triggers should not require user interaction in any way as they affect the replication/deployment process, which is usually a completely automated and unassisted process. Scripts that cause dialogs to appear and wait for user input will cause the replication/deployment process to want for that input. All undirected writes to **stdout** and **stderr** are redirected to **/dev/null** (UNIX) or **NUL** (Windows), however you can redirect them to files from within the scripts (including the standard ClearReplica logs if desired).

Additional **ClearReplica Replication Environmental Variables** are defined by the shipper process and available within your trigger script to allow you to write customized shipper process automation triggers.

Post_ship_trigger:

The **post_ship_trigger** definition is optionally defined in the **shipper.conf** and helps automate or augment the replication or deployment process.

The definition can designate a single trigger script (or executable) located in the ***m_bay/_triggers*** directory in the associated ClearTrigger depot that is to be called **immediately after** any requested shipping packet is created and shipped (or created and placed in the **_hold_out** directory), even if the packet did not process successfully (providing an opportunity to clean up any data changes made in the **pre_ship_trigger**). If the script exits with a zero return code then the packet is created and immediately shipped (unless it is destined to the **_hold_out** directory If the script exits with a non-zero (failed) status then the appropriate ***warning*** message is written to the ClearReplica **shipper logs**.

The scripts or programs called by these triggers should not require user interaction in any way as they affect the replication/deployment process, which is usually a completely automated and unassisted process. Scripts that cause dialogs to appear and wait for user input will cause the replication/deployment process to want for that input. All undirected writes to **stdout** and **stderr** are redirected to **/dev/null** (UNIX) or **NUL** (Windows), however you can redirect them to files from within the scripts (including the standard ClearReplica logs if desired).

Additional **ClearReplica Replication Environmental Variables** are defined by the shipper process and available within your trigger script to allow you to write customized shipper process automation triggers.

Pre_receive_trigger:

The **pre_receive_trigger** definition is optionally defined in the **receiver.conf** and helps automate or augment the replication or deployment process.

The definition can designate a single trigger script (or executable) located in the ***m_bay/_triggers*** directory in the associated ClearTrigger depot that is to be called **immediately before** any received packet is processed. If the script exits with a zero return code then the packet is processed.). If the script exits with a non-zero (failed) status then the packet is not processed (tossed) and the appropriate ***error*** message is written to the ClearReplica **receiver logs**.

The scripts or programs called by these triggers should not require user interaction in any way as they affect the replication/deployment process, which is usually a completely automated and unassisted process. Scripts that cause dialogs to appear and wait for user input will cause the replication/deployment process to want for that input. All undirected writes to **stdout** and **stderr** are redirected to **/dev/null** (UNIX) or **NUL** (Windows), however you can redirect them to files from within the scripts (including the standard ClearReplica logs if desired).

Additional **ClearReplica Replication Environmental Variables** are defined by the shipper process and available within your trigger script to allow you to write customized receiver process automation triggers.

Post_receive_trigger:

The **post_ship_trigger** definition is optionally defined in the **receiver.conf** and helps automate or augment the replication or deployment process.

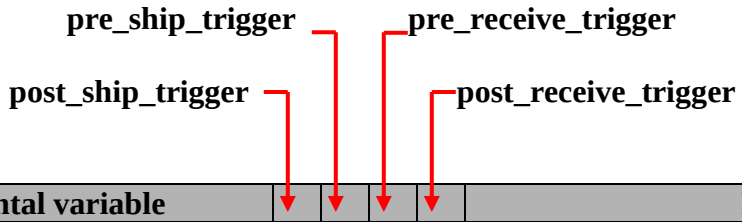
The definition can designate a single trigger script (or executable) located in the ***m_bay/_triggers*** directory in the associated ClearTrigger depot that is to be called **immediately after** any processed packet is processed, even if the packet did not process successfully (providing an opportunity to clean up any data changes made in the **pre_receive_trigger**). If the script exits with a non-zero (failed) status then the appropriate ***warning*** message is written to the ClearReplica **receiver logs**.

The scripts or programs called by these triggers should not require user interaction in any way as they affect the replication/deployment process, which is usually a completely automated and unassisted process. Scripts that cause dialogs to appear and wait for user input will cause the replication/deployment process to want for that input. All undirected writes to **stdout** and **stderr** are redirected to **/dev/null** (UNIX) or **NUL** (Windows), however you can redirect them to files from within the scripts (including the standard ClearReplica logs if desired).

Additional **ClearReplica Replication Environmental Variables** are defined by the shipper process and available within your trigger script to allow you to write customized receiver process automation triggers.

ClearReplica Replication Trigger Environmental Variables

This section lists the ClearReplica Replication Environmental Variables that are set and available during the execution of ClearReplica Replication Trigger Scripts that are called as a result of one of the ClearReplica Trigger Types. Variables recently added or changed in 10.0 are so marked*.



Environmental variable					Meaning/Example
CLEARREPLICA_DEPLOYMENT_DIR				D	If and only if the processed packet was a "Deployment" packet this has the **destination directory** of the deployed to location.
CLEARREPLICA_KEY_ID	X	X	X	X	Key number associated with the **replication definition** (i.e. **"102"** for the definition "**102**>/vobs/demo>\demo>.>/[D]c:\temp_deploy>.>usa**"**
CLEARREPLICA_LOGGING_DIR	X	X	X	X	The current ClearReplica Logging directory
CLEARREPLICA_PACKAGE_NAME	X	X	X	X	The name of the packet that caused this script to fire (i.e. **"mship.102.atlanta.london"**)
CLEARREPLICA_PACKAGE_TYPE	X	X		X	Type of packet created by the associated **replication definition** Either **"vob_to_vob"** for "**replication" type definitions** or **"vob_to_flat**" for "**deployment" type definitions**.
CLEARREPLICA_PACKET_STATUS	X	X	X	X	Status of the current packet that caused this script to fire. From: "**OK**" – Packet was successfully processed (shipped/received) "**OK with Warnings**" - Packet was successfully processed (shipped/received), but there were warnings written to logs files. "**NOK**" – Packet was NOT successfully processed "**Pending**" – always for pre_ship_trigger
CLEARREPLICA_RECEIVER_EMAIL_FILE				X	File containing the email body that includes per file version warning of versioned elements not processed during the packet. If there were not warning then this variable is unset.
CLEARREPLICA_RECEIVER_HAS_EMAIL				X	Set to "yes" if there were per file version warnings issued during the processing of the received packet, otherwise set to the "no" value. When set to "yes" the email contents as held in the file contained in the environment variable **CLEARREPLICA_RECEIVER_EMAIL_FILE**.
CLEARREPLICA_RECEIVER_VERSION			X	X	The version of clearreplica_receiver running this process
CLEARREPLICA_RECEIVER_WHICH			X	X	The receiving class of the processing receiver or not set for he "classless" (default) receiver.

CLEARREPLICA_RECEIVING_DIR			X	X	The **replication_receiver_directory** that received the current packet.
CLEARREPLICA_RECEIVING_LOCATION	X	X	X	X	The associated pneumonic that the shippers calls the receiver of the packet in the actual **replication definition** that caused the trigger to fire (i.e. **"usa"** for the definition "102>/vobs/demo>\demo>.>/[D]c:\temp_deploy>.>**usa**")
CLEARREPLICA_RECEIVING_PROTECTIONS	0	0	X	X	Protections provided by the associated **replication_receiver_directory** entry in the **receiver.conf** file. From: "**receiver**" – New element will have the (user:group) ownership that matches the current user:group of the currently running "receiver" process. "**preserve**" - New element will have the (user:group) ownership that matches the ownership of the associated elements in the original VOB at the **shipper** location. "**exactly=user:group**" – New element will have the (user:group) ownership that matches the provided user:group.
CLEARREPLICA_RECEIVING_VOB				R	If and only if the processed packet was a "Replication" packet this has the **VOB Tag** of the receiving location.
CLEARREPLICA_REPLICATION_KEY	X	X			Actual **replication definition** that caused the trigger to fire (i.e.**"102>/vobs/demo>\demo>.>/[D]c:\temp_deploy>.>usa"**)
CLEARREPLICA_SHIPPER_VERSION	X	X			The version of clearreplica_shipper running this process
CLEARREPLICA_SHIPPING_CLASS	X	X			The shipping class (0..9) of the processing shipper or not set for he "classless" (default) shipper.
CLEARREPLICA_SHIPPING_LOCATION	X	X	X	X	Associated pneumonic that the shippers calls itself (i.e. **"atlanta")**
CLEARREPLICA_SHIPPING_VOB	X	X			The VOB tag of the VOB that contains the data in the associated packet. (i.e. **"\demo"** {Windows} or **"/vobs/demo"** {Unix})
CLEARREPLICA_THIS_LOCATION	X	X	X	X	The associated pneumonic that the location calls itself (i.e. **"atlanta")**
CLEARREPLICA_TRIGGER_TYPE	X	X	X	X	Type of **replication trigger** being executed. From: "pre_ship" "post_ship" "pre_receive" "post_receive"

Whereas	Means
X	Always set for the **replica trigger type.**
D	Set for the **replica trigger type** if processing a "deployment" packet.
R	Set for the **replica trigger type** if processing a "replication" packet.

Creating ClearReplica Startup/Shutdown Triggers

The ClearReplica replication process can be further automated and augmented by the addition of **Startup/Shutdown Triggers Definitions**. Optional Start/Shutdown Triggers Definitions can be added to automatically run scripts after the successful startup or shutdown of ClearReplica shipper or receiver processes. You can use any in-house script to perform custom logging, send email notification, install/execute deployed files, lock VOBs or even redirect packets by just adding a ClearReplica **Startup or Shutdown Trigger**.

ClearReplica has two **(2) startup/shutdown trigger types** defined in either of the ClearReplica Configuration Files; **shipper.conf** or **receiver.conf** help you manage your distributed environment:

startup_trigger
shutdown_trigger

The scripts or programs called by these triggers should not require user interaction in any way as they affect the replication/deployment process, which is usually a completely automated and unassisted process. Scripts that cause dialogs to appear and wait for user input will cause the startup/shutdown process to want for that input. All undirected writes to **stdout** and **stderr** are redirected to **/dev/null** (UNIX) or **NUL** (Windows), however you can redirect them to files from within the scripts (including the standard ClearReplica logs if desired).

The trigger program or script itself must reside in the ***m_bay/_triggers*** directory in the associated ClearTrigger depot. Only the script or program name is included in the definition, ClearReplica process will look for that program only in the ***m_bay/_triggers*** directory.

Replication Trigger scripts (if any) are defined in the associated configuration file as described in the table below:

Trigger Definition (optional)	**Details (Located in shipper.conf and/or receiver.conf)**
startup_trigger	Optional script or executable that is called **immediately after** the successful startup of a ClearReplica receiver process.
shutdown_trigger	Optional script or executable that is called **immediately after** the successful shutdown of a ClearReplica shipper process

Additional **ClearReplica Start/Stop Environmental Variables** are defined by the shipper or receiver processes and available within your trigger script to allow you to write customized replication/deployment process automation triggers.

Startup_trigger:

A **startup_trigger** definition can be optionally defined in the **shipper.conf** or **receiver.conf** files and helps automate or augment the startup processes.

The definition can designate a single trigger script (or executable) located in the ***m_bay/_triggers*** directory in the associated ClearTrigger depot that is to be called **immediately after** the successful startup of a ClearReplica shipper or receiver process. If the script exits with a zero return code then the shipper or receiver process continues. If the script exits with a non-zero (failed) status then the shipper or receiver startup is considered denied by that script and the appropriate ***error*** message is written to the ClearReplica **shipper logs** or **receiver logs**.

The scripts or programs called by these triggers should not require user interaction in any way as they affect the replication/deployment process, which is usually a completely automated and unassisted process. Scripts that cause dialogs to appear and wait for user input will cause the replication/deployment process to want for that input. All undirected writes to **stdout** and **stderr** are redirected to **/dev/null** (UNIX) or **NUL** (Windows), however you can redirect them to files from within the scripts (including the standard ClearReplica logs if desired).

Additional **ClearReplica Start/Stop Environmental Variables** are defined by the shipper ad receiver processes and available within your trigger script to allow you to write customized startup/shutdown process automation triggers.

Shutdown_trigger:

A **shutdown_trigger** definition can be optionally defined in the **shipper.conf** or **receiver.conf** files and helps automate or augment the shutdown processes.

The definition can designate a single trigger script (or executable) located in the ***m_bay/_triggers*** directory in the associated ClearTrigger depot that is to be called **immediately after** the successful shutdown of a ClearReplica shipper or receiver process. If the script exits with a zero return code then the shipper or receiver process continues. If called, an appropriate ***error*** or ***success*** message is written to the ClearReplica **shipper logs** or **receiver logs** indicating the script's return code.

The scripts or programs called by these triggers should not require user interaction in any way as they affect the replication/deployment process, which is usually a completely automated and unassisted process. Scripts that cause dialogs to appear and wait for user input will cause the replication/deployment process to want for that input. All undirected writes to **stdout** and **stderr** are redirected to **/dev/null** (UNIX) or **NUL** (Windows), however you can redirect them to files from within the scripts (including the standard ClearReplica logs if desired).

Additional **ClearReplica Start/Stop Environmental Variables** are defined by the shipper process and available within your trigger script to allow you to write customized shipper process automation triggers.

ClearReplica Start/Stop Trigger Env. Variables

This section lists the ClearReplica Start/Stop Environmental Variables that are set and available during the execution of ClearReplica Start/Stop Trigger Scripts that are called as a result of one of the starting for stopping of ClearReplica Shippers or Receivers.

Environmental variable					Meaning/Example
CLEARREPLICA_LOGGING_DIR	X	X	X	X	The current ClearReplica Logging directory
CLEARREPLICA_THIS_LOCATION	X	X	X	X	Associated pneumonic that the shipper or receiver calls itself (i.e. **"atlanta")**
CLEARREPLICA_RECEIVER_ACTION			X	X	"startup" during **receiver_start** trigger or "shutdown" during **receiver_stop** trigger
CLEARREPLICA_ RECEIVER _ENDED_AT				X	Seconds since epoch at the time the receiver ended/stopped
CLEARREPLICA_ RECEIVER _VERSION			X	X	The version of clearreplica_receiver running this process
CLEARREPLICA_ RECEIVER _STARTED_AT			X		Seconds since epoch at the time the receiver started
CLEARREPLICA_RECEIVER_WHICH			X	X	The receiving class of the processing receiver or not set for he "classless" (default) receiver.
CLEARREPLICA_SHIPPER_ACTION	X	X			"startup" during **shipper_start** trigger or "shutdown" during **shipper_stop** trigger
CLEARREPLICA_SHIPPER_ENDED_AT		X			Seconds since epoch at the time the shipper ended/stopped
CLEARREPLICA_SHIPPER_VERSION	X	X			The version of clearreplica_shipper running this process
CLEARREPLICA_SHIPPER_STARTED_AT	X				. Seconds since epoch at the time the shipper started
CLEARREPLICA_SHIPPING_CLASS	X	X			The shipping class (0..9) of the processing shipper or not set for he "classless" (default) shipper.

Whereas	Means
X	Always set for the **replica trigger type.**

Setting/Changing the Replication Epoch Times

ClearReplica automatically manages its **Replication Epoch Times** so that any missed packets (because of lost connectivity) are resent when connectivity is reestablished. You will only need to modify the Replication Epoch Times for a **replication definition** if you wish repeat a packet or more commonly wish to start the replication at a particular time when creating the initial replication packet for a older or larger VOB that has been initially copied to it's replicated location. To change the time that the next packet will be built from add or modify the existing **replication_epoc_time** in the **replication_receiver_dir** at the receive location. This time is contained in the a file named:

epoc.###.{destination}.{origination}.epoc

Where:

- ### is the replication_id from the associated **replication definition** **at the originating site**
- **{destination}** is the destination location as defined in the associated **replication definition** **at the originating site**
- **{origination}** is the origination location as defined in the associated **shipper.conf** file **on the originating site**

For example, in order to change the **Replication Epoch Time** in Japan for the **replication_definition** 001 from Atlanta you would create or modify the file **epoc.001.japan.atlanta.epoc** in the **replication_receiver_dir** in Japan.

To set the time to 11:00 am (Atlanta time) on Jan 1, 2005 you would place a single time string in the file like the one below.

01-Jan-2005.11:00:00 UTC +5:00

To set the time to 5:00 pm (Atlanta time) on Jan 1, 2005 you would place a single time string in the file like the one below.

01-Jan-2005.17:00:00 UTC +5:00

Having done so the next packet from the originating site (Atlanta) for this replication definition will contain changes from that time to the current time. set the time to 5:00 pm (Atlanta time) on Jan 1, 2005 you would place a single time string in the file like the one below.

NOTE: use the **UTC offset** that the "**originating site**" uses for itself as defined in its **shipper.conf** file.

Replicating rmelem from the shipper

By default the ClearReplica "shipper" does not replicate destructive commands to remote replicas as that remote replica is often a backup of the local replica and this default behavior prevents the replication of inadvertent destruction in one replica to other replicas. The ClearReplica "shipper" can however be configured to replicate the rmelem command to one or more replicas.

Note: This feature requires the use of ClearTrigger 12.6 or higher on the "shipping" locations VOB.

To inform the ClearReplica shipper that you wish it to package "rmelem" information in the packets it sends to remote locations, just create a fully writable directory named "**_dhist**" in the "**_ship_out**" directory (or that directory's **relocated location**). If ClearTrigger 12.6 or higher encounters this directory it will place encrypted 'rmelem information" there for the rmelem commands that successfully completed within the scope of a replication key.

ClearReplica shipper will consider this information when creating replication packets.

Note: The "shipper" will include the rmelem information in the packets it sends, but it is ultimately up to the "receiver" to determine it if will honor or ignore the included rmelem request.

Refer to the next section entitled "**Accept/Reject rmelem request from remote shipper**" if needed.

Accept/Reject rmelem request from remote shipper

By default the ClearReplica "receiver" does not accept any destructive commands from remote replicas as that local replica is often a backup of the remote replica and this default behavior prevents the replication of inadvertent destruction in one replica to other replicas. The ClearReplica "receiver" can however be configured to accept the replicated cleartool rmelem commands.

Each ClearReplica "receiver" directory can be configured differently by using the **rmelem processing parameter** when defining the **receiver directory** in the **receiver.conf** file. The valid values and what they mean are listed in the table below:

rmelem Processing Parameter (values)	Meaning
allow_rmemlem_none	**the default if not provided – ensures rmelem request from other replicas ignored**
allow_rmemlem_file	**ensures that only rmelem request for "files" from other replicas are accepted/processed**
allow_rmemlem_dir	**ensures that only rmelem request for "directories" from other replicas are accepted/processed**
allow_rmemlem_all	**ensures that all rmelem request for "files" and "directories" from other replicas are accepted/processed**

Replicating rmbranch from the shipper (new in 12.0)

By default the ClearReplica "shipper" does not replicate destructive commands to remote replicas as that remote replica is often a backup of the local replica and this default behavior prevents the replication of inadvertent destruction in one replica to other replicas. The ClearReplica "shipper" can however be configured to replicate the rmbranch command to one or more replicas.

Note: This feature requires the use of ClearTrigger 12.10 or higher on the "shipping" locations VOB.

To inform the ClearReplica shipper that you wish it to package "rmelem" information in the packets it sends to remote locations, just create a fully writable directory named "**_dhist**" in the "**_ship_out**" directory (or that directory's **relocated location**). If ClearTrigger 12.10 or higher encounters this directory it will place encrypted 'rmbranch information" there for the rmbranch commands that successfully completed within the scope of a replication key.

ClearReplica shipper will consider this information when creating replication packets.

Note: The "shipper" will include the rmbranch information in the packets it sends, but it is ultimately up to the "receiver" to determine it if will honor or ignore the included rmelem request.

Refer to the next section entitled **"Accept/Reject rmbranch request from remote shipper"** if needed.

Accept/Reject rmbranch request from remote shipper (new in 12.0)

By default the ClearReplica "receiver" does not accept any destructive commands from remote replicas as that local replica is often a backup of the remote replica and this default behavior prevents the replication of inadvertent destruction in one replica to other replicas. The ClearReplica "receiver" can however be configured to accept the replicated cleartool rmbranch commands.

Each ClearReplica "receiver" directory can be configured differently by using the **rmbranch processing parameter** when defining the **receiver directory** in the **receiver.conf** file. The valid values and what they mean are listed in the table below:

Rmelem Processing Parameter (values)	Meaning
allow_rmbranch_none	**the default if not provided – ensures rmbranch request from other replicas ignored**
allow_rmbranch	**ensures that all rmbranch request for (e.g. cleartool rmbranch foo.c@@\main\b1) from other replicas are accepted/processed. Any local branch instance locks are honored.**

Replicating rmver from the shipper (new in 12.0)

By default the ClearReplica "shipper" does not replicate destructive commands to remote replicas as that remote replica is often a backup of the local replica and this default behavior prevents the replication of inadvertent destruction in one replica to other replicas. The ClearReplica "shipper" can however be configured to replicate the rmver command to one or more replicas.

Note: This feature requires the use of ClearTrigger 12.10 or higher on the "shipping" locations VOB.

To inform the ClearReplica shipper that you wish it to package "rmver" information in the packets it sends to remote locations, just create a fully writable directory named "**_dhist**" in the "**_ship_out**" directory (or that directory's **relocated location**). If ClearTrigger 12.10 or higher encounters this directory it will place encrypted 'rmver information" there for the rmver commands that successfully completed within the scope of a replication key.

ClearReplica shipper will consider this information when creating replication packets.

Note: The "shipper" will include the rmver information in the packets it sends, but it is ultimately up to the "receiver" to determine it if will honor or ignore the included rmver request.

Refer to the next section entitled **"Accept/Reject rmver request from remote shipper"** if needed.

Accept/Reject rmver request from remote shipper (new in 12.0)

By default the ClearReplica "receiver" does not accept any destructive commands from remote replicas as that local replica is often a backup of the remote replica and this default behavior prevents the replication of inadvertent destruction in one replica to other replicas. The ClearReplica "receiver" can however be configured to accept the replicated cleartool rmver commands.

Each ClearReplica "receiver" directory can be configured differently by using the **rmver processing parameter** when defining the **receiver directory** in the **receiver.conf** file. The valid values and what they mean are listed in the table below:

Rmelem Processing Parameter (values)	Meaning
allow_rmver_none	**the default if not provided – ensures rmver request from other replicas ignored**
allow_rmver	**ensures that all rmbranch request for (e.g. cleartool rmver foo.c@@\main\b1\17) from other replicas are accepted/processed. Any local element or branch instance locks are honored.**

Replicating rmtype -lbtype from the shipper

By default the ClearReplica "shipper" does not replicate destructive commands to remote replicas as that remote replica is often a backup of the local replica and this default behavior prevents the replication of inadvertent destruction in one replica to other replicas. The ClearReplica "shipper" can however be configured to replicate the **rmtype -lbtype** command to one or more replicas.

Note: This feature requires the use of ClearTrigger 12.7 or higher on the "shipping" locations VOB.

To inform the ClearReplica shipper that you wish it to package "rmtype -lbtype" information in the packets it sends to remote locations, just create a fully writable directory named "**_dhist**" in the "**_ship_out**" directory (or that directory's **relocated location**). If ClearTrigger 12.7 or higher encounters this directory it will place encrypted 'rmtype -lbtype information" there for the rmtype -lbtype commands that successfully completed within the scope of a replication key.

ClearReplica shipper will consider this information when creating replication packets.

Note: The "shipper" will include the rmtype -lbtype information in the packets it sends, but it is ultimately up to the "receiver" to determine it if will honor or ignore the included rmtype -lbtype request.

Refer to the next section entitled "**Accept/Reject rmtype -lbtype request from remote shipper**" if needed.

Accept/Reject rmtype -lbtype request from remote shipper

By default the ClearReplica "receiver" does not accept any destructive commands from remote replicas as that local replica is often a backup of the remote replica and this default behavior prevents the replication of inadvertent destruction in one replica to other replicas. The ClearReplica "receiver" can however be configured to accept the replicated cleartool rmtype (label type) commands.

Each ClearReplica "receiver" directory can be configured differently by using the **rmlbtype processing parameter** when defining the **receiver directory** in the **receiver.conf** file. The valid values and what they mean are listed in the table below:

Rmlbtype Processing Parameter (values)	Meaning
allow_rmlbtype_none	**the default if not provided – ensures rmtype - lbtype request from other replicas ignored**
allow_rmlbtype	**ensures that all rmtype - lbtype request for (e.g. cleartool rmtype lbtype:FOO) from other replicas are accepted/processed. Any local label type locks are honored.**

Replicating rmtype -brtype from the shipper

By default the ClearReplica "shipper" does not replicate destructive commands to remote replicas as that remote replica is often a backup of the local replica and this default behavior prevents the replication of inadvertent destruction in one replica to other replicas. The ClearReplica "shipper" can however be configured to replicate the **rmtype -brtype** command to one or more replicas.

Note: This feature requires the use of ClearTrigger 12.8 or higher on the "shipping" locations VOB.

To inform the ClearReplica shipper that you wish it to package "rmtype -brtype" information in the packets it sends to remote locations, just create a fully writable directory named "**_dhist**" in the "**_ship_out**" directory (or that directory's **relocated location**). If ClearTrigger 12.8 or higher encounters this directory it will place encrypted 'rmtype -brtype information" there for the rmtype -brtype commands that successfully completed within the scope of a replication key.

ClearReplica shipper will consider this information when creating replication packets.

Note: The "shipper" will include the rmtype -brtype information in the packets it sends, but it is ultimately up to the "receiver" to determine it if will honor or ignore the included rmtype -brtype request.

Refer to the next section entitled "**Accept/Reject rmtype -brtype request from remote shipper "accept_reject_rmbrtype_request**" if needed.

Accept/Reject rmtype -brtype request from remote shipper

By default the ClearReplica "receiver" does not accept any destructive commands from remote replicas as that local replica is often a backup of the remote replica and this default behavior prevents the replication of inadvertent destruction in one replica to other replicas. The ClearReplica "receiver" can however be configured to accept the replicated cleartool rmtype (branch type) commands.

Each ClearReplica "receiver" directory can be configured differently by using the **rmbrtype processing parameter** when defining the **receiver directory** in the **receiver.conf** file. The valid values and what they mean are listed in the table below:

Rmlbtype Processing Parameter (values)	Meaning
allow_rmbrtype_none	**the default if not provided – ensures rmtype - brtype request from other replicas ignored**
allow_rmbrtype	**ensures that all rmtype - brtype request for (e.g. cleartool rmtype brtype:integration) from other replicas are accepted/processed. Any**

	local branch type locks are honored.

Replicating rmtype -attype from the shipper (new in 12.0)

By default the ClearReplica "shipper" does not replicate destructive commands to remote replicas as that remote replica is often a backup of the local replica and this default behavior prevents the replication of inadvertent destruction in one replica to other replicas. The ClearReplica "shipper" can however be configured to replicate the **rmtype -attype** command to one or more replicas.

Note: This feature requires the use of ClearTrigger 12.10 or higher on the "shipping" locations VOB.

To inform the ClearReplica shipper that you wish it to package "rmtype -attype" information in the packets it sends to remote locations, just create a fully writable directory named "**_dhist**" in the "**_ship_out**" directory (or that directory's **relocated location**). If ClearTrigger 12.10 or higher encounters this directory it will place encrypted 'rmtype -attype information" there for the rmtype -attype commands that successfully completed within the scope of a replication key.

ClearReplica shipper will consider this information when creating replication packets.

Note: The "shipper" will include the rmtype -attype information in the packets it sends, but it is ultimately up to the "receiver" to determine it if will honor or ignore the included rmtype -brtype request.

Refer to the next section entitled "**Accept/Reject rmtype -brtype request from remote shipper "accept_reject_rmattype_request**" if needed.

Accept/Reject rmtype -attype request from remote shipper

By default the ClearReplica "receiver" does not accept any destructive commands from remote replicas as that local replica is often a backup of the remote replica and this default behavior prevents the replication of inadvertent destruction in one replica to other replicas. The ClearReplica "receiver" can however be configured to accept the replicated cleartool rmtype (attribute type) commands.

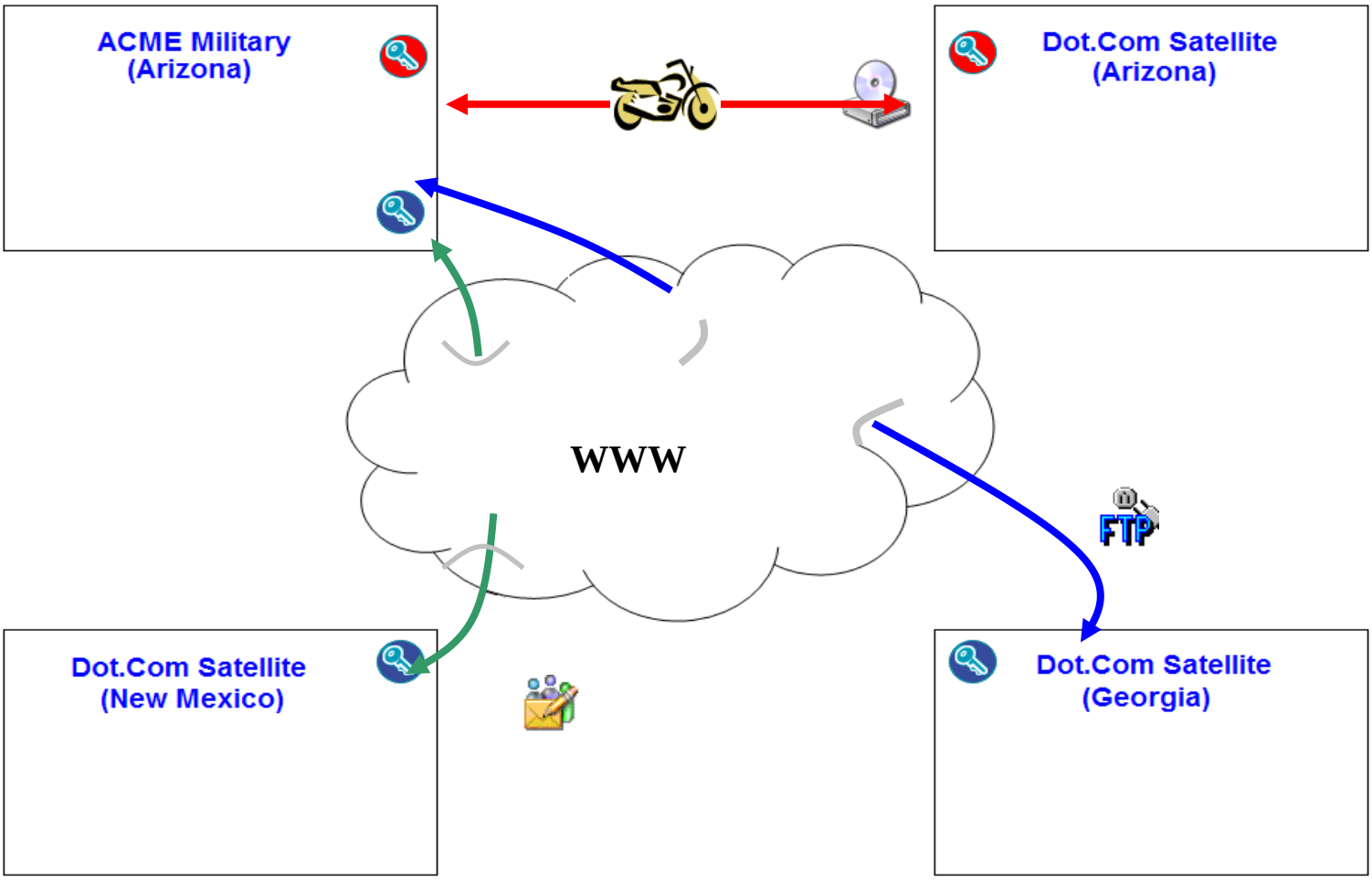
Each ClearReplica "receiver" directory can be configured differently by using the **rmattype processing parameter** when defining the **receiver directory** in the **receiver.conf** file. The valid values and what they mean are listed in the table below:

Rmlbtype Processing Parameter (values)	Meaning
allow_rmattype_none	**the default if not provided – ensures rmtype - attype request from other replicas ignored**
allow_rmattype	**ensures that all rmtype - attype request for (e.g. cleartool rmtype attype:integration) from other replicas are accepted/processed. Any local attribute type locks are honored.**

Using Peer_to_Peer Encryption

All data is by default encrypted using an internal ClearReplica Data Encryption key so that only ClearReplica can decrypt ClearReplica data packets (perfect for closed or otherwise private networks). Additional encryption can be enabled that requires the sender and receiver to agree on a generated encryption key known only to the Sender and Receiver, thus allowing for the secure transportation of packets across open, public or 3rd party networks or even thought the use of 3rd party courier services. Even those with ClearReplica cannot decrypt the packet without the Peer-to-Peer encryption key.

For example two companies ACME Military and Dot Com Satellite are co-developing a Militant Satellite Surveillance system called BigEye. Both organizations have offices in Arizona, just ½ mile apart and elected to use a bike courier service to deliver each day's ClearReplica packet on DVD. Additionally, Dot.Com Satellite has offices in New Mexico and Georgia that will help develop BigEye; the offices in Georgia allows FTP thought their firewall, but the office in New Mexico only allows email through their firewall. Thought they plan to implement a private network in the future at this moment they do not own any of the infrastructure between any of the locations… as the public internet is used between ACME Military in Arizona and the Dot.Com Satellite offices in New Mexico and Georgia, while the 3rd party bike courier is used between the companies in Arizona.



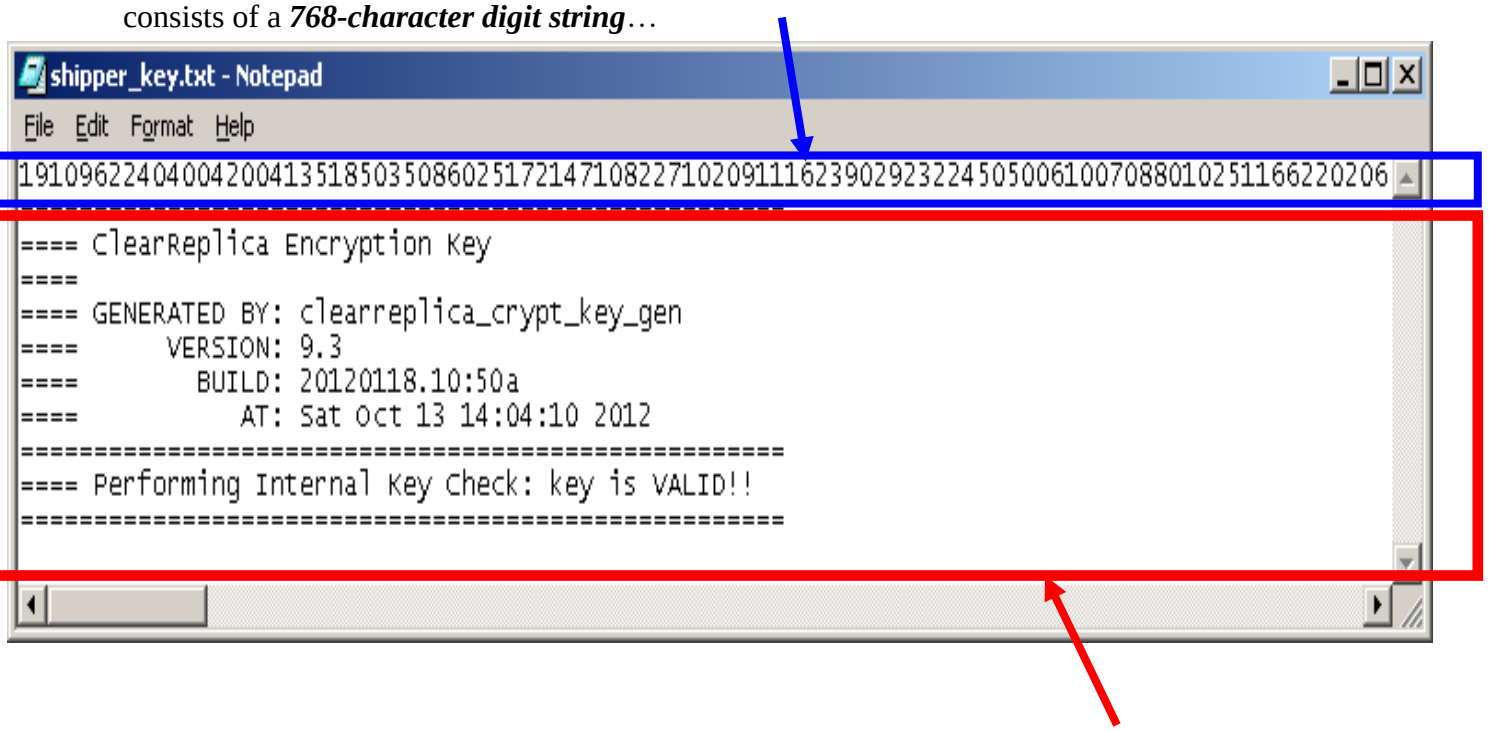
Now, data can be safely transported through the public network without having to be concerned about the prying eyes of those that might sniff the packets between the destinations; they can't decrypt the packet without the both ClearReplica ***and*** the **peer_to_peer encryption key** that is only known by the sender and the receiver (which is ***NOT*** included in the packet).

The same can be said for the packets that are sent between ACME Military and Dot.Com Satellite offices in Arizona where a 3rd party Courier is used to transport the DVD between the sites. Since both sites agree to use the peer_to_peer encryption key they are the only ones that can view or process that ClearReplica data.

ClearReplica **"peer_to_peer" encryptions keys** can be created at either location by using the **clearreplica_crypt_key_gen** command. Once generated at one location (site A) the ClearReplica administrator at can send the key to the ClearReplica administrator at another location (site B) where that administrator can install the keys at that location so packets sent from Site A can be read at Site B.

If a ClearReplica "**receiver**" process receives a packet encrypted from a ClearReplica "**shipper**" process not using the exact same encryption key, a message is written to the "receiver" error logs and that packet is ***destroyed***.

The "Peer_To_Peer" encryption key generated by the **clearreplica_crypt_key_gen** program consists of a ***768-character digit string***…



followed by and Carriage Return and then miscellaneous key generation information.

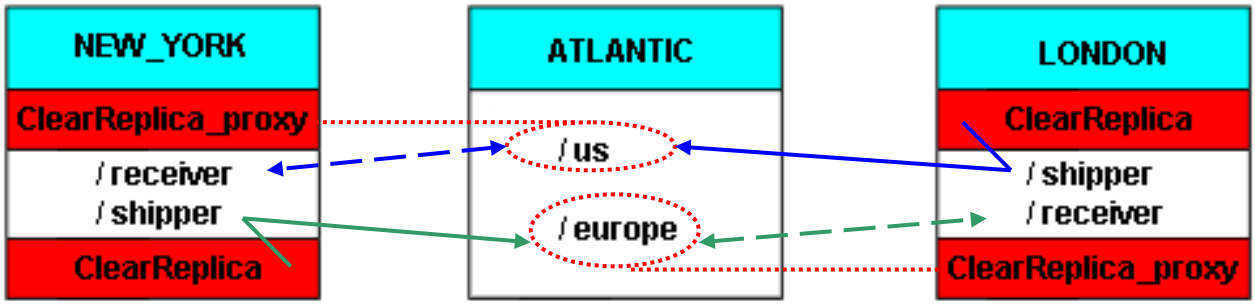
The whole first line of the file is the actual peer_to_peer encryption key,
while all other lines are informational in nature and not used by ClearReplica in any way. Feel free to place (or remove) any information that helps you track or maintain the key after the first line.

Creating ClearReplica Proxy Machines

The **clearreplica_proxy** program facilitates the transfer of ClearReplica data between two ClearReplica servers that do not have direct access to each other, but share access to one or more machines between them that can be used as a ClearReplica **'proxy'** machines. The machines used in the transfer that are not actually the source or destination ClearReplica Server constitute the 'proxy chain' of machines. The **clearreplica_proxy** program can also facilitate the transfer of data between machines that do not have a static IP address (i.e. salesman laptops) so that either "replication" or "deployment" data can be sent to those machines anytime they are connected to the web.

The program will 'push' all current ClearReplica 'epoc' data from a directory on the local machine and place that 'epoc' data on a specific directory on a remote machine so that data can be processed by that machine or a passed along to a machine that is even farther in the 'proxy chain'.

The program will then 'pull' any complete ClearReplica package files existing in that directory on the remote machine to a directory on the local machine. This local directory should either be a local ClearReplica 'receiver' directory or a directory looked at by another ClearReplica 'proxy' machine farther in the 'proxy chain'.



In the above example: New York "ships" to London through the **/europe** directory on the **ATLANTIC** machine while London "ships" to New York through the **/us** directory on the **ATLANTIC** machine. For "shipping" purposes, each destination machine treats the 'proxy" machine as if it was the actual destination machine (the shipper is unaware of any proxy machine). The **clearreplica_proxy** program running on each ***destination*** machine syncs the appropriate directory on the 'proxy machine' with the actual receiver directory on the ClearReplica Server.

NOTE: The proxy machine in this case does not require any ClearReplica software or licenses. The proxy machine itself is not aware it is a proxy machine.

NOTE: You can have any number of proxy machines between destinations, if a proxy machine also pulls from another proxy machine that that proxy machine must have **clearreplica_proxy** program run from it. **No license is required to run or use clearreplica_proxy**.

Starting clearreplica_proxy:

The clearreplica_proxy program can be started manually or within a startup script or cron job (for UNIX) or startup script or scheduled 'AT' job (for Windows).

Each invocation of the script will process all valid and existing data available at that time before completing. The program will always write its result stdout and stderr so you should redirect its output to you file of choice in scheduled jobs or task.

USAGE: clearreplica_proxy -ver | -help | -man | proxy_file

Such that:

-ver - produces clearreplica_proxy product version message

-help - produces clearreplica_proxy USAGE message

-man - produces this clearreplica_proxy man page

proxy_file - Will process ClearReplica receiver data to/from another machine according to the proxy_file as described in the section below entitled '**ClearReplica proxy_file**' of the man page.

Clearreplica proxy_file:

The **ClearReplica proxy_file** holds the communication information used to communicate between the ClearReplica Proxy machines. The proxy_file can be located anywhere and is referenced by the **clearreplica_proxy** program.

The proxy_file should contain at least three (3) lines at the BEGINNING of the file.

[Line #1] must contain a standard ftp program or the path to one if it is not expected to be in the path of the running process. This FTP line should not contain any space in the path.

[Line #2] must contain the directory that is the expected 'receiver' directory for ClearReplica for this location or another 'proxy' directory to be used by another ClearReplica 'proxy' machine.

[Line #3] must contain the 'FTP communication key' that was generated by the **ClearReplica_hostentry** program for the location that this process will 'pull' ClearReplica receiver data from and place in the directory contained in Line #2.

The 'FTP communication key' contains the encrypted machine and location information needed to find and pull the ClearReplica receiver data from the remote machine. That machine **DOES NOT** need to have ClearReplica software or licenses

NOTE: All three (3) lines must exist for this to be a valid proxy_file. Only the first three (3) lines are read for data, all other lines are ignored and treated as comment lines.

Examples of valid proxy_file data are in the table below:

Example Type		Example proxy_file data:
Windows	**(ftp in process path)**	**ftp** **c:\clearreplica\receiver** **[yv-INL~XdeR6Hf$g:2d0TX$Xhiw$Ir)8(p]**
	(ftp *NOT* in process path)	**C:\WINNT\system32\ftp** **c:\clearreplica\receiver** **[yv-INL~XdeR6Hf$g:2d0TX$Xhiw$Ir)8(p]**
UNIX	**(ftp in process path)**	**ftp** **/clearreplica/receiver** **[yv-INL~XdeR6Hf$g:2d0TX$Xhiw$Ir)8(p]**
	(ftp *NOT* in process path)	**/usr/bin/ftp** **/clearreplica/receiver** **[yv-INL~XdeR6Hf$g:2d0TX$Xhiw$Ir)8(p]**

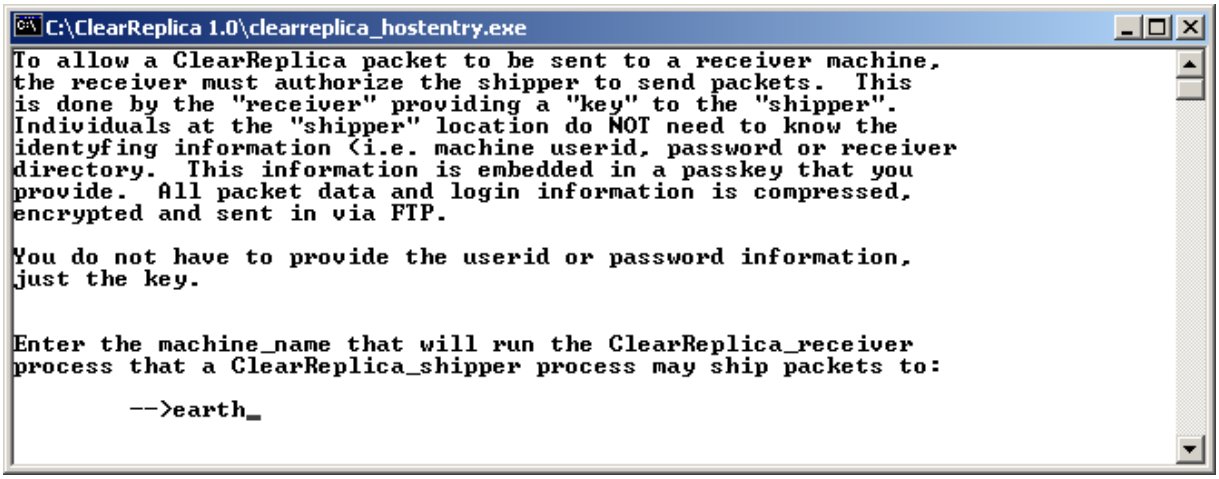
Clearreplica_hostentry

The **clearreplica_hostentry** program takes no parameters and can be executed by anyone responsible for making a replication_key. They key is provided so that the administrator of the "receiver" site does not have to provide the "shipper" and actual userid, password, machine name or name of any "receiving" directory.

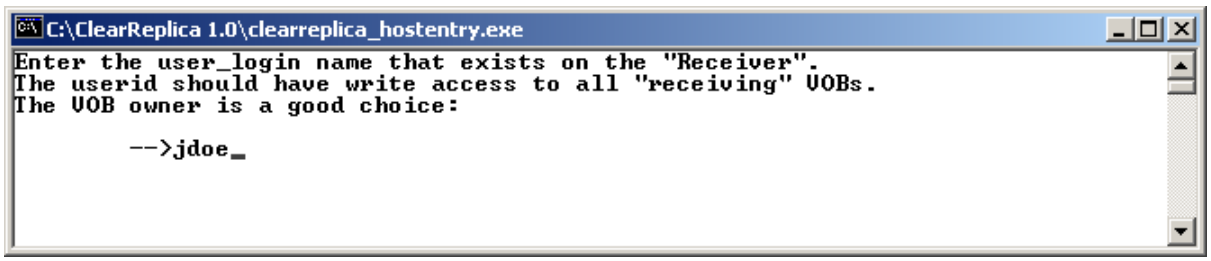
The program ask for

- The machine name of the "receiver"
- The login id that will be used by the transport program
- The password for the provided
- The directory that the "receiver' allows the "shipper" to send to replication packets.

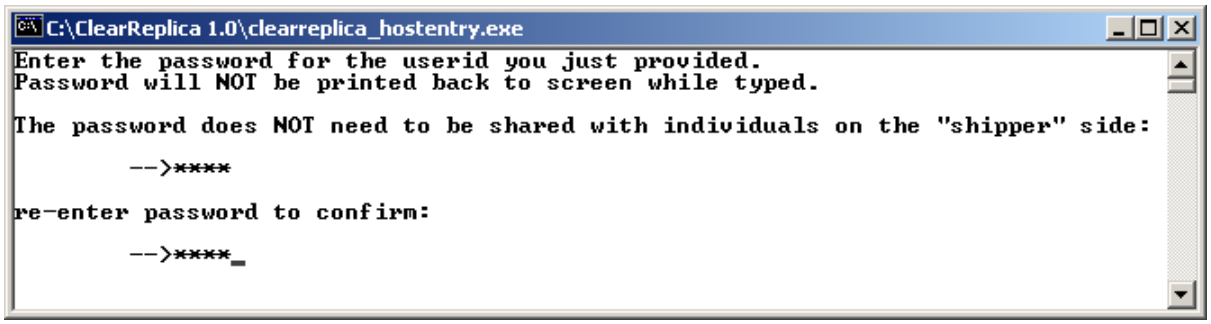
The **clearreplica_hostentry** program generates a **replication_key** with the provided information. Executing the clearreplica_hostentry program from the Windows OS might look like the example below:



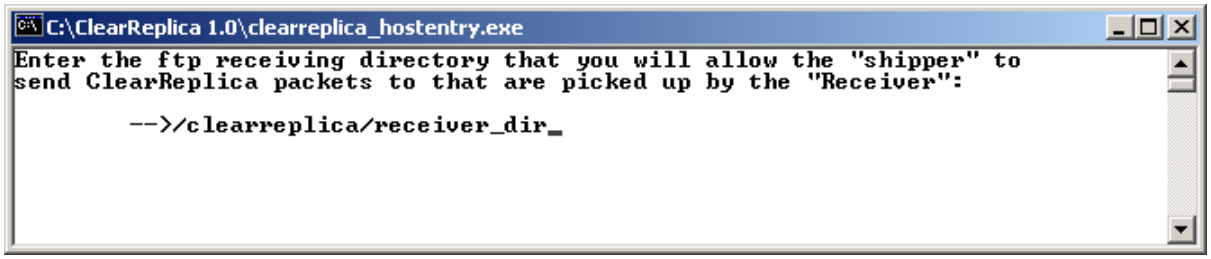
Enter the name (or IP address) of the "receiver" machine that will run the **clearreplica_receiver** program then enter a carriage return to accept the input.



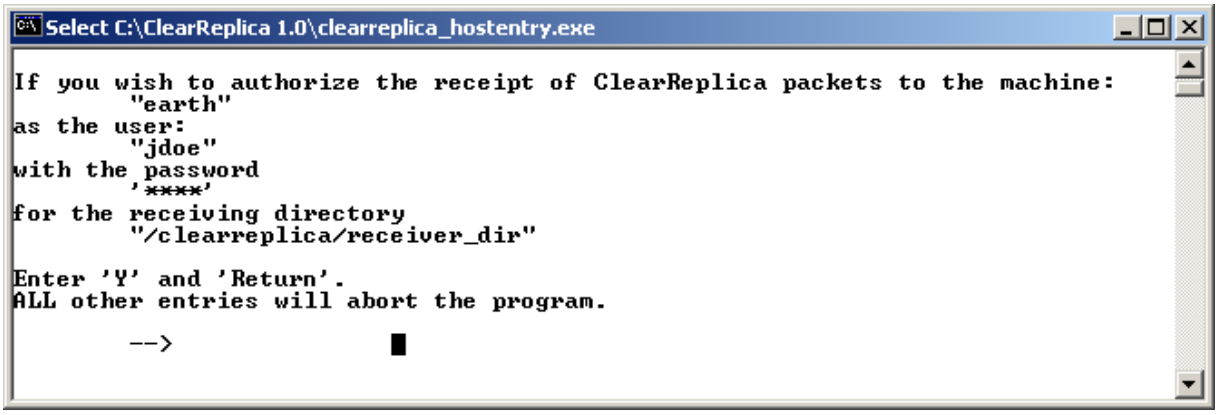
Enter the userid that should be used to ftp the encrypted packets to the "receiver" machine then enter a carriage return to accept the input.



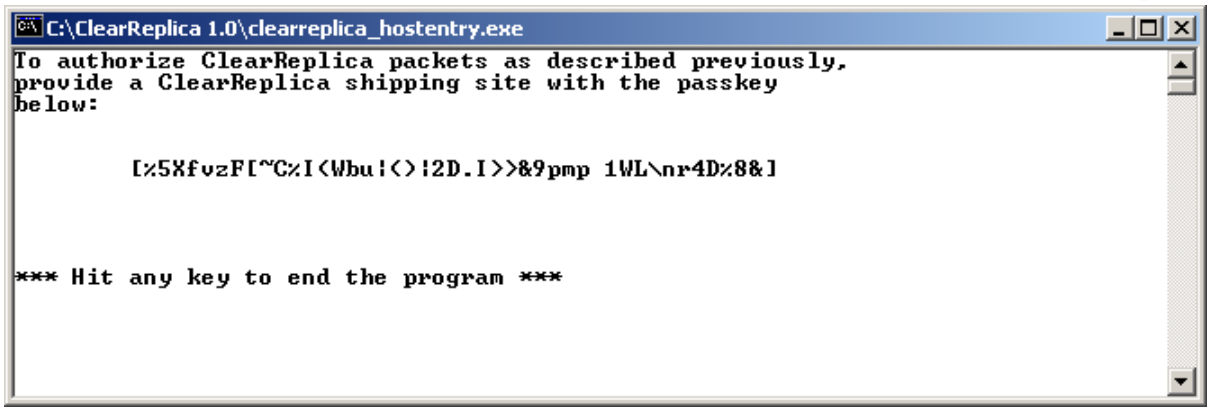
Enter the password for the given userid. For Windows it is not echoed to the terminal and must be entered twice. For UNIX it is echoed and must only be entered once. Once entered then enter a carriage return to accept the input.



Enter the **replication_receiver_directory** where the shipper will be allowed to send ClearReplica replication packets. The directory can be an absolute path from the ftp root or relative to the ftp root. The provided directory should exist and be writable by the input userid. Once entered then enter a carriage return to accept the input. "Replicate" VOBs to different locations (combining VOBs):



If the information displayed is correct then enter 'Y' to display the generated **replication_key**.



Provide the key to the ClearReplica administrator for the "shipper" site to use to send ClearReplica packets to the receiver. The key is case sensitive and MUST be entered exactly as printed including the surrounding square brackets.

Clearreplica_shipout

The **clearreplica_shipout** program will forcibly tell ClearReplica shipper process to "inspect" the definition and build a packet with changes since the last known good packet was created and shipped. ClearTrigger is normally responsible for informing the ClearReplica shipper that there are possible changes to replicate to a remote site. ClearTrigger performs this action so that ClearReplica does not waste CPU inspecting the VOB for changes and does not waste company bandwidth sending empty packets.

This is useful when the changes themselves as "Site A" were not created at that location but were changes received from a ClearCase Multisited replica from another replica.

USAGE: clearreplica_shipout -ver | -help | [clearbit_file replication_id]

Such that:

-ver - produces clearreplica_shipout **product version message**

-help - produces clearreplica_shipout **USAGE message**

-clearbits_file - is the associated clearbits file that holds the ClearReplica License

-replication_id - is the associated replication id to mark as "check for replication"

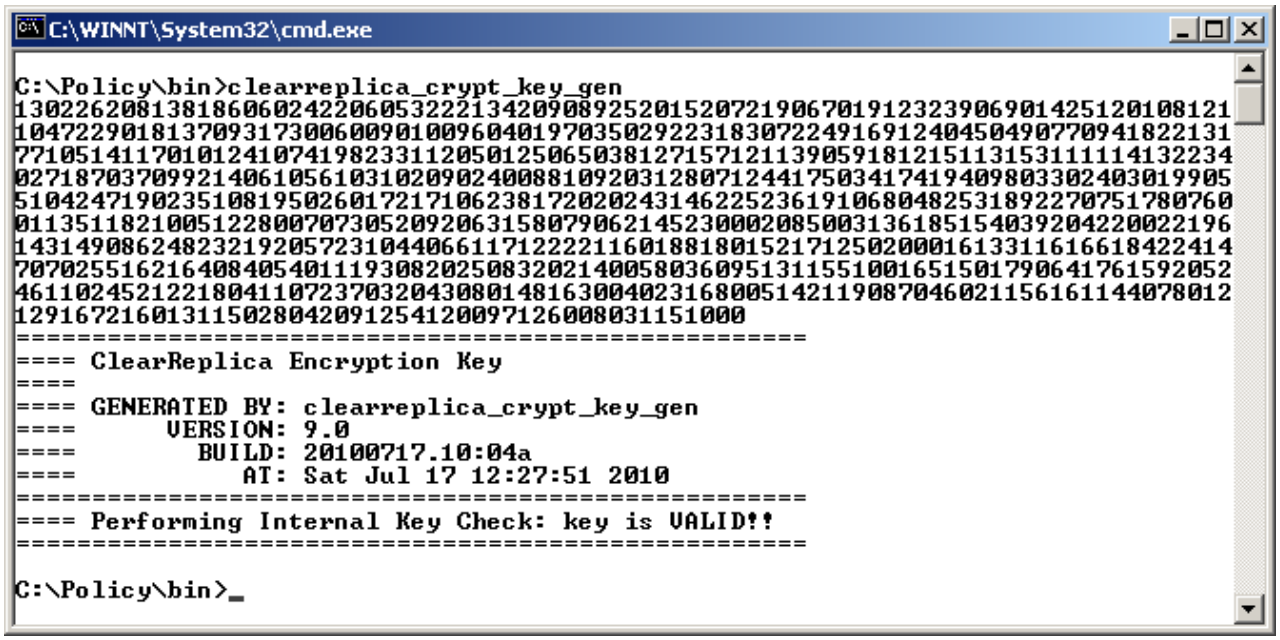
The program will check all appropriate **shipper.conf** files in the associated depot (including any defined shipper **class** files) until a match for the **replication_id** is found. Once a **replication_id** match is found then that replication_id is marked "check for replication" so ClearReplica shipping process will create its associated packet if needed.

Clearreplica_crypt_key_gen

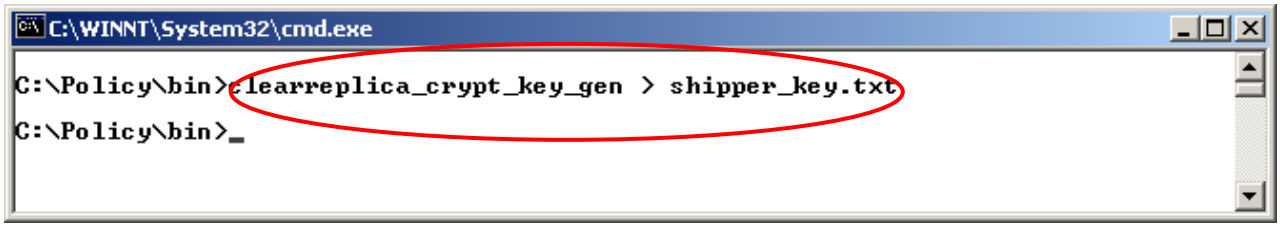
The **clearreplica_crypt_key_gen** program takes no parameters and can be executed by anyone responsible for making **"peer_to_peer" encryption keys**. This command can be run from any machine on the shipper or receiver, but the generated key must be used at both to enable data encrypted at the shipper to be read at the receiver. It is expected that a ClearReplica administrator form one site will generate the key and share that key with the ClearReplica Administrator of the other site

If a ClearReplica "**receiver**" process receives a packet encrypted from a ClearReplica "**shipper**" process not using the exact same encryption key, a message is written to the " receiver" error logs and that packet is destroyed.

The **clearreplica_crypt_key_gen** program generates a **"peer_to_peer" encryption key** to stdout. Executing the clearreplica_crypt_key_gen program from the Windows OS might look like the example below:



The program output consists of a 768-character digit string followed by and Carriage return and then miscellaneous key generation information. To avoid having to type or cut/paste the key it is suggested that one redirect the output to a file.

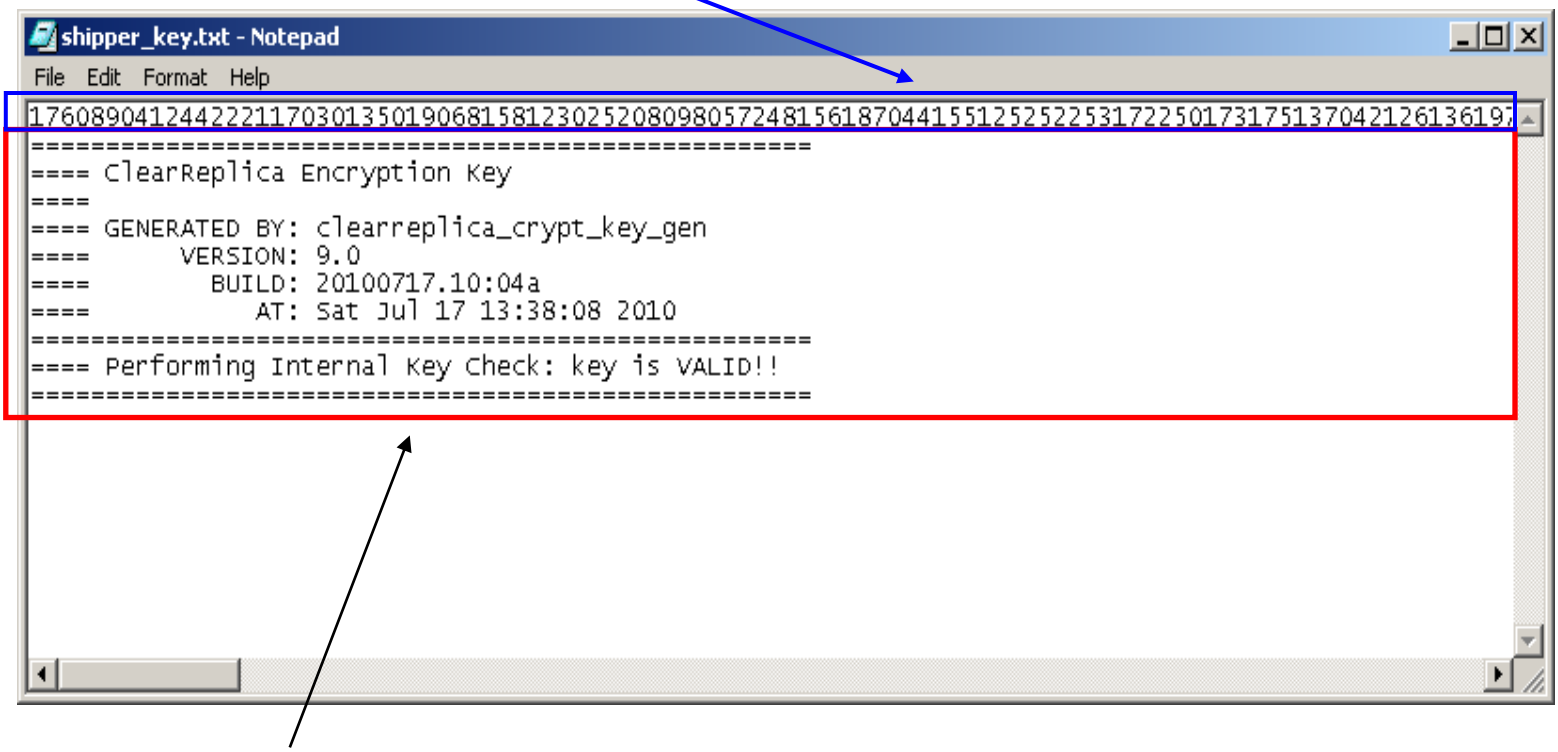


then that file can be sent to any appropriate "receiver" or "shipper" sites you with to share the encryption key with.

If you want a ***ClearReplica Shipper*** process to use this key you would set the encryption value to "**peer_to_peer**" in the appropriate **shipper.conf** file and then create and place the encryption key contents into that shippers appropriate encryption key file in the ***…/depot/m_bay/config/keys*** directory. For example, the shipper process that is associated with the **…/depot/m_bay/config/shipper.conf** file would use the peer_to_peer encryption key stored in the **…/depot/m_bay/config/keys/shipper_key.txt** file, while the shipper process that is associated with the **…/depot/m_bay/config/shipper_1.conf** file would use the peer_to_peer encryption key stored in the **…/depot/m_bay/config/keys/shipper_key_1.txt** file.

If you want a ***ClearReplica Receiver*** process to use this key you would set the encryption value to "**peer_to_peer**" in the appropriate **receiver.conf** file and then create and place the encryption key contents into that shippers appropriate encryption key file in the ***…/depot/m_bay/config/keys*** directory. For example, the shipper process that is associated with the **…/depot/m_bay/config/receiver.conf** file would use the peer_to_peer encryption key stored in the **…/depot/m_bay/config/keys/receiver_key.txt** file, while the shipper process that is associated with the **…/depot/m_bay/config/receiver_atl.conf** file would use the peer_to_peer encryption key stored in the **…/depot/m_bay/config/keys/receiver_key_atl.txt** file.

Remember, **the whole first line of the file is the actual peer_to_peer encryption key**,



while all other lines are informational in nature and not used by ClearReplica in any way. Feel free to place (or remove) any information that helps you track or maintain the key after the first line.

Clearreplica_shiptest

The **clearreplica_shiptest** program will send a test packet to test communication between ClearReplica remote locations.

It is most often used for:

- determine if the hostmap **location_name** (**location_host_entry**) is properly defined. The sender creates and sends a test packet to the destination location to test validity of the **shipper.conf** file and the location_name's associated **location.hostmap**

- determine the true bandwidth expected between ClearReplica Locations. The command returns the actual bandwidth experience when sending the test packet of the requested size.

This is most useful when determining the validity of your initial ClearReplica setup and ClearReplica communication between remote sites.

USAGE: clearreplica_shiptest -ver | -help | [cleartrigger_depot remote_location [class=0..9] [meg=1..100])]

Such that:

-ver - produces clearreplica_shipout **product version message**

-help - produces clearreplica_shipout **USAGE message**

cleartrigger_depot - The **full path** to the associated ClearTrigger Depot. This can be a network (UNIX) or UNC (Windows) path, but the local path is preferred if the process is to run on the same machine that holds the ClearTrigger Depot.

remote_location - A required and valid **location_name** (**location_host_entry**) that is associated with **hostmap** file of the name **<location>.hostmap** where <location> is the name given to the "receiver" location by the "shipper" location (i.e. "atlanta", or "london"). There is a 1-to-1 correspondence between the **location_name** (e.g. "atlanta") provided and a file named "**atlanta.hostmap"** in the **{depot}/m_bay/config/hostmaps** directory.

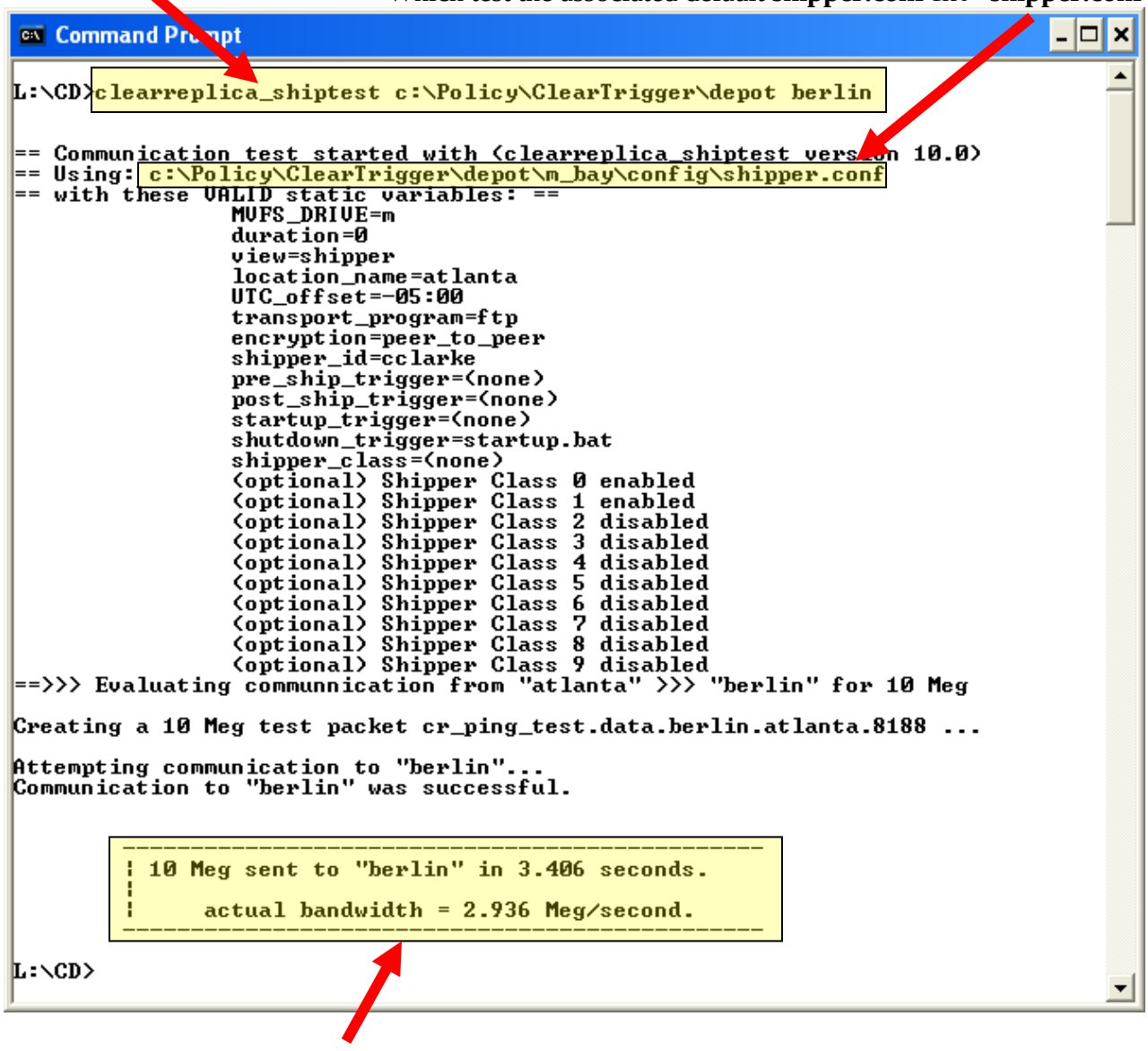
[class=0..9] - Optional: use a shipper.conf file other the default "shipper.conf" file in "m_bay/config" directory in the associated depot. When used the shipper**_[class].**conf file is used. If class is "2" then "shipper**_2**.conf" will be used. Values 0-9 are valid. When omitted the default "classless" shipper.conf file is used.

[meg=1..100] - Optional: The size (in Meg) of the test packet to send to the remote location. If omitted the default of meg=1 is used

The program will check the appropriate **shipper.conf** files (as per the exclusion or use of the "**class=**" values passed in) in the associated depot and test the validly of the runtime variables within the **shipper.conf** file. If the associated **shipper.conf** file is valid it users those parameters and the encrypted values within the associated **hostmap file** (**location_host_entry**) to send a test packet to the remote location. It then displays any errors encountered if unsuccessful or the actual bandwidth experienced if successful.

For example to test a ClearReplica environment between Atlanta and Berlin one might type the command below:

Which test the associated default **shipper.conf** file "**shipper.conf**"



Which results in a **10Meg** (the default) test packet being sent to "Berlin" which was sent in 3.406 seconds for an actual bandwidth of **2.936 Meg/Second**. At this point you know that you have established proper communication between "Atlanta" and "Berlin" and what the expected bandwidth might be between those locations.

ClearReplica_shipper

The **clearreplica_shipper** program creates and ships the ClearReplica packets based on the **shipper.conf** file configuration settings. Only one clearreplica_shipper process can run per ClearTrigger depot. The shipper.conf file is located in the **m_bay/config** directory in that depot. The **clearreplica_shipper** program can be started manually or within a startup script or cron job (for UNIX) or startup script or scheduled 'AT' job (for Windows). This process can only be started by the userid designated by the **shipper_id** configuration variable defined in the **shipper.conf** file.

USAGE: clearreplica_shipper -ver | -help | path_to_ClearTrigger_depot (-start | -once | -stop | -restart | -reonce | -queue_stop | -dequeue_stop) [class] [-wait]

Such that:

-ver - produces clearreplica_shipper **product version message**

-help - produces clearreplica_shipper **USAGE message**

path_to_ClearTrigger_depot - The **full local path** to the associated ClearTrigger Depot. ClearReplica receiver normally runs on the same machine that holds the ClearTrigger Depot.

-start - **starts the process** and continually runs for **infinite passes** as per the configuration parameters in the shipper.conf file.

-once - **starts the process** and runs for **one (1) pass** as per the configuration parameters in the shipper.conf file.

-stop - **Stops any ClearReplica_shipper** process for the associated depot

-restart - Performs a ClearReplica_shipper **–stop** then a **–start** for the associated depot.

-reonce - Performs a ClearReplica_shipper **–stop** then a **–once** for the associated depot.

-queue_stop - Attempt the stop of a ClearReplica_shipper (**–stop**) if the process cannot stop because it is working on a packet the stop is queued to be invoked when the process has completed its current packet.

-dequeue_stop - Remove a "queued" stop for a ClearReplica_shipper process.

[class] - use a shipper.conf file other the default "shipper.conf" file in "m_bay/config" directory in the associated depot. When used the shipper**_[class].**conf file is used. If class is "2" then "shipper**_2**.conf" will be used. Values 0-9 are valid.

-wait - Start the shipper process in the foreground, the calling process will wait for the shipper process to complete. The default is for the shipper to run in the background and immediately return control to the calling process. Useful for the monitoring of shipper runs as a scheduled task.

Starting/stopping the ClearReplica "Shipper"

The **clearreplica_shipper** executable is located at both **{depot}/bin/{arch}** directory and the **{depot}/bin/_arch/{arch}** directory (where {depot} is the path to the associated ClearTrigger depot and {arch} is the appropriate architecture for the machine that runs the shipper).

The allowed syntax of the command is as follows:

clearreplica_shipper -ver | -help | path_to_ClearTrigger_depot (-start | -once | -stop | -restart | -reonce | -queue_stop | -dequeue_stop) [class] [-wait]

where **path_to_depot** is the path to the associated ClearTrigger depot. **Do NOT us a UNC path for the path_to_depot argument for the clearreplica_shipper.**

Use the **–start** option to start the clearreplica_shipper for a given depot and continue to run passes as per the associated **shipper.conf** file.

Use the **–once** option to start the clearreplica_shipper for a given depot and make only one pass through the currently associated **shipper.conf** file.

Use the **–stop** option to stop the clearreplica_shipper for a given depot.

Use the **–restart** option to stop and then immediately start the clearreplica_shipper and continue to run passes as per the associated **shipper.conf** file.

Use the **–reonce** option to stop and then immediately start the clearreplica_receiver for a given depot and make only one pass through the currently associated **shipper.conf** file.

Use the **–queue_stop** option schedule a stop of a shipper that is currently processing a packet.

Use the **–dequeue_stop** option to remove a 'queued" stop for a shipper for a given depot.

Use the **optional [class]** option to use a shipper configuration other than the one defined in the default "shipper.conf" file in "m_bay/config" directory in the associated depot. When "class" is used the shipper**_[class].**conf file is use such that if which is "2" then "shipper**_2**.conf" will be used. Values 0-9 are valid.

When started the clearreplica_shipper places itself in the background (unless the –wait switch is used). The clearreplica_shipper MUST run as the appropriate **shipper_id** as defined in the associated **shipper.conf** file.

Use the **–wait** option to start the shipper process in the foreground causing the calling process or program to wait for the shipper to finish.

For UNIX servers, the clearreplica_shipper should be started from the server startup scripts or equivalent.

For Windows servers, the clearreplica_shipper should be started as a scheduled startup task that runs when the system boots.

Starting multiple clearreplica_shipper processes

To improve performance or to distribute administration between multiple group members of VOB owner groups it is possible to start multiple ClearReplica_shipper processes by taking advantage of the [class] parameter passed to ClearReplica_shipper. These processes can even be started from different machines to further improve performance. When using multiple receiver processes and therefore multiple shipper.conf files there are some additional rules to consider:

- Each **shipper.conf** file must have its own dedicated view that is writable by the associated user in its **shipper_id** variable. Alternately the process could run at mutually exclusive times.

  Note: ClearReplica_shipper will prevent a second shipper from running with a view used by a currently running shipper for the associated depot.

- Multiple ClearReplica shipper processes running for a single ClearReplica depot can be used to further improve performance. These processes can even reside on different machines with different architectures to better take advantage of available CPU and bandwidth.

ClearReplica_receiver

The **clearreplica_receiver** program processes received ClearReplica packets based on the **receiver.conf** file configuration settings. Any number of clearreplica_receiver processes can run per ClearTrigger depot. Any receiver.conf files must be located in the **m_bay/config** directory in that depot. The **clearreplica_receiver** program can be started manually or within a startup script or cron job (for UNIX) or startup script or scheduled 'AT' job (for Windows). This process can only be started by the userid designated by the **receiver_id** configuration variable defined in the **receiver.conf** file.

USAGE: clearreplica_receiver -ver | -help | path_to_ClearTrigger_depot (-start | -once | -stop | -restart | -reonce | -queue_stop | -dequeue_stop) [which] [-wait]

Such that:

-ver - produces clearreplica_receiver **product version message**

-help - produces clearreplica_ receiver **USAGE message**

path_to_ClearTrigger_depot - The **full local path** to the associated ClearTrigger Depot. ClearReplica receiver normally runs on the same machine that holds the ClearTrigger Depot.

-start - **starts the process** and continually runs for **infinite passes** as per the configuration parameters in the shipper.conf file.

-once - **starts the process** and runs for **one (1) pass** as per the configuration parameters in the shipper.conf file.

-stop - **Stops any ClearReplica_receiver** process for the associated depot

-restart - Performs a ClearReplica_receiver **–stop** then a **–start** for the associated depot.

-reonce - Performs a ClearReplica_receiver **–stop** then a **–once** for the associated depot.

-queue_stop - Attempt the stop of a ClearReplica_shipper (**–stop**) if the process cannot stop because it is working on a packet the stop is queued to be invoked when the process has completed its current packet.

-dequeue_stop - Remove a "queued" stop for a ClearReplica_shipper process.

[which] - use a receiver.conf file other the default "receiver.conf" file in "m_bay/config" directory in the associated depot. When used the receiver**_[which].**conf file is use. If which is "2" then "receiver**_2**.conf" will be used while if which is "from_atl" then "receiver**_from_atl**.conf" will be used.

-wait - Start the receiver process in the foreground, the calling process will wait for the receiver process to complete. The default is for the receiver to run in the background and immediately return control to the calling process. Useful for the monitoring of receiver runs as a scheduled task.

Starting/Stopping the ClearReplica "Receiver"

The **clearreplica_receiver** executable is located at both **{depot}/bin/{arch}** directory and the **{depot}/bin/_arch/{arch}** directory (where {depot} is the path to the associated ClearTrigger depot and {arch} is the appropriate architecture for the machine that runs the receiver).

The allowed syntax of the command is as follows:

clearreplica_receiver -ver | -help | [path_to_ClearTrigger_depot (-start | -once | -stop | -restart | -reonce | -queue_stop | -dequeue_stop) [which]]

where **path_to_depot** is the path to the associated ClearTrigger depot. **Do NOT us a UNC path for the path_to_depot argument for the cleareplica_receiver.**

Use the **–start** option to start the clearreplica_receiver for a given depot and continue to run passes as per the associated **receiver.conf** file.

Use the **–once** option to start the clearreplica_receiver for a given depot and make only one pass through the currently associated **receiver.conf** file.

Use the **–stop** option to stop the clearreplica_receiver for a given depot.

Use the **–restart** option to stop and then immediately start the clearreplica_shipper and continue to run passes as per the associated **receiver.conf** file.

Use the **–reonce** option to stop and then immediately start the clearreplica_receiver for a given depot and make only one pass through the currently associated **receiver.conf** file.

Use the **–queue_stop** option schedule a stop of a receiver that is currently processing a packet.

Use the **–dequeue_stop** option to remove a 'queued" stop for a receiver for a given depot.

Use the **optional [which]** option to use a receiver configuration other the one defined in the default "receiver.conf" file in "m_bay/config" directory in the associated depot. When "which" is used the receiver**_[which].**conf file is use such that if which is "2" then "receiver**_2**.conf" will be used while if which is "from_atl" then "receiver**_from_atl**.conf" will be used.

When started the clearreplica_receiver places itself in the background (unless the –wait switch is used). The clearreplica_receiver MUST run as the appropriate **receiver_id** as defined in the associated **receiver.conf** file.

Use the **–wait** option to start the shipper process in the foreground causing the calling process or program to wait for the shipper to finish.

For UNIX servers, the clearreplica_receiver should be started from the server startup scripts or equivalent.

For Windows servers, the clearreplica_receiver should be started as a scheduled startup task that runs when the system boots.

Starting multiple clearreplica_receiver processes

To improve performance or to distribute administration between multiple group members of VOB owner groups it is possible to start multiple ClearReplica_receiver processes by taking advantage of the [which] parameter passed to ClearReplica_receiver. These processes can even be started from different machines to further improve performance. When using multiple receiver processes and therefore multiple receiver.conf files there are some additional rules to consider:

- Each **receiver.conf** file must have its own dedicated view that is writable by the associated user in its **receiver_id** variable. Alternately the process could run at mutually exclusive times.

  Note: ClearReplica_receiver will prevent a second receiver from running with a view used by a currently running receiver for the associated depot.

- A single VOB ***portion*** should not be processed by multiple receivers. For example the VOB /vobs/a_vob can receive packets from two different receivers so long as those packets contain mutually exclusive data (i.e. /vobs/a_vob/src changes can be received by one receiver process while /vobs/a_vob/doc changes can be received by another).

- A single **receiver.conf** file can refer to ***multiple* receiver directories**, but each receiver directory ***should only be referred to by one receiver.conf*** file.

- ClearReplica receiver processes running for a single ClearReplica depot can be spread over multiple receivers running from different machines to further improve performance.

ClearReplica Log Files

Since ClearReplica processing is fully automated and unattended it produces an extensive set of log files for both the "shipper" and "receiver" processes. All ClearReplica log files are located in the **m_bay/logs** directory for the associated ClearTrigger Depot.

"Shipper" Logs

Since ClearReplica "shipper" processing is fully automated and unattended it produces an extensive set of "shipping" log files for its processes. All ClearReplica "shipper" log files are located in the **m_bay/logs** directory and start with "**shipper_"** in their name. There are six (6) shipper log files:

Shipper Log file	Containing entries in chronological order
shipper_all.txt	All ClearReplica shipper log messages (except deployment details and TOC details)
shipper_deploys.txt	All ClearReplica shipper ***deployment details*** (version list of files contained in "deployment "Packets.
shipper_errors.txt	All ClearReplica shipper ***error*** messages
shipper_start_stop.txt	All ClearReplica shipper ***start*** or ***stop*** messages
shipper_TOCs.txt	All ClearReplica shipper ***TOC details*** (version list of files contained in "replication "Packets.
shipper_warnings.txt	All ClearReplica shipper ***warning*** messages

Since multiple ClearReplica "shipper" processes can be started (one for each **shipper.conf** class file), each process has its own logging space (set of logging files). One "shipper" process might run against "**shipper.conf**" while another runs against the shipper class "**shipper_0.conf**" file and still a third might run against "**shipper_1.conf**". These processes would produce log files as illustrated below:

Running the clearreplica_receiver against these files produces…		
shipper.conf	**shipper_0.conf**	**shipper_1.conf**
shipper_all.txt	shipper_all_**0**.txt	shipper_all_**1**.txt
shipper_deploys.txt	shipper_deploys_**0**.txt	shipper_deploys_**1**.txt
shipper_errors.txt	shipper_errors_**0**.txt	shipper_errors_**1**.txt
shipper_start_stop.txt	shipper_start_stop_**0**.txt	shipper_start_stop_**1**.txt
shipper_TOCs.txt	shipper_TOCs_**0**.txt	shipper_TOCs_**1**.txt
shipper_warnings.txt	shipper_warnings_**0**.txt	shipper_warnings_**1**.txt

"Receiver" Logs

Since ClearReplica "receiver" processing is fully automated and unattended it produces an extensive set of "receiving" log files for its processes. All ClearReplica "receiver" log files are located in the **m_bay/logs** directory and start with "**receiver_"** in their name. By default there are six (6) receiver log files:

Receiver Log file	Containing entries in chronological order
receiver_all.txt	All ClearReplica receiver log messages (except deployment details and TOC details)
receiver_deploys.txt	All ClearReplica receiver ***deployment details***
receiver_errors.txt	All ClearReplica receiver ***error*** messages
receiver_start_stop.txt	All ClearReplica receiver ***start*** or ***stop*** messages
receiver_TOCs.txt	All ClearReplica receiver ***TOC details*** (version list of files contained in "replication "Packets.
receiver_warnings.txt	All ClearReplica receiver ***warning*** messages

Since multiple ClearReplica "receiver" processes can be started (one for each **receiver.conf** file), each process has its own logging space (set of logging files). One "receiver" process might run against "**receiver.conf**" while another runs against "**receiver_atl.conf**" and still a third might run against "**receiver_london.conf**". These processes would produce log files as illustrated below:

Running the clearreplica_receiver against these files produces…		
receiver.conf	**receiver_atl.conf**	**receiver_london.conf**
receiver_all.txt	receiver_all**_atl**.txt	receiver_all**_london**.txt
receiver_deploys.txt	receiver_deploys**_atl**.txt	receiver_deploys**_london**.txt
receiver_errors.txt	receiver_errors**_atl**.txt	receiver_errors**_london**.txt
receiver_start_stop.txt	receiver_start_stop**_atl**.txt	receiver_start_stop**_london**.txt
receiver_TOCs.txt	receiver_TOCs**_atl**.txt	receiver_TOCs**_london**.txt
receiver_warnings.txt	receiver_warnings**_atl**.txt	receiver_warnings**_london**.txt

"Relocated" ClearReplica directories

By default all ClearReplica writes (other than to the VOB itself) are performed within directories underneath the **m_bay** directory that is located in the ClearTrigger **depot** directory. For organizations that wish to run ClearTrigger and ClearReplica from a read-only share, media or device, both ClearTrigger (12.5 and higher) and ClearReplica (9.2 and higher) allow for ***directory relocation***. Directory relocation can also useful when managing disk space or improving performance.

All directories written to by ClearReplica can be relocated to another directory for writing. The ClearReplica directories (listed relative to the ClearTrigger "depot") and the binaries that write to them are listed in the table below:

"depot" relative directory	Binaries that write to this directory			
	clearreplica_shipper	clearreplica_receiver	clearreplica_shipout	cleartrigger
daemon	✓	✓	☐	☐
m_bay_hold_out	✓	☐	☐	☐
m_bay_ship_out	✓	☐	✓	✓
m_bay_epocs	✓	☐	☐	☐
m_bay_logs	✓	✓	☐	☐

✓ **= Directory is written to by this process**
☐ **= Directory is NOT written to by this process**

To "relocate" the directory so it writes to another directory create and populate a file named "**_clearreplica_relocated.txt"** in that directory where the contents of **_clearreplica_relocated.txt** are at a minimum a single line that contains the **relocated destination directory** to which you want to be the new destination of the relocated writes. Only the ***first*** line is evaluated and expected to contain the relocated destination directory, all other lines are treated as a comment and ignored.

When selecting a relocated destination directory consider these imitations:

- Only absolute directory names should be used
- Drive resident names are allowed (Windows)
- UNC directory names are NOT allowed (Windows)
- Network directory names are allowed (UNIX)

Examples of valid content for a **_clearreplica_relocated.txt** are below:

Receiver Log file	Containing entries in chronological order
Windows local	C:\some_other\logs
Windows mapped drive	J:\some_other\mapped_shipout
UNIX	/some_other/directory/logs
UNIX network path	/net/machine/some_other_dir/logs

Note: Remember the "**_ship_out**" directory is written to by **ClearReplica *and*** ClearTrigger, so if redirecting the "_ship_out" directory you must also inform ClearTrigger of its relocation by setting the appropriate **cleartrigger_relocated_shipout** alias in the ClearTrigger clearbits file (further information on ClearTrigger Relocation Alias Enhancers can be found in the section so named in the ClearTrigger Administration Guide).

"Receiver Email" Notification

Since ClearReplica "receiver" processing is fully automated and unattended it produces an email file that you can elect to send when there are versioned files that cannot be "received" at the receiving location. The file contains the "email body" that describes the file versions that were in a packet, but could not be "received" because either:

- the version already exist in the receiving VOB
- the version could not be created because of an existing "reserved checkouts"

When either situation arises, this email body is created and stored in a file located at the value of the environment variable **CLEARREPLICA_RECEIVER_EMAIL_FILE** and the value of the environment variable **CLEARREPLICA_RECEIVER_HAS_EMAIL** is set to the "yes" value.

So in your **Post_Receive_Trigger** script you can elect to send the email if **CLEARREPLICA_RECEIVER_HAS_EMAIL** is set t the "yes" value. The email body one of two forms.

If file versions already exist in the receiving VOB then the email will resemble the one below:

```
#####################################################
[Mon Apr 07 12:12:46 2008 UTC -05:00] -- Already Existing files for package
(epoc.108.atlanta.atlanta.crc) \m1_bup is below:
#####################################################
These file versions already exist at this location and will not be overwritten:

   \m1_bup\src\abs_arc1.doc@@\main\0
   \m1_bup\src\abs_arc1.doc@@\main\1
```

If file versions could not be created because of "reserved checkouts" then the email will resemble the one below:

```
#####################################################
[Mon Apr 07 12:12:49 2008 UTC -05:00] -- Trouble replicating the file
versions for package (epoc.108.atlanta.atlanta.crc) \m1_bup below:
#####################################################
These file versions exited in the package, but could not be replicated. For more
details view (l:\cleartrigger\depot80\m_bay\logs\receiver_warnings.txt):

   \m1_bup\src\abs_arc2.doc@@\main\2
   \m1_bup\src\abs_arc2.doc@@\main\3
```

If both cases exist then the single email body will contain both sections. You can read and or modify the email body contents before sending the email in your **Post_Receive_Trigger** script.

ClearReplica Packet Sizes

All ClearReplica Packets are encrypted and compressed before they are sent. The maximum size for an individual packet is **2 Gig**. If you are running on a 64-bit architecture and need the ability to create larger packet sizes you may request special 64-bit binaries from ABS that have a **100 Gig** limit. **ClearReplica automatically breaks up larger packets to fit under its shipper maximum to send to the receiver location.** The exception to this case is when a ***single versioned file*** to be included in the packet is by itself larger than the packet limit as the data for a ***single version*** is not split between packets.

When a packet to be built by a **ClearReplica** "**shipper**" process would exceed the maximum allowed size for that operating system then a warning is written to the "shipper" warning log that reports that the resulting key would be too large and not sent. If such a warning is observed then the associated **replication definition** should be change to produce smaller keys (cover a smaller period of time or a smaller portion of the VOB).

When a packet received by a **ClearReplica** "**receiver**" process exceeds the maximum allowed size for that operating system then a warning is written to the "receiver" warning log that reports that the received packet was too large and tossed. If such a warning is observed then sites ClearReplica administrator should inform the administrator of the shipping/sending site that the associated **replication definition** should be change to produce smaller keys (cover a smaller period of time or a smaller portion of the VOB). This might be the case if site A (with a 64-bit ClearReplica shipper running a on a SUN 59 64-bit machine) produces a 3 Gig packet and sends it to site B (with a HP 32-bit machine) that has a 2 Gig packet limit. In such cases "receiver" warning should prompt the administrator at site B to inform the administrator at site A to send small packets by adjusting the associated **replication definition**.

Upgrading from earlier ClearReplica versions to 12.0

The packet format stayed the same from ClearReplica 9.3 to ClearReplica 11.0.3. The packet format was changed in 12.0 to accommodate new features. ClearReplica receiver 12.0 or higher must receive packets from ClearReplica shippers running 12.0 or higher. Additional information was added to ClearReplica 12.0 packets though its size was actually reduced. Still, an upgrade from earlier versions to 12.0 or higher is very straightforward. Only the binaries need to be replaced. ClearReplica 12.0 or higher receivers should receive packets from ClearReplica shippers running 12.0 or later software so all ClearReplica clients in the topology must be upgraded to at least 12.0. To upgrade from earlier ClearReplica versions to ClearReplica 12.0 or higher follow these steps:

- Stop all ClearReplica ***shipper*** processes in your topology. Be sure to use the proper –stop commands to ensure that no packet is partially processed. This also provides a bit longer for the receiver to process any pending packets.
    - Remove any existing unprocessed **epoc.*.*.*.cr** and **epoc.*.*.*.pack** files in any shipper bays (m_bay/_hold_out, m_bay/epoc or their redirected directories). These files will be rebuilt with the new binaries.
      Do not remove any **epoc.*.*.*.epoc** files.
- Stop all ClearReplica ***receiver*** processes in your topology. Be sure to use the proper –stop commands to ensure that no packet is partially processed.
    - Remove any existing unprocessed **epoc.*.*.*.cr** and **epoc.*.*.*.pack** files in any receiver directories (defined in your receiver.conf files). These files will be rebuilt with the new binaries.
      Do not remove any **epoc.*.*.*.epoc** files.
    - Change any appropriate **replication_receiver_dirs** entries in that locations **receiver.conf** file. Refer to the section **Converting pre 12.0 receiver_directory definitions to 12.0**.
- Replace earlier binaries with 12.0 or higher binaries in your topology.
- Restart all ClearReplica process (shippers and receivers) in your topology.

