

Data flow to ensure always having a consistent Point-In-Time (Restartable) Copy of data at the remote site (v1.1)

## Data flow to ensure always having a consistent Point-In-Time (Restartable) Copy of data at the remote site (v1.1)

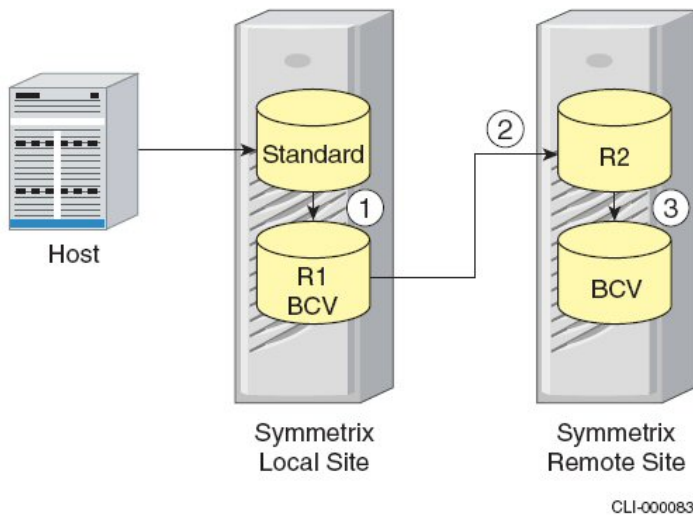
### 2 Site – Single hop configuration

Author.....: Noelle Milton Vega

Date.....: February 21, 2005

Note: Although this paper uses EMC terminology in its discussion,  
the concepts apply generally to all replication implementations.

Figure 1 shows that the copy path for a single-hop configuration is from the local BCV pair (1) to the SRDF pair (2) to the remote BCV pair (3). The remotely associated BCV holds the DBMS restartable copy.



**Figure 1. Automated Data Copying in a Single-Hop Configuration**

The figure above shows the copy configuration for a **single-hop disaster recovery solution**. Whether single-hop or multi-hop disaster recovery solution is employed, the idea is the same: always have a copy of data on remote disk devices which, from the perspective of the application, is point in time consistent. This means that the application can be cleanly started at the remote site from the data that resides on the disk devices at that remote site. This is what is meant by a "restartable copy of data". The copy of data may not be the most up-to-date but, however out-of-date it may be, it *is* a restartable copy. In the configuration shown above, each of the 4 devices is necessary to ensure that some point-in-time consistent restartable copy is always available at the remote site. We discuss the purpose of each next.

#### **Single-hop disaster recovery solution** (see diagram above)

As shown above, for each standard (STD) volume that is to be remotely replicated, two TimeFinder mirrors (one in each Symmetrix), and one SRDF mirror are necessary (though not all mirrors are *established* at the same time). Each of the four Symmetrix Logical Volumes (SLV) shown serve a specific purpose in this configuration. We discuss them here.

- (1) **STD Device** – This is the device that the host sees, and reads and writes data from and to. Except for write I/O transactions **on-the-fly**, this device always contains the most up-to-date copy of the data in production. That said, this copy can never be counted on to restart an application in the case of failure. The reasons are these: (i) in the case of disaster at the local site, this copy wouldn't be unavailable; (ii) on-the-fly I/O transactions (i.e. **as yet uncommitted write I/O's**) make this copy *inconsistent* from the view point of the application when a failure occurs; and (iii) even if we were to ignore on-the-fly transactions, you cannot know whether the data that *has* successfully made it to this device and all others used by the same application, collectively form a *consistent point-in-time* data set... that is, whether it forms a data-set from which the application can be easily **restarted**. This is why we need the BCV/R1 device.

Data flow to ensure always having a consistent Point-In-Time (Restartable) Copy of data at the remote site (v1.1)

(2) **BCV/R1 Device** – When *established* as a mirror to the STD device, the BCV/R1 device contains an exact copy of the data on the STD device. Through admin coordination, if we quiesce the I/O's going to this mirror by momentarily shutting down the application (or in the case of an Oracle database, momentarily put it in hot-backup mode) and unmounting the associated filesystems, we can then *split* the mirror and ensure a consistent point-in-time copy of data on the BCV/R1 device. Once the TimeFinder split operation is complete, we can resume normal production operation (i.e. mount the filesystems and start the application) – but again, with the BCV/R1 device split-off and thus also now read/write enabled on the FA port to which it is mapped. Note that once the application is restarted (against the data on the STD device), the copy of the data on the BCV/R1 device is no longer the most current, but it *is* point-in-time consistent (restartable). This is the data (on BCV/R1) that will be replicated to the remote Symmetrix R2 device, via the SRDF link.

You might ask, "To save storage, why can't we bypass the BCV/R1 device and, through SRDF, establish a link directly between the STD device and the remote R2 device?". In other words, why can't we do this: (STD/R1 <----srdf----> R2). In theory this is possible, but there are a couple of issues with that configuration. First is the issue of performance. Maintaining a direct link between the STD device and the remote R2 device introduces an application performance penalty caused by distance propagation delay, link speed, buffer credit flow control in the case of a distance SAN (etc.). In other words, an application I/O cannot be acknowledged until it first fulfills the commit requirements dictated by the Synchronous or Asynchronous policy in force. Second, an SRDF link has a higher probability of failing than an internal TimeFinder link, and you do not want your application to stop on I/O busy/denied because it cannot be propagated according to the synch/async policy. So this is why it is necessary to employ an intermediate BCV/R1 device – performance, and reliability.

*Note that, for the same performance reasons described previously, when the STD and BCV/R1 devices are established (via TimeFinder), the BCV/R1 and R2 SRDF device pair should never be established, since that would require application writes to be propagated to the remote site in addition to the local BCV/R1 device. They should be explicitly suspended via the SRDF symrdf(1M) command but, in any event, will be unable to communicate with the remote R2 device anyway since the BCV/R1 device, when it is established with the STD device might be I/O disabled at the source FA port (read/write disabled).*

Data flow to ensure always having a consistent Point-In-Time (Restartable) Copy of data at the remote site (v1.1)

- (3) **R2 Device** – The next question you might ask is “Why is it necessary to employ an intermediate R2 device? To save storage, why can’t we go directly from BCV/R1 to BCV/R2 when copying the consistent point-in-time copy from the local to remote site?”.

To answer this, lets restate the goal: at all times, there should be a consistent point-in-time (i.e. restartable) copy of the data at the remote site, even if that copy is somewhat out-of-date.

Imagine that we eliminate the R2 device from the design above, and instead have the following direct SRDF link: (BCV/R1 <----srdf----> BCV/R2). The BCV/R2 device contains a consistent point-in-time (i.e. application restartable) copy of the data at the remote site. When we actually *establish* the (BCV/R1 <----srdf----> BCV/R2) srdf link, the consistent point-in-time copy that existed on the BCV/R2 device is now gone, since tracks on BCV/R2 are being updated. If, for some reason, the remote copy process cannot be completed (say a disaster at the source end, or a severed DWDM optical cable), you no longer have met the goal of ***always*** having a consistent point-in-time (i.e. restartable) copy of the data at the remote site. Moreover, even though it is likely that the copy/update process will succeed without problems most of the time, the fact remains that, for the period beginning from the start of the copy/update to the end, you have no consistent point-in-time during that time period (because you are transitioning from one consistent point-in-time copy to the next). So during that copy/update window, you have nothing to fall back on at the remote site, should it suddenly need to become live. So this why we need to employ an intermediate R2 device. We first make a copy to the R2 device, and leave the consistent point-in-time copy that is resident on the remote BCV alone (by virtue of making sure that those two volumes are in a *split* state). When we are sure that the copy to the R2 device has successfully completed, we then can establish a copy FROM the R2 device to the (remote) BCV device via the Timefinder *symmir(1m)* command.

Copy and link established / split sequence:

Step 1: Create a local consistent point-in-time copy.

```
(STD)      ← TimeFinder Established           → (BCV/R1)
(BCV/R1) ←          SRDF Split                 → (R2)
(R2)      ← TimeFinder Split or Established    → (BCV)
```

Step 2: Step 1 complete. Next, transfer the local copy to an intermediate staging area at the remote site.

```
(STD)      ← TimeFinder Split                 → (BCV/R1)
(BCV/R1) ← SRDF Established                   → (R2)
(R2)      ← TimeFinder Split                 → (BCV)
```

Step 3: Step 2 complete. Next, duplicate the consistent point-in-time copy from the intermediate staging area at the remote site to its final resting volume at that remote site.

```
(STD)      ← TimeFinder Split or Established    → (BCV/R1)
(BCV/R1) ← SRDF split                         → (R2)
(R2)      ← TimeFinder Established           → (BCV)
```

Note: “*Split or Established*” means that, at that stage in the process, that link can be split or established (in other words, it makes no difference).