TECHNICAL BRIEF

Lifecycle Fixed Content Manager 100 Series solution

Archival WORM storage on magnetic disks

DECEMBER 2004

1. Executive summary

Write-once-read-many storage has traditionally encompassed paper, microfilm, microfiche, optical disks and WORM tape. More recently, government regulators in the most strictly controlled financial sectors have accepted rewritable magnetic disk storage as compliant archival WORM storage, provided that the storage firmware does not allow regulated data to be modified or deleted. This has made it possible for the storage system itself to enforce record retention policies that were formerly handled in an expensive and error-prone manual fashion.

The Lifecycle Fixed Content Manager 100 Series solution is such a system. It provides a compelling solution to the records retention problem, making regulator-compliant disk-based storage safe, economical, easy to use and continuously accessible through future generations of hardware evolution.

2. Archiving and retention considerations

The generation of massive amounts of electronic information, together with new regulatory requirements, is driving the archive process to center stage in most enterprises. This, along with the demand for rapid search and retrieval of archive data, is driving the need for a new class of archive storage solution — an intelligent archive solution. More than 90 percent of archive data is stored on tape today, where StorageTek® is a market leader. As the need for disk-based archives grows, customers can rely on StorageTek to provide total information lifecycle management solutions.

3. Lifecycle Fixed Content Manager 100 Series solution

The Lifecycle Fixed Content Manager 100 Series solution is a storage firmware. It is a software product that, when married with qualified hardware, creates a storage appliance that meets statutory requirements for archival WORM storage. The Lifecycle Fixed Content Manager 100 Series solution has only three interfaces: the NFS and CIFS network file sharing protocols [1, 2] and a Web-based control console [3]. The operations allowed through the consol are tightly constrained, to make it impossible for the customer to violate retention policies. For example, the customer is not able to reset the clock. The Lifecycle Fixed Content Manager 100 Series storage system relieves the customer of the need to manually enforce retention policies, and dramatically reduces the work needed to maintain the accessibility of retained records throughout their designated retention periods.

3.1 Storage nodes and access nodes

Lifecycle Fixed Content Manager 100 Series solution is designed from the ground up for economical long term storage. In traditional storage systems there is no clear separation of the disk hardware from the storage software; disk addresses are managed directly by file systems and databases. Lifecycle Fixed Content Manager 100 Series solution is founded on object-based storage (CAS). The content address is a guaranteed unique identifier (GUID) that is derived directly from the contents of each block of data. As a Lifecycle Fixed Content Manager 100 Series system grows and new kinds of hardware are added, the GUID addresses used to locate stored data and metadata do not change.

A Lifecycle Fixed Content Manager 100 Series solution has two kinds of components: storage nodes and access nodes. A system diagram is shown in **Figure 1**.

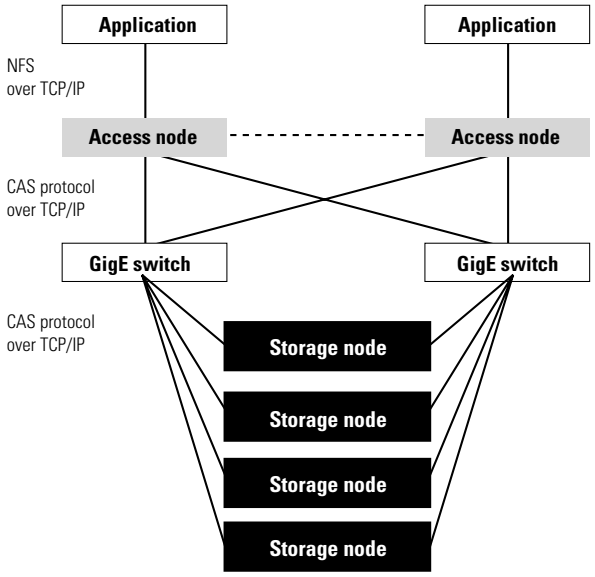

Figure 1: *The Lifecycle Fixed Content Manager 100 Series storage appliance. The Lifecycle Fixed Content Manager 100 Series solution software ties together a cluster of storage servers, using standard hardware and mass-market disks, into a fault-tolerant and scalable storage system. The Lifecycle Fixed Content Manager 100 Series solution access node software runs on servers that translate between familiar file sharing protocols and the secure client/server protocol used to talk to the storage nodes.*

The nodes use standard hardware and disks — robustness and failure tolerance derive from sophisticated service clustering algorithms. Data is always replicated on more than one storage node, and hardware failures are invisible to the Lifecycle Fixed Content Manager 100 Series solution storage client.

The Lifecycle Fixed Content Manager 100 Series solution storage consists of a collection of storage nodes. The system can be configured with as few as four storage nodes, and can grow in blocks of four nodes. The system automatically and invisibly adjusts as storage nodes are added or removed, reassigning responsibility for different GUID address ranges and migrating data as needed. New hardware may be very different from existing hardware. This kind of storage scheme is much like biological growth; new cells are added and old cells die, but the organism persists. There is never a need for a forklift upgrade. All of the data in the storage system remains continuously available, despite the fact that the hardware has changed.

The Lifecycle Fixed Content Manager 100 Series solution access nodes are used to present the Lifecycle Fixed Content Manager 100 Series solution storage in a familiar format. They translate between sophisticated SSNAP [4][1] storage protocol used to talk to the storage nodes and the familiar NFS and CIFS file sharing protocols. Because the blocks of data are uniquely named based on their contents, the access nodes are able to avoid repeatedly storing duplicate data, thus saving storage space and communications bandwidth within the system.

The access nodes also run a Web-based management console, which allows nodes to be added and removed, and file systems to be created and made accessible to selected clients. There is very little management needed in the traditional sense because all file systems draw space as needed from a single common pool of virtualized storage and failures are handled transparently. Users can create special WORM file systems which enforce retention requirements (for regulatory compliance), and can schedule snapshots of file systems, also with enforced retention (for backup).

4. Magnetic disk-based WORM

The disposition of regulated records which no longer need to be kept has traditionally been handled manually. With paper, individual documents which are no longer required by statute can be selected to be destroyed. With optical and tape media, records are typically aggregated and media not destroyed while they hold any data that is still required to be kept. Magnetic disk WORM storage is more like paper in this respect, since individual records can be (if desired) deleted once their retention requirements have been met.

For applications that employ disk for WORM storage, the storage system must provide the ability to:

- Store records that are guaranteed not to change
- Store records containing fixed references to other records
- Mark individual records with a retention period
- Keep records accessible during their entire retention period
- Enforce non-deletability during the retention period
- Allow the retention period of a protected record to be extended

In addition, for a storage system which is used to meet statutory retention requirements, such as SEC rule 17a-4, there cannot be a super-user who can override the retention requirements: the storage system is not deemed to enforce statutory retention requirements if, for example, some officer of the company has the authority to delete a record before its set retention has concluded. This is analogous to the situation in the world of paper retention, in which policies and procedures prevent even high officers in a company from violating statutory retention requirements. For ordinary "best practices" retention, on the other hand, businesses merely wish to tighten control of who is authorized to override preset retention policies. For example, if the Lifecycle Fixed Content Manager 100 Series system is being used to hold all of a company's file system backups, it is prudent to carefully control who (if anyone) has the authority to prematurely delete backup records.

5. Creating WORM file systems

The Lifecycle Fixed Content Manager 100 Series solution Web console is described fully in the Lifecycle Fixed Content Manager 100 Series solution operation guide. It provides facilities for monitoring the health of the Lifecycle Fixed Content Manager 100 Series storage system and sending e-mail alerts to the system administrators when they need to add more storage capacity (i.e. additional storage servers). It also provides the interface for creating new file systems. The Web console runs off the Lifecycle Fixed Content Manager 100 Series solution access nodes, which is also where the NFS and CIFS file sharing interfaces run. If desired, independent file system administrative domains, called access node groups can be defined. All the access nodes in all groups can share a single virtualized storage pool, while maintaining complete privacy and security for the file systems administered within each group.

[1] The Secure Self-Naming Archive Protocol [4] is a parallel session-level client-server protocol which insulates storage clients (such as access nodes) from storage node issues such as failure recovery, addition of storage and offsite replication.

Figure 2 shows the file systems page on the Web console. File systems are made visible to applications by *exporting* them. In this example, the file system e-mail archive is currently exported on **portal-0.example.com.** Which portal a file system is exported on can be freely changed — the file system is independent of the particular access node. The file system information indicates that the e-mail archive file system is a WORM volume.

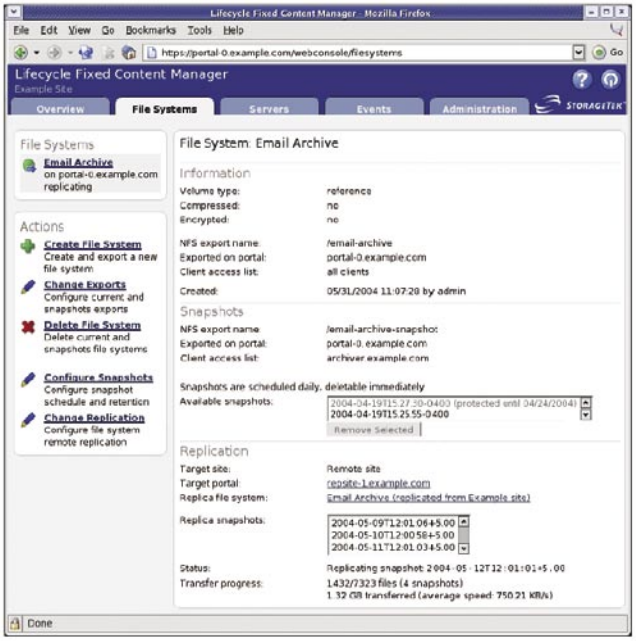

Figure 2: Web console file system management. The file system information for e-mail archive is shown. This is a compliance WORM volume that can never be deleted, and uses a WORM protocol to allow NAS clients to store data with a retention requirement that is enforced by the storage system.

This means that it supports a protocol for marking records for retention through the file system interface — this is described in section 5. This file system is also a *compliance volume*. This means that even the most highly privileged system administrator cannot override the retention requirements that applications specify at the time that the records are stored. Note that the action "Delete File System" is grayed, to indicate that this is an operation that is not allowed. The storage system also provides a mechanism for applications to verify, through the file system interface, that records have been written correctly to the intended compliance volume (see Section 7). In a compliance deployment, an application server (e.g. an e-mail archive server) would typically have exclusive access to a compliant file system.

This could be achieved by providing a dedicated connection to a dedicated portal: the client-facing network interface on the access node is connected only to a dedicated network interface on the application server.[2] Alternatively, a similar effect can be achieved without dedicating portals by using the client *list* export control.

In the example illustrated in **Figure 2**, only the machine **archiver.example.com** is allowed to access the **e-mail archive** file system. The popup menu shown in **Figure 3** was used to create the **e-mail archive** file system. If "WORM file system" was not selected, this would have been created as an ordinary file system without enforcement of retention requirements.[3] A *compliance volume* is one which cannot be deleted; an e*nterprise volume* is one that can be deleted only by a designated enterprise administrator. Both volumes are otherwise identical in enforcement of retention, as described in Section 5. The type of a volume is selected when the volume is created, and cannot be changed. At the time of installation the system can be configured only to create WORM volumes.

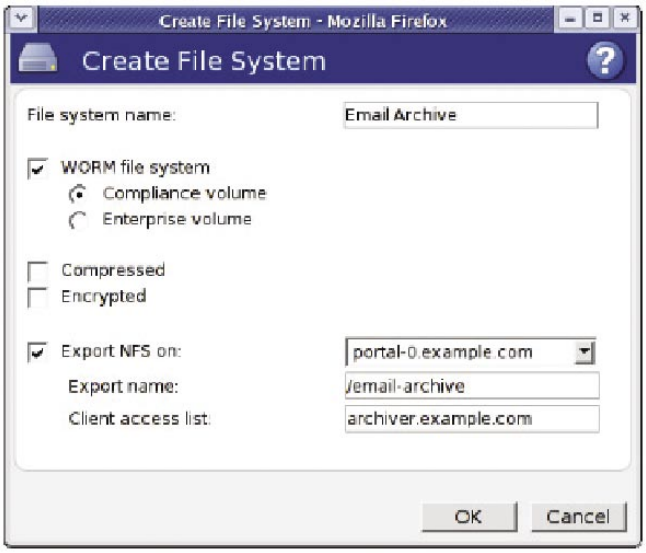

*Figure 3: Web console file system management. The file system information for **e-mail archive** is shown. This is a compliance WORM volume that can never be deleted, and uses a WORM protocol to allow NAS clients to store data with retention requirements that are enforced by the storage system.*

[2] Routing between interfaces on the application server is turned off.

[3] If snapshots are enabled, retention requirements can be enforced on the file system snapshots even if none are enforced on the file system itself [3].

Compression and/or encryption can be turned on for a file system when it is first created (and cannot be changed). This is a transparent feature which takes place below the visible file system level, though it has some impact on performance. Although compression is generically useful for reducing storage needs, in many situations data being stored is already compressed and so further compression is not helpful. Typically, the most significant compression comes from the *coalescence* property of the Lifecycle Fixed Content Manager 100 Series storage system: when files are written, the access node breaks them up into blocks in a predictable and data-aware manner, and duplicates of the same block (across the entire storage system) share the same storage space. If encryption is used, the encryption file system is stored in the storage nodes in a manner that is secure from any other access node group.

6. Using WORM file systems

Only one access node is currently supported by the Lifecycle Fixed Content Manager 100 Series solution for providing access to WORM file systems. This protocol is compatible with the scheme defined by "Network Appliance" [5] for writing WORM files through a standard NFS or CIFS interface. Turning a network file sharing protocol into a WORM protocol involves four issues:

- How do you indicate that a record is complete?
- How do you indicate how long a record must be kept?
- How do you extend the retention period?
- How do you treat directories?

6.1 Indicating record completion

The first issue arises because the NFS protocol has no file close operation — for a CIFS-only scheme this would not be a problem. We must interpret some other operation as indicating that we have finished writing a file and it should no longer be modifiable. This issue is addressed in the supported WORM protocol by creating a file in a writable state, writing it and then making it read-only for all users. The file is complete when it has been made read-only. For a Lifecycle Fixed Content Manager 100 Series solution WORM file system, call complete files are guaranteed to have been committed safely to fault-tolerant storage.

6.2 Setting retention periods

A file that has been made read-only is referred to as *protected*. If no retention period has been specified before the file is protected, it is protected permanently: the storage system will prevent it from ever being deleted. If, on the other hand, the file's *last access time* has been set to a date in the future before the file is protected, the file is only protected until that date.

6.3 Extending retention periods

In some cases, retention periods need to be extended. This might happen as a result of legal proceedings, for example, or because a record has a variable retention requirement which must be extended periodically until the true end date is known. For example, a record that must be kept for the lifetime of a bank account plus seven years would start off with a retention requirement of seven years and have this requirement extended year by year. In a WORM file system, a protected file cannot be changed except to have its retention date extended. This is done in the supported WORM protocol by setting the protected file's *last access time* to a new date that is later than the previous setting. If a file is already protected permanently, the access time cannot be changed.

6.4 Directories

WORM applications normally maintain their own database of records. The role of the storage system is to keep the records inviolate for the necessary retention period and to *allow the records to contain fixed references to other records*. The latter goal can be achieved in a file system context by never allowing the path name of the file to change once the file has been protected. This guarantees that stored path names will always provide access to the correct record. In the supported WORM protocol, this goal is achieved by never allowing directories to be moved or renamed, and by never letting protected files be moved or renamed.

6.5 WORM protocol summary

In summary, for a WORM file system:

- Directories cannot be moved or renamed.
- A writable file can be protected by making it read-only.
- If the *last access time* has not been set, the file is protected forever.
- If the *last access time* is set to a future date before the file is protected, the file is only protected until that date.
- The retention date of a file can be extended (but not shortened) by changing the *last access time*.
- Protected files cannot be moved or renamed.
- After its retention date has passed, a file can be deleted.

7. Usage examples

This section provides examples of the use of the Lifecycle Fixed Content Manager 100 Series solution WORM file systems from UNIX and from Windows. We advise the use of the Lifecycle Fixed Content Manager 100 Series solution NFS interface for both UNIX and Windows since it has higher performance than the CIFS interface.[4]

7.1 UNIX NFS

Here is an example of a command line interaction with a Lifecycle Fixed Content Manager 100 Series solution WORM volume from a UNIX NFS client:

```
/> mkdir test
/> cd test
/test> umask 0
/test> touch -a -t 200801010000 t1
/test> ls -lu t1
-rw-rw-rw-1 nhm staff 0 1 Jan 2008
t1
/test> chmod a-w t1
/test> rm t1
override r--r--r--nhm/staff for t1?
y
rm: t1: Read-only file system
/test>
```

This example illustrates creating a writable file which has its access time in the future and protecting it and then trying to remove it. When the specified retention date is reached, the file becomes deletable but the contents are not changeable. Once the retention date has passed, a file may be protected again, with a new retention period, by making it writable, changing the retention date, and making it read-only:

```
/test> chmod a+w t1
/test> touch -a -t 201001010000 t1
/test> chmod a-w t1
/test>
```

A protected file may have its retention date extended, but may not otherwise be changed. For example,

```
/test> touch -a -t 201006010000 t1
/test>
```

Would extend the retention set in the previous example.

Unprotected files may be freely moved, renamed and deleted. Empty directories may be deleted. Directories and protected files may not be moved or renamed:

```
/test> umask 0
/test> touch t2
/test> rm t2
/test> mkdir d1
/test> rmdir d1
/test> mkdir d2
/test> mv d2 d3
mv: rename d2 to d3: Read-only file
system
/test> mkdir d4
/test> touch d4/t3
/test> rmdir d4
rmdir: d4: Directory not empty
/test>
```

Files may be created in a read-only state. If a file is created read-only, it is not protected. The only way to verify that a file is protected is to try to change it or its attributes — the fact that a file is read-only does not imply that it is protected. To protect a file, it must transition from a writable to a read-only state. Unprotected files may be freely moved or renamed. Hard-links cannot be created to a protected file. A file that has never had its *atime (last access time)* set, or that has had the atime set into the past (according to the server's time) will be protected permanently if it is set read-only; otherwise it will have a finite retention period. Because of the UNIX/Windows year 2038 problem,[5] access times later than the beginning of the year 2038 cannot be set. A mechanism for setting retention dates past this point will be supported in a future version of this product.

7.2 Windows NFS and CIFS

WORM file systems operate in exactly the same manner as described above through Windows NFS and through CIFS. All files have an access time associated with them and can be protected by setting them read-only. This could be illustrated using **`cygwin`** scripts that look almost exactly like the UNIX examples given above, or with DOS batch scripts using commands such as **"`attrib +r t1.`"** Instead we will give a purely graphical example using Windows NFS, to emphasize the point that, using Microsoft's Services for UNIX, NFS can be used as a normal Windows file system. A Samba/CIFS example would look virtually identical.

[4] Note that NFS file systems can be used as conveniently as native Windows file systems if Microsoft's Services for UNIX package has been installed (available as a free download).

[5] Dates in both UNIX and Windows are stored as a number of seconds since January 1, 1970. Since most programs deal with these times as signed 32-bit numbers, the latest date that can be represented is January 18, 2038, 22:14:07 UTC.

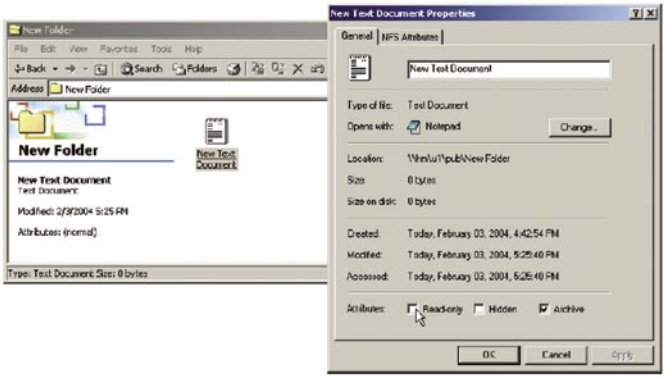

*Figure 4: Using an NFS WORM file system through Windows. Using the Windows graphical interface we create a **new folder** on the WORM file system and within it a **new text document**. We protect the file from ever changing by setting it read-only without first setting the access time in the future.*

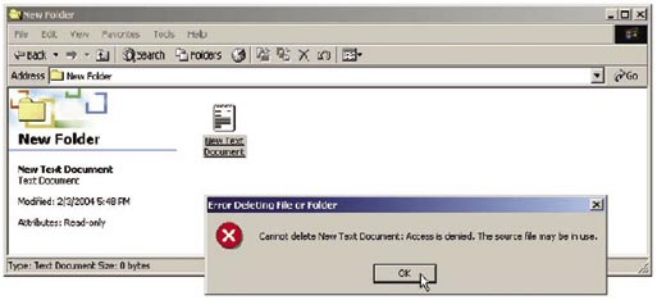

Figure 5: Using an NFS WORM file system through Windows. Once the file has been protected, any attempt to delete it will cause an "access denied" error.

In this example (illustrated in **Figures 4 and 5**), an exported WORM file system NFS share **`\u1\pub`** is available from a portal named **`nhm`**. A window on this share is opened by running the command **`\\hm\u1\pub`** from the start menu. A new folder called "**`New Folder`**" is created on this file system (from the file menu), and within it a text document called "**`New Text Document`**" is created (by right-clicking within the folder on **`New Text Document`**). We right-click on the document to get the properties menu to reach the point shown in **Figure 4**. Setting this document read-only will protect it permanently, since we have not changed the access time, which is exactly the same as the *last modified time*. In **Figure 5**, we see the result of trying to delete the protected document. It is worth noting that folders (directories) cannot be renamed on a WORM volume, and so creating a **`New Folder`** in this manner is not very useful. WORM volumes would normally be created directly by applications using file system operations.

8. Content certificates

The Lifecycle Fixed Content Manager 100 Series solution is an object store, based on guaranteed-unique identifiers and content addresses. This fact is normally completely hidden from an application when the storage is accessed via traditional file sharing protocols. For regulatory compliance purposes, the CAS nature of the storage system provides additional assurance that records have been written accurately: the cryptographically strong hashes used to generate the block names are computed by the portal and recomputed by the system.

The system refuses to store a block of data if the block name does not match the data. If a record of block name hashes is made (or of hashes of hashes), this record can be used later to prove with mathematical certainty that files have not changed [6, 7]. Furthermore, the content certificate can be used by applications to verify that they are talking to the right file system and that files are, in fact, protected.

To make object level information visible through a file system interface, Lifecycle Fixed Content Manager 100 Series solution WORM file systems provide synthetic *content certificates*. These content certificates are XML documents that expose object-level meta-information about files stored in Lifecycle Fixed Content Manager 100 Series solution. No extra information is stored in Lifecycle Fixed Content Manager 100 Series solution in order to support this feature: meta-information that is normally part of the underlying storage scheme is simply made visible here, turned into XML documents on demand. For the file **`/pathname/myfile`**, a corresponding content certificate is made visible in the synthetic file **`/pathname/.object/cert/myfile`**. The **`.object`** subdirectory of every directory is reserved for this purpose. A typical content certificate is exhibited in **Figure 6**. The format of the XML schema used here is documented in detail elsewhere [8]. Content certificates allow direct verification that a file is part of a protected WORM volume, that the file itself is protected and for how long. *If a certificate is separately recorded, this record can be used to later prove, with certainty, that the file contents have not changed.* In fact, any set of files can later be proved not to have changed by recording a single cryptographic hash of appropriate information from a corresponding set of certificates.

```
<?xml version= “1.0”?>
<ContentCert xmlns=”http://www.example.com/schema/ContentCert”
      Xmlns :xsi= ”http://www.w3.org/2001/XMLschema-instance”
      Xsi:schemaLocation=
      ”http://www.example.com/schema/ContentCert
      http://www.example.com/schema/ContentCert.xsd”>

<volume>C34242B7AF4E6B9A</volume>
<namespace>AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAB=</namespace>
<handle>XG51d2NvbW1hbmR7XFBCc2vjdG1vbn1bMV17XHJ1ZnN=</handle>
<version>XBjb3VudGVy=</version>
<Stream number=“0”>
      <immediate>
         <content>e3N1Y31cb2xkc2VjdG1vb1tcdGh1c2VjIH4gXG==</content>
      </immediate>
</Stream>

<Stream number=”1”>
      <direct>
         <blockName>ZG93Ym94e1xjb2xvcmJveHtzZWNibHV1fxtcbWFrZXJ+</blockName>
      </direct>
</Stream>

<WORMCompatible enterprise=”true”>
      <protected expiry=”2004-12-31T08:00:00Z”/>
<WORMCompatible>

</ContentCert>
```

Figure 6: *Content certificate. Each file in a WORM file system has a content certificate. This provides information that allows an application to verify that the file is on a particular volume (independent of the file system name), what kind of volume (e.g., compliance) and contains information that, if recorded, can be used later to prove to a third party that the contents of the file have not changed.*

9. Summary

The Lifecycle Fixed Content Manager 100 Series solution provides an easy to use, well controlled and safe storage system optimized for large-scale and long-term disk-based storage of archival data. Individual records can be marked through a convenient file system interface with retention requirements, which are then enforced by the storage system itself. Retained records are guaranteed to remain safe and continuously accessible, even as storage is added and removed from the system and new hardware replaces old hardware.

10 End notes

[1] B. Callaghan, B. Pawlowski and P. Staubach, “NFS Version 3 Protocol Specification,” RFC 1813, June 1995.

[2] Storage Network Industry Association, “Common Internet File System (CIFS) Technical Reference 1.0,” SNIA, March 2002.

[3] StorageTek, “The Lifecycle Fixed Content Manager 100 Operation Guide.”

[4] “SSNAP Protocol Specification for Product 1.0.”

[5] C. Lueth, “WORM storage on magnetic disks using SnapLock Compliance and SnapLock Enterprise,” Network Appliance Technical Report TR3263, September 2003.

[6] U.S. Postal Service, “USPS Electronic Postmark White Paper,” Authentidate, Inc., September 2003.

[7] Surety, Inc., “Ensuring Record Integrity with AbsoluteProof, technical whitepaper,” Surety, Inc., 2003.

[8] StorageTek, “Lifecycle Fixed Content Manager 100: Content Certificates: Integrating Verifiability with Enterprise Business Logic.”

ABOUT STORAGETEK®

Storage Technology Corporation (NYSE: STK) is a $2 billion global company that enables businesses, through its information lifecycle management strategy, to align the cost of storage with the value of information. The company's innovative storage solutions manage the complexity and growth of information, lower costs, improve efficiency and protect investments. For more information, visit www.storagetek.com, or call 1.800.275.4785 or 01.303.673.2800.

WORLD HEADQUARTERS

Storage Technology Corporation
One StorageTek Drive
Louisville, Colorado 80028 USA
1.800.877.9220 or 01.303.673.5151

FT 0003 A 12/04