**Use cases**

|  |  |  |
| --- | --- | --- |
| Date | Comment | Version |
| Tuesday, September 27, 2011 | Initial document | 0.0.1 |
| Tuesday, September 27, 2011 | Updated lots of stuff | 0.0.2 |
| Tuesday, September 30, 2011 | Updated more stuff | 0.0.3 |
|  |  |  |

1. **Incorporate existing NAS storage shares into TFS infrastructure.**

User imports existing NAS (CIFS/NFS) data into TFS.

**Invariants**

* + - 1. Existing CIFS/NFS shares will be incorporated into TFS global name space.
      2. Contents of each share server/target will be “imported into” and ”shared using” respective directories.
      3. Existing file permissions and data will be kept.
      4. Single shares can be comprised of multiple CIFS and/or NFS backend storage units joined together.

**Actions**

1. User inputs credential and network share path (URL)
2. Spawn TFS import task
   1. Mount remote NAS share locally (/import/NAS/URL)
   2. Traverse share and add file + directory into TFS DB
      1. Generate encryption key
         1. Encrypt with system master key
      2. Profile each file
3. **Incorporate existing SAN storage shares into TFS infrastructure.**

User imports existing SAN (iSCSI) storage into TFS.

**Invariants**

* + - 1. Imported iSCSI LUNs may contain sparse file system and data.
      2. Imported iSCSI LUNs are virtualized by TFS as a single giant file which in term is the backend of a TFS virtual iSCSI target.

**Actions**

1. TBD
2. **Add new NAS storage shares into TFS infrastructure.**

User adds (new) NAS (CIFS/NFS) shares into TFS.

**Invariants**

* + - 1. TFS will create a TFS directory for its use
      2. Share can be used either for backup or additional storage

**Actions**

1. User inputs credential and network share path (URL)
2. Spawn TFS prep task
   1. Mount remote NAS share locally (/import/NAS/URL)
   2. Traverse share and add file + directory into TFS DB
      1. Generate encryption key
         1. Encrypt with system master key
      2. Profile each file
3. LUNs may be deduped, compressed, and encrypted in place.**Tape support**
   1. Import existing archival storage (tape) into cloud.
   2. Data from tape will be restored directly into TFS CIFS or NFS share.Export backup into cloud.
      1. Supported through TFS virtual block device (iSCSI virtual tape library LUN)
4. Data backed up may be deduped, compressed, and encrypted. **Backup/recovery**
   1. Implicit backup – every file in TFS store is backed up continuously to backend storage or cloud
      1. Policy based backup depending on duplication and archival requirements.
      2. QoS maintained during backup to cloud.
      3. Efficient backup of compressed and delta data
   2. Backend storage types
      1. Cloud.
   3. CIFS/NFS shares (e.g. remote sites for offsite disaster recovery or multiple paths to data)Security
      1. Data to backend storage may be encrypted (optional).
   4. Compression/de-dupe
      1. Data to backend storage may be compressed and de-duped (optional).
   5. Managed snapshots for data in shares and LUNs.
      1. Quick recovery from data corruption.
5. **ILM and tiering support**
   1. TFS will tier file storage based on usage patterns (learning) and data type
      1. Hot - (online/local store) cache only?
      2. Warm - (nearline/network share)
      3. Cold – (cloud store/archive)
   2. ILM
      1. TFS supports file profiles which specifies:
         1. Aging and purging of file cache
         2. Security/encryption requirements
         3. Policy based tiering preferences
6. **Offsite mirroring and/or caching of data shares**
   1. **Mirror data from CIFS or NFS shares to multiple site locations for high speed data sharing and redundancy.**
   2. **Cache data from primary site to satellite office locations for sharing of data w/o having to purchase additional storage systems.**
7. **Accelerate VM provisioning**
   1. **Instantly create a clone of NFS share or iSCSI LUN**
8. **Performance**
   1. **Data cached on high speed (SSD) storage for fast IO of frequently accessed data.**
   2. **Data segmented into chunks for efficient processing.**