SRM Storage and File Types

Jean-Philippe Baud & James Casey

April 26, 2006

Overview

This document describes core type definitions as defined in SRM v3 and defines more concretely the difference between get and put operations within the specification with regards to file lifetimes.

It describes the subset of values for these types that will be used by LCG tools, and proposes a mapping onto the "storage classes" that LCG experiments require.

It finally describes two new type definitions for usage with the SRM specification: **Storage Class** and **Cache Attributes**.

Description of SRM Type Definitions

File Storage Types

As of SRM v2.1, there are three file storage types which represent the lifetime of a file within the storage system and have the following semantics

- **Volatile**: This has an expiration time and the storage may delete all traces
- **Durable:** This has an expiration time, but the storage may not delete the file, and should raise an error condition instead.
- **Permanent**: This has no expiration time.

NOTE: A "file" in this sense equates directly with a SURL or a namespace entry within the storage system. It may have many different internal "copies" within the system (e.g. on tape, in disk pools, …) , but deleting the file means deleting it from the storage namespace.

Retention Policies

Quality of Retention is a kind of Quality of Service. It acts as a notification to the storage system to the users intentions about the "preciousness" of the file, and what would/could be done if the storage system lost the data.
There are three retention policies with the following semantics:

- **Replica**: Replica quality is appropriate for data that can be replaced since other copies can be accessed in a timely fashion.
- **Output**: Output quality is an intermediate level and is used for data which can be replaced by length or effort-full processes.
- **Custodial**: Custodial quality indicates that the data is in some sense unique, and all attempts should be made to ensure it is kept.

Access Latency Mode

This refers to how latency to access a file is improvable. Latency is changed by storage systems replicating a file internally on a storage area of different access latency.
Three access latency modes are defined, but in practice only the first two are used:

- **ONLINE**: This is the mode with the lowest latency, implying that there exists a copy of the file in the cache and available for the client
- **NEARLINE**: This represents file stored on a high latency medium, such as tape, which can have their latency automatically improved to ONLINE by a staging operation
- **OFFLINE**: A file in OFFLINE requires human intervention to achieve low latency.

File Storage Type semantics on Get and Put operations

In the SRM specification, **PrepareToPut** and **PrepareToGet** are seen as symmetric operations, dealing with storing and retrieving SURLs from a storage system, and attributes in the operations are relevant to this SURL.

We believe that this leads to confusion, and there is an inherent asymmetry in the two operations:

- **PrepareToPut** is concerned with files within the storage system, represented by SURLs. A file has a file storage type associated with it.
- **PrepareToGet** is concerned with the cache copy associated with a particular SURL, in that we are asking for the storage system to provide us with a disk-resident cache copy of a file with some associated properties in terms of its accessibility.

In particular we believe that the requested lifetime and associated File Storage Type does not refer to the same thing; On a **PrepareToPut** operation, the storage type and lifetime apply to the SURL (or file within the storage system) and on a **PrepareToGet** operation the storage type and lifetime apply to the requested cache copy of the file that is returned as a TURL.

So in a **PrepareToGet** request we could ask for a File Storage Type in terms of it being Permanent, Durable or Volatile. During the **PrepareToGet** operation what we are really concerned with is the File Storage Type for the cache copy, not the file type itself.

LCG Usage of SRM File Storage Types

The LHC experiments have expressed their wish to only use permanent files in storage systems – i.e. there should never be automatic deletion of files from the storage namespace. This means that all **PrepareToPut** operations will only use the **File Storage Type** parameter with value “Permanent”. **PrepareToGet** operations will use either “Permanent” or “Volatile” depending on whether the cache copy is required to be garbage collected by the storage system or not.

This is even true for "scratch" files, since the experiments will do their own bookkeeping, and clean up the files afterwards. There are two methods which are supplied to remove the files:

- **srmRm**: This is a hard delete of the SURL, which will remove it from the namespace
- **srmRemoveFiles**: This will remove a file if there are no other current users of the file, otherwise it returns an error. It can be seen as a softer version of **srmRm.**

Proposed additions to the SRM Specification

Storage Classes

Currently the SRMv3 PrepareToPut does not allow any way for a given File Storage Class to specific Retention Quality. One possibility would be to require an extra argument of type **EnumRetentionQualityMode** (which already exists in the specification) which would specify the user requested retention quality.

Unfortunately, this does not seem to map well to LCG actual usage. We would suggest using a different concept instead, **Storage Class**, which maps directly into the minimum number of copies of a file that should be stored on Disk and Tape:

Storage Class	*Minimum Required Copies*	
	Tape	Disk
Tape1Disk0	1	0
Tape1Disk1	1	1
Tape0Disk1	0	1
Tape0DiskN	0	>1
TapeNDisk0	>1	0
TapeNDisk1	>1	1
TapeNDiskN	>1	>1
…	…	…

One important principle when asking for a cache copy is that we do not "promote" a file in terms of its file storage type, e.g. For a volatile file we cannot ask for a permanent copy, since that "promotes" its lifetime to infinite from finite. To do this, a new operation would be required, e.g. **ChangeStorageClass**, which perhaps could be limited to only VO administrators to stop normal users changing the type of files.

Cache Attributes

Also, we would like to be able to specify extra properties for the TURL. Some of these are currently inferred by storage systems, e.g. the uid/DN/VO of the requesting user defines a VO-specific pool that they will be mapped into, the IP address of the requesting host can be used to put a file either on a pool for local access or for WAN access. Also some are currently in the SRM protocol, e.g. access Protocol and file lifetime.

We list below some TURL properties as a starting for discussion:

- Access Pattern (random/sequential)
- Access Speed (high/ low)
- Access Protocol (gridftp, rfio, dcap, …)

Also there are some properties then inherent to the node the file resides on that can affect the quality of service of access to the TURL.

- File System parameters (block size,…)
- TCP Transfer Protocol Properties (for WAN/LAN transfers)

We believe there should be a new parameter on the **PrepareToGet** and **PrepareToPut** operations that specific the attributes that are required for the TURL returned to get and put a file respectively.