Hi, all,

We (Alex, Junmin, and I) have spent many hours analyzing all the comments we got on the SRM.v2.1.Rev2 spec. I'll try to address all the issues and implications, as well as changes made in this revision (SRM.v2.1.Rev3). If you have any objections to these changes, please let us know by Thurs. Feb 27th. After this date we'll freeze this version if there are no further objections.

The new spec SRM.v2.1.Rev3 is enclosed. We left the changes in, so you can see what was done. Click on "highlight changes on screen" to see what was done.

I know that we are not going to resolve all outstanding issues without perhaps another all-hands meeting. So, if you are "reasonably" happy with this version, please say so. I'd like to hear from anybody who has put time to read this carefully, especially if you are OK with this version.

Arie

1) A single directory for all file types.

This is the most significant change. The people from Jlab (Chip, Andy, Bryan) suggested to have a single directory for all the files in the SRM per user. Any file types can be put in that directory. The main advantage is that it simplifies all the directory methods, in that the client does not have to specify space type.

Notes:

- This has no effect on space reservations; the client can still reserve and negotiate space as before. When a file of a certain type is brought into the directory, the space use is counted against the total space of that type.
- The same directory can now hold files of different types.
- Types of files can be changed without changing the directory or copying a file to a different space.

Recommendation:

- We agree with this suggestion, and made the changes in the spec.
- If you disagree, speak up now!

The following are items related to the above choice.

1.1) srmLs with fileType

srmLs can still be requested for files of a certain type. For example, a client may want to see his/her "durable files".

Recommendation:

- If an srmLs is requested, only the files of that type and the corresponding directories (if any) will be shown. If file type is not specified, all file types will be listed.

1.2) Assignment of files to spaces

The above suggests that the total space that files of a certain type occupy is counted against the space of that type that the client has. However, we did not feel that it is necessary to specify that.

Recommendation

- It is up to each SRM implementation how to allocate file types to space types. For example, one SRM may choose to put volatile files in durable space, and another may not. Furthermore, an SRM can choose to have a single physical device, but partition it dynamically to space types.
- Important: no file-lifetime should exceed the space-lifetime it is put into.

1.3) Should we have space types?
Given that SRMs can assign files to any space they choose to, we considered the possibility of having only one global space reservation without space types. We concluded that this information may be quite important to some SRMs. Also, regardless of how file assignment to spaces is done and how space management is implemented (e.g. "best effort" vs. absolute), space reservation by type is useful information on the intentions of the client.
Recommendation
- keep reservation of space by type.

2) Current directory
The concept of "current directory" is associated with a session. Since we do not have sessions in SRMs because of non-blocking calls, it was suggested not to support that.
Recommendation
- We agree, and therefore this concept was removed from the spec.
- Specifically, srmCd and srmPwd were removed.
- Implication: full path must always be provided.

3) srmLs – support any number of levels
It was suggested to provide a numeric flag to specify the number of levels.
Recommendation
- We agree. Was "numOfLevels" was added to spec.
- The flag "allLevelRecursive" was left in.
- If both are specified "numOfLevels" takes precedence.

4) Add the ability to assign file type with srmCp
When a client wants to perform srmCp of a directory, that directory may have multiple file types. The Client may not have space on the corresponding file types. Thus, we added a parameter to srmCp to designate the target file types.
Once copied, the client can changeFileType as desired.
Recommnedation
- Add "fileTypeToBeAssigned" to srmCp
- Note: this is not needed for srmCopy since the source file type is not known, and we have a file type designation for each file in the TCopyFileRequest array.

5) Have all types as XML-compliant types
Recommendation
- Agreed. What added to the spec in introduction, under "definition of terms".

6) Where to find WSDL file and SOAP endpoint

The SOAP endpoint can be specified as part of the WSDL, since WSDL is extensible to allow the description of endpoints. One suggestion was to make it part of the SURL. However, it is better to have this part of the WSDL, since it permits the SOAP endpoint to change. The question is how to get the WSDL for a particular SRM. In the future, there will be some kind of Web Service Discovery services, but for now we need to find a standard solution. It was suggested that the location for the WSDL could be constructed from the SURL.
Recommendation

- The general standard for the location of the WSDL is of the form: https://host:port/srm/version/srm.wsdl, where host and port is different for each SRM. The host:port is taken from the SURL.
- The following is the standard for the current version: https://host:port/srm/2.1/srm.wsdl, where only host:port is different for each SRM. For example: if the SURL is: srm://dm.lbl.gov:4003/myspace/myfile1, then the WSDL for the SRM is on https://dm.lbl.gov:4003/srm/2.1/srm.wsdl.
- In the WSDL file, one can have the SRM SOAP endpoint specified into a different host:port. For example, the soap endpoint in the WSDL file could be https://dataportal.lbl.gov:5000.
- Given the SURL above, When the client connects to the SRM at https://dataportal.lbl.gov:5000, it will pass the string /myspace/myfile1 to the SRM as an argument for the methods that need it.

7) Overwrite feature
It was suggested to have the feature that when srmCopy is performed an Overwrite flag is provided to avoid overwriting file that was already copies.
Recommendation

- We agreed. We thought that 3 cases are useful: Never, Always, WhenFileSizeDoesNotMatch, and added that to the spec.

Other items mentioned in emails

The following are issues that were mentioned in emails but were not acted on. If you think they should be acted on, then let all know.

1) Space IDs
It was suggested to introduce space IDs. We discussed this in CERN, and concluded that it only complicates the methods. We saw no reason to have multiple spaces of the same type per user. If users need to organize files in some space type, they can do that with directories.

2) Duration for "reassign to user".
There was a question about "reassign to user" and why have a duration with it. The purpose of "reassign to user" is to allow a user to copy file from another user, but only for a limited time. After duration expires, the files can be removed as far as the owner is concerned. It is an "automatic garbage collection" type of service.
No changes made.

3) Having more types of spaces
The suggestion was made to have all kinds of attributes to spaces, like a space that puts files in two physical locations. We agree, but think that such attributes should be associated with the file, not the space. In any case, this is a major change that we could leave to SRM.v3.0. No changes made.

4) Error codes
One suggestion was to have only three error codes: success, try-again, failed. All other information should be provided in a string. We think this is OK for a user who get this feedback, but not sufficient for middleware software that accesses an SRM. For example, if the error is NOT_ENOUGH_SPACE, the calling program may want to try and allocate more space. "Failed" is not enough. Middleware programs can parse the string information, but information needs to be agreed in some format as well. We don't feel strongly about the error codes, except that all meaningful codes for a middleware program should be provided explicitly, not in strings. Suggestions are welcome.

5) Use of enum
The suggestion was made to avoid these, and use strings. How do we specify in the spec the choice of value we wish to permit and standardize on? No changes made.

6) The use of ACLs
It was suggested to avoid ACLs since it is not a standard. Nowhere in the spec was ACL mentioned. ACLs can be used as a choice of implementation to support permissions, but they are not visible externally to the client.

7) Support for "group" with directories
It was suggested to provide that. Since "group" setup is not supported dynamically by any file system, it is a capability that is generally not available to SRMs. We have the capability for the owner of a file to addPermission to another user. We believe this is powerful enough. No changes made.

8) Global File Names (GFNs)
GFNs are managed by other components, such as RLS. The suggestion was to remove it from SRM calls altogether. However, if GFNs are provided, then file sharing is facilitated. Thus, we removed it from srmChangeFileType as suggested, but left it as an optional parameter in TGetFileRequest, TPutFileRequest, and TCopyFileRequest. It is optional, no harm done.
Note:

- In principle, SRMs could go to the RLS, do a reverse mapping SURL → GFN, have a forward mapping GFN → all SURLs, and search for a match. It is a lot easier to search if the GFN is provided. By letting the GFN to be provided in the method, the user can potentially get service improvements, which is an incentive to provide it.

9) Add (offset, count) to srmLs
This is for directories that have thousands of files, and the user may want to see some files down stream. We observed that the order of files could also be specified, like sort-by creation date, or sort-by size. We also observed that if we have an SRM on top of a file system, this will still take a long time to get the entire list out, and then return the desired subset. It is probably a better

practice to have sub-directories when thousands of files are involved. This suggestion was made by Chip. Any opinions? Should we add this?

10) DurationAssigned in TMetaDataPathDetail
There was a question about the usefulness of that. We think that it is useful for a client to know what the original lifetime was assigned, not only what is left. That can give them information on whether they are pre-fetching files too early. In any case, this is optional. We can remove that if other people think it is useless.

11) Some additional questions from JP Baud

- srmLs could probably report also the status of the file
  - What "status" do you have in mind?

- which command should be used to know if a given file has already a copy on disk on a specific SRM?
  - SrmCheckPermission will do that. For security reasons only an authorized user should be able to see files that he/she is authorize to see.

- What reply should give srmLs for a file not in the disk cache anymore but still in HSM?
  - We think it should be returned with a file type "permanent". If it is both in HSM and a "volatile" disk cache, it should be returned as both "permanent" and "volatile" (but returning only "volatile" is OK too, I think).

- Are the commands srmMkdir and srmLs limited to a DRM or do they also apply for an HRM?
  - both. No reason to limit that. In the case of HRM the TstorageSystemInfo may be needed, where with DRM it may be controlling the space, and the TstorageSystemInfo may not be needed.

- Is the owner reported by srmLs the original owner of the permanent file or the owner of the copy of the file possibly in the volatile space?
  - No. This is too much to keep track of. Like unix, the owner of the file is the user who copied it.