Table of Contents

Issues in the SRM v2.2 Specifications (Sept. 2006 Edition)

In what follows we list the issues found in the SRM v2.2 specifications that are published here.

1. **StatusOfPutRequest** should return SRM_ABORTED after a successful **AbortRequest** has been executed. That is the case for **srmStatusOfGetRequest**.
    - Fixed in the new spec
    - Condition checked in S2 test suite, ***basic*** tests
2. SRM_DONE is not used and should be removed from the WSDL
    - **It won't be fixed in 2.2**
3. **srmAbortFiles** returns SRM_ABORTED at file level in some implementation while only SUCCESS, INVALID_PATH and FAILURE are possible. What is the right behavior?
    - At file level the method should return SUCCESS and not ABORTED. It will be better specified in the spec
    - Condition checked in S2 test suite, ***basic*** tests
4. The Overwrite Mode WHEN_FILES_ARE_DIFFERENT is not supported by any implementation. What is the behavior foreseen ?
    - This funcionality should not be implemented at the moment. Files can be different when the declared size for a SURL differs from the actual one
5. **Asynchronous** can return REQUEST_INPROGRESS. Here is the list of asynchronous calls that do not follow this rule in the spec:
    1. **srmLs** (request level)
    - The spec will be fixed for v2.2.
    - Condition checked in S2 test suite, ***basic*** tests
6. Some Status methods for asynchronous SRM functions should return REQUEST_INPROGRESS:
    1. **srmStatusOfLsRequest** (both at request and file level)
    2. **srmReserveSpace** (at request level)
    - The spec will be fixed for v2.2
    - Condition checked in S2 test suite, ***basic*** tests
7. In the spec there is reference to “volatile/durable/permanent” space, even though those attributes are never defined. See for instance **srmReleaseSpace**
    - 2.3.2.b will be changed to OUTPUT or CUSTODIAL retention quality space, instead of durable or permanent space.
8. No methods allow getting the associated description (if it exists), given a space or request token. No methods allow for the retrieval of all space token descriptions for a given VO or FQAN. This requires a change in the WSDL.
    - **It won't be fixed in v2.2**
9. The spec does not specify if Token Descriptions are case sensitive or not. Different implementations behave differently.
    - The spec will be changed for v2.2 to state that Token Descriptions are case sensitive and immutable.
    - Condition checked in S2 test suite, ***usecase*** tests
10. The SRM method **srmReleaseFiles** is used to reset the TURL lifetime to 0. Therefore it can be performed on files resulting from **srmPrepareToGet**, **srmBringOnline**, **srmPrepareToPut** and **srmPutDone**. **srmReleaseFiles** is not allowed after an **srmCopy** since there is not directly exposed TURLS associated with the file (this means that files are released after an **srmPutDone** in a **srmCopy** operation in order to avoid that for the implementations honoring pins the file stays around in some space with some lifetime ?)
    - **srmReleaseFiles** is only valid after a **PrepareToGet** or a **BringOnline** operation. To release a TURL after a **PrepareToPut**, **AbortFiles/AbortRequest** should be used. If a user issues **srmReleaseFiles** after an **srmPrepareToPut** or **srmPutDone** then the system returns SRM_INVALID_REQUEST.

- Condition checked in S2 test suite, ***usecase*** tests

11. If the pin on a TURL is still not expired and the overwrite option was specified in the method that created the file (either **srmPrepareToPut** or **srmCopy**), then users can overwrite the files (which TURLs would they use since the TURL resulting from an **srmPrepareToPut** or **srmCopy** is not available/not valid ?). Once open sessions are completed, all other (disk) copies will be marked as invalid. The overwrite option does not imply a removal of the SURL. The file will stay in the namespace but the content will change.
    - Still under discussion
    - Proposal: The overwrite mode is considered more powerful over pin. It is allowed to mark an valid TURL invalid when the owner of the surl issues an overwrite command, just like it is OK to call srmRm() and cause the valid TURLs be erased.
12. In **srmReleaseFiles**, if the request token is not provided, then **authorizationID** is needed. It is not clear what this field is for. It looks like both SURL and the request token can be left unspecified.
    - When request token is provided and no SURLs are provided, all files that belong to the request (that is associated with the request token) will be released. When request token is not provided andvSURLs are provided, those SURLs that belong to the caller will be released. But in this case, some verification is needed that those SURLs belong to the caller, and that can be by the authorizationID. There cannot be the case that none of SURLs and request token are provided, and At least one of requestToken and SURLs must be provided. AuthorizationID is an additional information that can verify the caller, and it is useful specially when everyone is mapped into one login in the grid-mapfile.**This info will be added to the spec for v2.2**.
13. Allow for NO_USER_SPACE (if the user provided a space token) and NO_FREE_SPACE (if the user has not provided a space token) at file level in **srmPrepareToPut*/*srmStatusOfPrepareToPutRequest**. In this case, if some file succeeded, the SRM_PARTIAL_SUCCESS should be returned at the request level.
    - Still under discussion.
14. Allow for SRM_FILE_BUSY and SRM_FILE_LIFETIME_EXPIRED (for durable SURLs) at file level status in an **srmLs**. Remove SRM_FILE_IN_CACHE. If file is not in cache, the correct error code to return is SRM_INVALID_PATH. SRM_FILE_LIFETIME_EXPIRED is a valid return code in the srmCopy operation.
    - Still under discussion.
15. **srmExtendFileLifeTime** behaviour:
    1. the method allows to change only one lifetime at a time (either surl lifetime or pin lifetime), depending on the presence or absence of the request token.
    2. when SURL lifetime is extended with **newFileLifetime**, the request token must not be specified.
    3. when the request token is specified, pin lifetime is extended.
    4. when lifetime input parameters are not specified, the server applies the default value. (-1 and 0 lifetime have the same interpretation as before; indefinite and default)
    - This will be added to the spec for v2.2.
16. **srmExtendFileLifeTime[InSpace]** has an ambiguity and allows for the following 2 possibilities in case the **newTimeExtended** exceeds the remaining lifetime of the space:
    1. Return PARTIAL_SUCCESS at the request level and SRM_FAILURE at the file level and the **TSURLLifetimeReturnStatus** returns the remaining lifetime.
    2. SUCCESS at the request and file level and **TSURLLifetimeReturnStatus** contains the remaining lifetime. Case b is the correct one and should be specified in the spec.
    - This will be added to the spec for v2.2.
17. On page 52 point h) and on page 58 point i) of the spec the method **srmGetRequestID** is mentioned instead of **srmGetRequestTokens**.
    - The spec will be fixed for v2.2.
18. In **srmUpdateSpace** the output parameter **lifetimeGranted** is the lifetime left **relative** to the calling time.
    - The spec will be fixed for v2.2.

19. **srmGetSpaceMetaData** returns 0 if there is no space left in the allocated space. The space has the max. size and users have to get warned if exceeding file size (more than asked) happens, and they have to accommodate their files accordingly up to the max space size, or sort out their files in the spaces or update the space before running out. SRM_EXCEED_ALLOCATION should be introduced as a possible return code in **srmGetSpaceMetaData** at request and file level.
    - ♦ Still under discussion.
20. It must be possible to list the TOP directory. We can impose that Directory SURLs must end with a "/", if this makes the implementation easy.
    - ♦ Still under discussion.
21. Streaming mode is allowed on **srmPrepareToGet**, **srmPrepareToPut**, **srmCopy** and **BringOnline** operations. If streaming mode is supported, in case there is no enough space, the server returns SRM_REQUEST_QUEUED and keeps trying for the duration of **desiredTotalRequestTime** specified by the user. In the explanation field for the calls, the server can make explicit the fact that a retry is ongoing. If instead a server does not support streaming mode, than at file level the errors SRM_NO_USER_SPACE (if the user has specified a space token on **srmPrepareToPut**) or SRM_NO_FREE_SPACE can be returned and the request return code can be either SRM_PARTIAL_SUCCESS (if some files were successful) or SRM_FAILURE.
    - ♦ This will be made explicit in the spec.
22. Introduce estimatedWaitTime = -1 to mean unknown value.
    - ♦ This will be made explicit in the spec.
23. The space token is mandatory in srmExtendFileLifeTimeInSpace. Paragraph 2.9.2a needs to be fixed.
    - ♦ This will be fixed in the spec.
24. After an **srmPrepareToPut** a client is not forced to call **srmPutDone** nor an **srmAbortRequest** in order to set to 0 the pintime of the TURL. Therefore the pintime of the TURL passed as an input argument to the call is used to make the TURL expire in case the client does not issue nor **srmPutDone** nor **srmAbortRequest**.
    - ♦ This will be made clear in the spec.
25. After an **srmPrepareToPut** the SURL exists but it is marked as SRM_FILE_BUSY in an **srmLs** operation. When a SURL is in this status, its lifetime cannot be extended.
    - ♦ This will be made clear in the spec.
26. In the method **srmExtendFileLifeTimeInSpace** arrayOfSURLs is optional. This means the new lifetime must be applied to all files in the space if **arrayOfSURLs** is NULL.
    - ♦ This will be stressed in the spec.
27. In the method **srmExtendFileLifeTimeInSpace** the pinLifeTime field in the returned **TSURLLifetimeReturnStatus** will be left NULL, since the method only applies to SURLs.
    - ♦ This will be stressed in the spec.
28. **srmExtendFileLifeTime** can be used to extend the Lifetime of a SURL (the result of **srmCopy** or **srmPutDone** operation). It can be used to extend the pinLifetime of valid TURLs (result of **srmPrepareToPut**, **srmPrepareToGet**). It cannot be used to extend the pinLifeTile after an **srmCopy** or **srmPutDone** operation. In this case the method should return SRM_INVALID_REQUEST at the file level and SRM_PARTIAL_SUCCESS or SRM_FAILURE at the request level. Furthermore, Lifetime for SURLs resulting from an **srmPrepareToPut** can be extended through this method.
    - ♦ This will be made explicit in the spec.
    - ♦ Condition checked in S2 test suite, usecase tests.
29. For SURLs on which an **srmPrepareToPut** is being executed, an **srmLs**, **srmBringOnline**, **srmPrepareToGet** will return SRM_FILE_BUSY, a second **srmPrepareToPut** or **srmCopy** will return SRM_FILE_BUSY if the SURL can be overwritten, otherwise they return SRM_DUPLICATION_ERROR.
    - ♦ This will be made explicit in the spec.
    - ♦ Condition checked in S2 test suite, usecase tests.
30. Lifetime of SURLs only start when **srmPutDone** is executed.
    - ♦ This will be made explicit in the spec.
31. On page 16 of the spec, 1.28 second bullet it is stated that the data structure **TGetFileRequest**

contains the **TAccessPattern**. This is not the case. The statement should be replaced with something that states that **TTransferParameters** can be specified during an **srmPrepareToGet/srmPrepareToPut/BringOnline** invocation. What specified in such data structure might collide with the characteristics of the space specified and in this case **TTransferParameters** must be ignored.
    - ♦ This will be fixed in the spec.
32. **srmSetPermission** and **srmRm** can be invoked even when the status of the surl is **SRM_FILE_BUSY**.
    - ♦ This will be made explicit in the spec.
33. **srmMv** cannot be invoked when the status of the surl is **SRM_FILE_BUSY**. The return code must be **SRM_INVALID_REQUEST**.
    - ♦ This will be made explicit in the spec.
34. **srmChangeSpaceForFiles** cannot be invoked when the status of the SURL is **SRM_FILE_BUSY**. In this case the return code at file level must be **SRM_INVALID_REQUEST**
    - ♦ This will be made explicit in the spec.
35. Specify what values can **lifetimeLeft** and **lifetimeAssigned** can assume in **TMetaDataPathDetail** and in **TMetaDataSpace** after the execution of an Ls. Are 0 and -1 allowed values ? What is their meaning ? Same for **remainingPinLifetime** and **remainingFileLifetime** in **TPutRequestFileStatus**, **TCopyRequestFileStatus**, **TSURLLifetimeReturnStatus** and everytime the Lifetime appears as an output paramenter: **srmReserveSpace**, **srmUpdateSpace**, **srmStatusOfUpdateSpaceRequest**, **srmStatusOfReserveSpaceRequest**, **srmPrepareToGet**, **srmStatusOfPrepareToGetRequest**
    - ♦ This will be made explicit in the spec.
36. What is the lifetime assumed in **srmUpdateSpace** if the newLifetime is unspecified ? The default or infinite ? **srmExtendFileLifetimeInSpace** unspecify newLifetime means default, for **srmReserveSpace** it means infinite.
    - ♦ Under discussion
    - ♦ It has been proposed the following: if lifetime is not given or zero, the default lifetime used is "Infinite" if the request comes from the Storage Element admin or "one day" otherwise. The behaviour should be the same for **srmReserveSpace**, **srmUpdateSpace**. The behaviour in case of **srmExtendFileLifetimeInSpace** should be as of now.
    - ♦ All lifetimes should be assumed to be the default if left unspecified.
37. In **srmReserveSpace** the desiredLifeTime is an unsigned long, while in all other SRM methods it is an int. Is this correct ?
    - ♦ It should be int. It will be fixed in the spec.
38. What are the possible values for **remainingTotalRequestTime** and **remainingDeferredStartTime** in **srmBringOnline** ? They are declared as int. Are negative values possible ?
    - ♦ Waiting for an answer.
39. In the output of the **srmLs** for a file when **fullDetailedList** is true, only the fields **path**, **size**, **userPermission**, **lastModificationTime**, **file type** and **lifetimeLeft** must be returned. Should **fileStorageType**, **retentionPolicyInfo** and **fileLocality** be returned as well?
    - ♦ Waiting for an answer.
40. The copy from LBNL to FNAL in push mode fails because the FNAL server returns SRM_FILE_IN_CACHE, which is not a valid Copy status at the file level. When overwriting is not set and the SURL exists at the target, the target SRM may say duplication error (if lifetime is still valid) or file_in_cache (if lifetime is expired). If lifetime is expired at the target there is a conflict since **srmCopy** says SRM_LIFETIME_EXPIRED and **srmPrepareToPut** says SRM_FILE_IN_CACHE at the file level. What should be the correct behaviour ?
    - ♦ Waiting for an answer.
41. If **srmReleaseFiles** is called after an **srmPrepareToPut** or an **srmPutDone** request, the call fails with SRM_INVALID_REQUEST.
    - ♦ It is already clear in the spec. No action
42. What are the possible values for **remainingTotalRequestTime** and **remainingDeferredStartTime** in **BringOnline** ? They are declared as int. Are negative values possible ?
    - ♦ Waiting for an answer.

- Proposal from Alex:"Negative time in our spec means "indefinite" time, and we cannot have remainingDeferredStartTime as negative. **remainingTotalRequestTime** may be. If request will be tried, by default, until all files in the request will be completed, the server may return negative since there is no 'expiration time' of the request... ? "

43. What is the actual meaning of the field lifetimeAssigned present in the output of an srmLs or srmGetSpaceMetaData request?
    - Waiting for an answer.
    - Proposal from Alex:"It was intended to hold the original lifetime that was assigned upon the request. [lifetimeAssigned - lifetimeLeft] = time passed so far since the assignment. The question comes then when lifetime got extended and what to assign on this lifetimeAssigned...? How about we define this lifetimeAssigned as the full amount of lifetime on the file before decreased by the passing time? Then, it can hold all lifetime extention."
44. Since the recursive **srmLs** has the major problem that the number of files returned is limited, it is proposed not to support it for the moment.
    - Under discussion.
45. A recursive Rmdir cannot remove the files contained in the subdirectories, but it can only remove subdirectories. Is this useful ? Do we have a use case for it ? There is a proposal to drop support for recursive Rmdir.
    - Under discussion.
46. If a SURL is being written (by a put or copy operation) and in the mean time an **srmRm** gets executed successfully, the subsequent **srmPutDone** will return SRM_INVALID_PATH at the file level.
    - This will be made clear in the spec
47. If 2 **srmPutDone** are executed for the same request for the same SURL, the second gets SRM_DUPLICATION_ERROR at the file level and SRM_SUCCESS or SRM_PARTIAL_SUCCESS or SRM_FAILURE at the request level.
    - This will be made clear in the spec
48. **srmReleaseSpace** is allowed on a space where a SURL is being created. If file is volatile and **forceFileRelease** has been set to true, the file is removed and SRM_INVALID_PATH returned by the **srmPutDone** at file level. If the file is permanent, the file is moved in the default space and the space is successfully released (if no other pinned files are there). The subsequent **srmPutDone** is successful. If **forceFileRelease** is set to false, the ReleaseSpace fails (SRM_FAILURE).
    - This will be made clear in the spec
49. After a put cycle, in the space there is a copy of the file even if the handle (TURL) is expired/gone. What is its lifetime ? This has implications on **srmReleaseSpace** of that space.
    - Under discussion
    - Proposal by Jean-Philippe: a copy of the file stays in space and its lifetime equals the SURL lifetime.
50. If 2 consecutive **srmPutDone** are issued on the same SURL, for that SURL the second **srmPutDone** will get an SRM_FAILURE at file level, while at request level either SRM_PARTIAL_SUCCESS or SRM_FAILURE must be returned.
    - This will be made clear in the spec.

-- Flavia Donno - 18 December 2006

This topic: SRMDev > IssuesInTheSpecificationsV22SeptemberEdition
Topic revision: r1 - 2006-12-19 - FlaviaDonno