7 DATA STORAGE REQUIREMENTS

The use of files and folders on-board introduces an additional layer of abstraction for on-board data management. This layer provides better visibility on the data stored on-board by organizing them through directories, structured files with attributes providing additional information (e.g. date of creation, last date of modification, protection access, data type, etc.).

The advantages are multiple: it improves on-board autonomy by allowing development of smarter algorithms to autonomously handle communication sessions (by downloading files according pre-defined criterions: by date, type, …), efficiency of on-board search algorithms by sorting data in relevant structures and ground operators visibility on stored data by allowing a visualization of data closer to current PC operating systems which allow operators to download first the most essential files.

The section provides generic requirements that shall be made applicable to missions after a proper tailoring to match the needs of that missions, e.g. in term of tolerance to radiations, PUS required services, etc.

In that section, the concept of Mass Memory System (MMS) is used in order to avoid overlapping with the concept of Data Storage System that is the subject of [SRD]. The definition of Mass Memory System is that a Data Storage System is made of one or several Mass Memory Systems.

7.1 Generic requirements

This chapter aims at describing the generic requirements applicable to the Data Storage System.

SAVOIR.MMS.GEN.0100

Storage media

The Mass Memory System shall provide one to several storage media in order to support the storage of data received through its physical interface(s).

Rationale: *The Mass Memory is composed of storage media that are in charge of storing the data.*

Verification Method: RoD, T

Parent: SAVOIR-DSS-GEN-010, SAVOIR-DSS-GEN-020

SAVOIR.MMS.GEN.0110

Volatility

A storage media shall be made either of volatile or non-volatile memory.

Rationale: *Both types of memories are used in space systems.*

Comment: *The choice depends on mission requirements and criticality of data to be stored. For example the Mission Time Line (MTL) of a payload could be stored in volatile memory (meaning that its loss is tolerated), while non-volatile memory shall be used for platform MTL involved in critical phases of the mission, for which data shall be persistent for autonomy in operations*

Verification Method: RoD, T

Parent: SAVOIR-DSS-GEN-020

SAVOIR.MMS.GEN.0120

Data retention time

The MMS shall provide data retention for at least <MM Retention Time> days from hardware unit manufacturing to <Satellite Launch Time> + <Planned Mission Time> for all storage media with non-volatile memory.

Rationale: *The data stored in non-volatile memory shall be retained for the complete mission time including manufacturing.*

Verification Method: RoD, T

Parent: SAVOIR-DSS-GEN-030

SAVOIR.MMS.GEN.0130

Capacity

The capacity of storage media shall be expressed at EOL.

Rationale: *The capacity required by the mission is a main driver for the design of the Data Storage System, and thus MM. It has to be sized to store all missions data, e.g. Science data for <max-#-hours-science> hours, Platform housekeeping for <max-#-hours-hk> hours, On-board Control Procedures, timelines, Software images and patches, etc.*

Verification Method: RoD

Parent: SAVOIR-DSS-GEN-040

SAVOIR.MMS.GEN.0140

Preservation of non-volatile memory

The content of the non-volatile memory of storage media shall be maintained even in case of a power cycle of the spacecraft or the storage media(s).

Rationale: *The data stored in non-volatile memory shall be kept under all circumstances.*

Verification Method: RoD, T

Parent: SAVOIR-DSS-GEN-050

SAVOIR.MMS.GEN.0150

Preservation of volatile memory

The content of the volatile memory of storage media shall be maintained except in case of a power cycle of the spacecraft or the storage media(s).

Rationale: *Platform reconfiguration shall not cause the loss of any data stored in volatile memory. Only a power cycle of the spacecraft or the storage media.*

Comment: *Platform reconfiguration means switching from nominal to redundant equipment (for at least one of the platform equipment, e.g. Processing Module, Star Tracker, etc…).*

Verification Method: T

Parent: SAVOIR-DSS-GEN-052

SAVOIR.MMS.GEN.0160

Protection against radiation

The content of the volatile memory of storage media shall be protected against the radiations.

Rationale: *Immunity against Single Event Effects, including destructive ones (e.g. Single Event Latch-up (SEL), and SEU, SEFI, SET).*

Verification Method: T

Parent: SAVOIR-DSS-GEN-060

SAVOIR.MMS.GEN.0170

Configuration

The MMS shall support configuration changes of storage media.

Rationale: *The MMS shall be fully configurable in order to support mission constraints, e.g. for switching-off faulty memory module, switching-on redundant memory module to replace a faulty one or add capacity).*

Comment: *This requirement also supports the implementation of removable storage media and the integration of the storage area they contain.*

Verification Method: RoD, T

Parent: SAVOIR-DSS-GEN-070

SAVOIR.MMS.GEN.0180

Accessibility of data for reading

All the memory of MMS storage media shall be accessible for reading from Ground.

Rationale: *All the memory used for data storage and internal MMS operation (e.g. FS data structure information / configuration) shall be accessible from Ground for analysis purpose.*

Verification Method: RoD, T

Parent: SAVOIR-DSS-GEN-080

SAVOIR.MMS.GEN.0190

Accessibility of data for writing

All the writeable memory of MMS storage media shall be accessible for modification by Ground.

Rationale: *It shall be possible to modify any writeable part of memory.*

Verification Method: RoD, T

Parent: SAVOIR-DSS-GEN-090

SAVOIR.MMS.GEN.0200

Time

The MMS shall rely on the time provided by the On-Board Time for its time related operations.

Rationale: *To support operations related to the time (like storing the file creation time in attributes).*

Verification Method: RoD, T

Parent: SAVOIR-DSS-GEN-100

SAVOIR.MMS.GEN.0210

Time correlation

The MMS time base shall be correlated with the On-Board Time with an accuracy of <MM OBT allowed time difference>.

Rationale: *In case the MM is using or recreating its own time base, the difference with OBT can't go above a mission dependent threshold to ensure consistency.*

Comment: *The time types used shall be the same to avoid accuracy loss in conversion.*

Verification Method: RoD, T

Parent: SAVOIR-DSS-GEN-110

SAVOIR.MMS.GEN.0220

User Entity identification

The MMS shall support the unique identification of User Entities.

Rationale: *The MMS shall be able to manage access rights related to different User Entities.*

Comment: *Note that a User Entity is any user of the MMS.*

Verification Method: RoD, T

SAVOIR.MMS.GEN.0230

MMS Layered architecture

The MMS shall implement a layered architecture.

Rationale: *The layered architecture makes possible to efficiently cope with the different memory technologies, provides flexibility in data storage and provides standard interfaces.*

Verification Method: RoD

Parent: SAVOIR-DSS-GEN-100

7.2 Interface requirements

This section provides requirements related to the MM interfaces. These interfaces are mission dependent and each MM may implement one to several interfaces to achieve the data storage functions.

7.2.1 Command/Control and data acquisition interfaces

SAVOIR.MMS.IF.0100

Command & Control interface

The MMS shall implement at least one interface for Command & Control.

Rationale: The MM must at least provide an interface for command/control (e.g. from OBC or P/L computer).

Verification Method: RoD, T

SAVOIR.MMS.IF.0110

Command & Control interface directionality

Any MMS Command & Control interface shall support both RX and TX.

Rationale: To allow reception of telecommands and emission of telemetry.

Verification Method: RoD, T

SAVOIR.MMS.IF.0120

Direct data acquisition interface

The MMS shall implement zero or more interface for direct data acquisition.

Rationale: The MM shall at least be able to receive data (RX) through these interfaces, and may use them to forward commands (TX) depending on mission needs. These interfaces would typically be connected to instruments generating data science to be stored.

Verification Method: RoD, T

SAVOIR.MMS.IF.0130

Direct data acquisition interface directionality

Any MMS interface used for direct data acquisition shall support at least RX, and optionally TX.

Rationale: Data acquisition may be unidirectional from the data source to the MMS for storage.

Verification Method: RoD, T

SAVOIR.MMS.IF.0140

Direct data acquisition types

Each MMS input interface used for direct data acquisition (i.e. excluding C&C interface) shall be classified among three types, defining the data handling level (raw, data link protocol, packet) and implicitly the supported mapping(s):

- RAW – received data is handled as a data flow without interpretation,
- LINK_PROTOCOL – received data is handled at the level of entity exchanged on the link (i.e. SpaceWire packet, 1553 Frame, etc.). Received data flow is interpreted to identify the address, and only Protocol Address mapping is supported,
- PACKET – received data is handled at the level of CCSDS packet. Received data flow is interpreted and reassembled into Packets (potentially PUS packets). Packet APID and Custom Packet Field mappings are supported for both standard and PUS packets, whereas Packet Service and Packet Service & Subservice mappings are supported only for PUS packets.

Rationale: *Data acquisition may be unidirectional from the data source to the MMS for storage.*

Comment: *Only Exclusive mapping type is supported for the RAW type.*

Comment: *The acquisition from hardware interfaces shall be handled externally to the FMS which means that the scheduling and hardware synchronization aspects are managed apart from the FMS. As an example, for the MM input interfaces classified with PACKET type, the FMS shall only handle data reassembled into packets, whatever the lower-level protocol and interface used. For the input interfaces classified with LINK_PROTOCOL type, the FMS shall only handle reassembled protocol datagrams (1553 Frame data words, SpaceWire packet cargo) with their associated address.*

Verification Method: T

Parent: SAVOIR-DSS-ORG-520

SAVOIR.MMS.IF.0141

Direct data acquisition unmapped

Each MMS input interface used for direct data acquisition shall discard data that are not mapped to any of the DMS (File or Packet).

Rationale: *To protect the MMS against unexpected data.*

Verification Method: T

Parent: SAVOIR-DSS-ORG-540

7.2.2 PUS Services

SAVOIR.MMS.IF.0150

PUS service 1 - request verification

The MMS shall support PUS-C services 1 (request verification).

Rationale: *The PUS defines standard services to access on-board resources from Ground. The PUS Service 1 defines request verifications services.*

Verification Method: T

Parent: SAVOIR-DSS-IF-010

SAVOIR.MMS.IF.0160

PUS service 2 – device access

The MMS shall support PUS-C services 2 (device access).

Rationale: *The PUS defines standard services to access on-board resources from Ground. The PUS Service 2 defines device access services.*

Verification Method: T

Parent: SAVOIR-DSS-IF-020

SAVOIR.MMS.IF.0170

PUS service 3 – housekeeping

The MMS shall support PUS-C services 3 (housekeeping).

Rationale: *The PUS defines standard services to access on-board resources from Ground. The PUS Service 3 defines housekeeping services.*

Verification Method: T

Parent: SAVOIR-DSS-IF-030

SAVOIR.MMS.IF.0180

PUS service 5 – event reporting

The MMS shall support PUS-C services 5 (event reporting).

Rationale: *The PUS defines standard services to access on-board resources from Ground. The PUS Service 5 defines event reporting services.*

Verification Method: T

Parent: SAVOIR-DSS-IF-040

SAVOIR.MMS.IF.0190

PUS service 6 – memory management

The MMS shall support PUS-C services 6 (memory management).

Rationale: *The PUS defines standard services to access on-board resources from Ground. The PUS Service 6 defines on-board data load and data dump operations.*

Comment: *The PUS Service 6 can be used to access the memory of storage media as well as the memory of the unit managing the storage media.*

Verification Method: T

Parent: SAVOIR-DSS-IF-050

SAVOIR.MMS.IF.0200

PUS service 15 – on-board storage and retrieval

The MMS shall support PUS-C services 15 (on-board storage and retrieval).

Rationale: *The PUS defines standard services to access on-board resources from Ground. The PUS Service 15 defines on-board storage and retrieval services.*

Verification Method: T

Parent: SAVOIR-DSS-IF-060, SAVOIR-DSS-ORG-760

SAVOIR.MMS.IF.0210

PUS service 17 – test

The MMS shall support PUS-C services 17 (test).

Rationale: *The PUS defines standard services to access on-board resources from Ground. The PUS Service 17 defines the capability to activate test functions implemented on-board and to report the results of such tests.*

Verification Method: T

Parent: SAVOIR-DSS-IF-065

SAVOIR.MMS.IF.0220

PUS service 20 – parameter management

The MMS can support PUS-C services 20 (parameter management).

Rationale: *The PUS defines standard services to access on-board resources from Ground. The PUS Service 20 defines parameter management service that can be used to manage the parameters related to the Data Storage Services.*

Verification Method: T

Parent: SAVOIR-DSS-IF-080

SAVOIR.MMS.IF.0230

PUS service 23 - file management

The MMS shall support PUS-C services 23 (file management).

Rationale: *The PUS defines standard services to access on-board resources from Ground. The PUS Service 23 provides the capability to manage on-board file systems and files.*

Verification Method: T

Parent: SAVOIR-DSS-IF-070

SAVOIR.MMS.IF.0231

PUS specific service for MMS – MMS management

The MMS shall support a PUS-C specific service giving access to MMS services.

Rationale: *The PUS Service 23 only provides a limited number of file services. The specific PUS-C service provides an extensive access to all MMS and FMS operations.*

Comment: *Only the subset of MMS (including Block Access, Store and FMS Services) that is required for a mission needs to be implemented.*

Verification Method: T

Parent: SAVOIR-DSS-IF-070, SAVOIR-DSS-ORG-570

7.2.3 Protocols

SAVOIR.MMS.IF.0240

Remote File Management protocol

The MMS shall support generic Remote File Management Protocol for remote access to the services provided by the FMS interface.

Rationale: *All the services provided by the FMS shall be accessible via a protocol for remote control by the OBC or Ground. Some PUS-C services (as 23) may rely on the FMS interface to operate.*

Comment: *The File Management protocol can only be used with mass memories able to implement File Management Services.*

Comment: *The File Management protocol is introduced in section 6.2. It is part of the Command and Control and can only be exchanged through the Command and Control interface (with OBC).*

Comment: *The remote access will be granted to the relevant interfaces defined in section -*

Verification Method: RoD

Parent: SAVOIR-DSS-ORG-180

SAVOIR.MMS.IF.0250

Remote Block Management protocol

The MMS shall support generic Remote Block Management Protocol for remote access to the services provided by the FMS interface.

Rationale: *This protocol allows for remote operations by the OBC or Ground at BAU level. SAU can still be accessed through PUS Service 6.*

Comment: *The File Management protocol can only be used with mass memories able to implement File Management Services.*

Comment: *The remote access will be granted to the relevant interfaces defined in section 8.1.3.*

Verification Method: RoD

Parent: SAVOIR-DSS-ORG-190

7.2.4 Communication standards

SAVOIR.MMS.IF.0260

SpaceWire standard

When using SpaceWire links the MMS shall comply with ECSS-E-ST-50-12C SpaceWire - Links, nodes, routers and networks.

Rationale: SpaceWire interfaces shall be compliant to the standard.

Verification Method: RoD

Parent: SAVOIR-DSS-IF-090

SAVOIR.MMS.IF.0270

SpaceWire protocol identification standard

When using SpaceWire links the MMS shall comply with ECSS-E-ST-50-51C SpaceWire protocol identification.

Rationale: SpaceWire interfaces shall be compliant to the standard.

Verification Method: RoD

Parent: SAVOIR-DSS-IF-100

SAVOIR.MMS.IF.0280

SpaceWire RMAP standard

When using SpaceWire for remote memory access, the MMS shall comply with ECSS-E-ST-50-52C SpaceWire - Remote memory access protocol.

Rationale: SpaceWire interfaces shall be compliant to the standard.

Verification Method: RoD

Parent: SAVOIR-DSS-IF-110

SAVOIR.MMS.IF.0290

SpaceWire CPTP standard

When using SpaceWire for CCSDS packet transfer the MMS shall comply with ECSS-E-ST-50-53C SpaceWire - CCSDS packet transfer protocol.

Rationale: SpaceWire interfaces shall be compliant to the standard.

Verification Method: RoD

Parent: SAVOIR-DSS-IF-120

SAVOIR.MMS.IF.0300

MIL-STD-1553 standard

When using the Mil-STD-1553 data bus, the MMS shall comply with ECSS-E-ST-50-13C Interface and communication protocol for MIL-STD-1553B data bus on-board spacecraft.

Rationale: Mil-STD-1553 interfaces shall be compliant to the standard.

Verification Method: RoD

Parent: SAVOIR-DSS-IF-130

SAVOIR.MMS.IF.0310

CAN standard

When using the CAN data bus, the MM shall comply with ECSS-E-ST-50-15C.

Rationale: CAN interfaces shall be compliant to the standard.

Verification Method: RoD

Parent: SAVOIR-DSS-IF-140

7.3 Data Storage Organization

This section defines the requirements related to the organization of data within the MMS.

7.3.1 Store

From a mission point of view, Stores are aiming at organizing the Storage Media capacity into distinct logical storage areas. This may be required to clearly delimit available storage spaces. This can be used to split the overall storage area and implement access restriction and quotas to User Entities) as well to separate data per type or criticality. Moreover, the splitting into logical areas brings the capability to use different organizations for data storage. For example, one Store may be managed by a File System suitable for the storage of small data blocks, whereas another Store may be managed by another different File System dedicated to the storage of large data blocks into files, and a others Stores may be managed by a specific Data Management System to cope with any sort of custom data structure, e.g. generated by Instruments. Note that a Block Access System is required to access a Store in order to abstract data access and deal with specificities of the underlying memory technologies.

A Store aiming at giving an abstraction level by providing logical addressing can be spread over several storage media even based on different technologies. These technologies can have different speed, different size of SAU and even limitation in term of access (Read only or Read and write). These differences have to be managed by upper levels starting with the Block Access System.

SAVOIR.MMS.ORG.0100

Storage Media management

The MMS shall be able to manage one or more Storage Media.

Rationale: *At least one Storage Media is required and there can be several Storage Media.*

Comment: *Storage media can be local (same location as the entity using it) or remote (connected through a communication link).*

Verification Method: T

Parent: SAVOIR-DSS-ORG-010

SAVOIR.MMS.ORG.0110

Store management

The MMS shall support the organisation of the overall data storage area provided by the Storage Media into one or several Stores.

Rationale: *To allow segmenting a Storage Media for distinct usage per upper layers or aggregating memory located into distinct Storage Media into a single Store.*

Comment: *As per definition, a Store is a collection of one to several BAU located in one or several Storage Media.*

Comment: *Stores logically segregate the entire memory space available to upper layers (e.g. File Systems).*

Verification Method: T

Parent: SAVOIR-DSS-ORG-020

SAVOIR.MMS.ORG.0120

BAU management

Accesses to a Store at BAU level shall be managed by a Block Access System.

Rationale: *The Block Access System provides services manage BAUs within a Store. It abstracts the memory technology and physical organization by exposing a logical contiguous storage area to the upper layer (BAU logical addressing).*

Verification Method: T

Parent: SAVOIR-DSS-ORG-030

SAVOIR.MMS.ORG.0130

Block access to remote Stores

The BAU located in a remote Storage Media (i.e. not directly embedded within the MMS) shall be accessed through a Remote Block Access Protocol.

Rationale: *The MMS supports storage not only within local Storage Media.*

Comment: *The Remote Block Access Protocol allows a File System to communicate with the BAS of a Store located into a remote storage media.*

Verification Method: T

Parent: SAVOIR-DSS-ORG-190

SAVOIR.MMS.ORG.0140

Block Access Systems

The MMS shall support one to several Block Access Systems.

Rationale: *Specific Block Access System may be necessary to:*

- *optimize the use of the memory space by providing an adapted BAU to the upper layer;*
- *cope with different memory technologies, e.g. one BAS for managing Flash based memory devices and another one for managing RAM based memory devices.*

Verification Method: T

Parent: SAVOIR-DSS-ORG-040

SAVOIR.MMS.ORG.0150

Store characteristics

A Store shall be characterized by at least:

- a unique identifier among all the Stores,
- the block access unit (BAU) size,
- the total storage capacity (in number of BAU),
- the collection of BAU (its structure allowing to identify and navigate between the BAUs, over one or several storage media),
- the access capability (Read-only or R/W),
- the permission to access it at User Entity level.

Rationale: *Required for upper layers to correctly locate and address the Stores. This requirement implies having a single BAU size per Store.*

Comment: *The way the collection is organized is intentionally let flexible to fit the architectural and mission constraints.*

Comment: *Permission per User Entity required the capability to uniquely identify User Entities.*

Verification Method: T

Parent: SAVOIR-DSS-ORG-050

SAVOIR.MMS.ORG.0160

Data Management Systems

The MMS shall support one to several Data Management Systems to manage the content of the Stores.

Rationale: *To support any kind of data organization within Stores, the DMS being in charge of the organisation of the data stored within a Store.*

Comment: *The DMS can be implemented by File Management Systems (FMS), by Packet Management System (PMS) or any other management system organizing data to be stored down to the management of BAUs.*

Verification Method: RoD, T

Parent: SAVOIR-DSS-ORG-120

7.3.2 File Management System

SAVOIR.MMS.ORG.0170

File Management System

The MMS shall support one File Management Systems (FMS) providing high-level services for data storage into Stores.

Rationale: *The FMS is an instantiation of a Data Management System supporting the organisation of the data stored within a Store as files.*

Comment: *A FMS is a specialization of a Data Management System which abstracts the data organization by the mean of files. A FMS relies on File System(s) and provides unified services to access abstract entities known as files and directories.*

Comment: *A Store managed by the FMS is called a File Store.*

Verification Method: RoD, T

Parent: SAVOIR-DSS-ORG-130

SAVOIR.MMS.ORG.0180

FMS independence from storage technology

The FMS shall be independent from the low-level storage technology and internal File System organization.

Rationale: *The FMS shall not have to deal with the underlying memory characteristics and FS organization specificities. The BAS and File System layers handle this task so that a FMS provides a unified interface whatever the memory type and logical organization.*

Verification Method: RoD

Parent: SAVOIR-DSS-ORG-140

SAVOIR.MMS.ORG.0190

FMS Services for PUS

The FMS shall comply with the file access and file management services required by the on-board software for supporting implementations of PUS service 23.

Rationale: *The PUS service 23 is the standard operational service to access files and then will be an interface to the Ground to access on-board files.*

Comment: *Other PUS services could rely on FMS services to store and manage their data, e.g. services 12, 13, 15, 18 and 20.*

Verification Method: T

Parent: SAVOIR-DSS-ORG-150

SAVOIR.MMS.ORG.0200

FMS Services for CFDP

The FMS shall comply with the file access and file management services required by the on-board software for supporting implementations of CFDP.

Rationale: *CFDP is a standard protocol for transferring files between the spacecraft and the Ground. Its implementation shall rely on FMS Services.*

Verification Method: T

Parent: SAVOIR-DSS-ORG-150

SAVOIR.MMS.ORG.0210

FMS Store access

The FMS shall determine how to access a File Store from its identifier:

- Through the local File System associated to the Store,
- Through a Remote File Management Protocol in the case of a remote Store.

Rationale: *Local Store can be directly accessed through function calls while remote Store shall be addressed through a dedicated protocol.*

Comment: *The Remote File Management Protocol allows calling services of an FMS embedded into another storage device.*

Comment: *The use of the Remote File Management Protocol implies that the Remote Store is able to process File Management System requests.*

Verification Method: RoD, T

Parent: SAVOIR-DSS-ORG-180

7.3.3 File System

SAVOIR.MMS.ORG.0220

File System

The FMS shall support one or more File Systems to manage data storage within Stores, possibly implementing different logical organizations.

Rationale: *The FMS relies on File Systems to implement the management of the files.*

Comment: *The logical organization of a Store is implemented by the File System.*

Comment: *Limitations on files handling/management may directly come from the used File System(s).*

Comment: *If FMS only supports one File System, then FMS and File System can be merged, i.e. requirements of FMS apply to File System.*

Verification Method: RoD, T

Parent: SAVOIR-DSS-ORG-160

SAVOIR.MMS.ORG.0230

File System characteristics

The File System shall identify its characteristics including:
- data organisation scheme (e.g. contiguous, block, i-node),
- maximum number of entries (files and directories) per directory (root and subdirectories),
- maximum depth of the directory tree (0 = flat system),
- supported attributes per entry, e.g. "hidden" attribute, permission attributes,
- maximum number of attributes per entry,
- the capability to lock files,
- the type of checksums supported,
- referencing system (string, numerical value) and in case of string referencing system:
  - directory separator (e.g. "/" or "\"),
  - metacharacters (e.g. "*" and "?"),
  - maximum number of characters per entry identifier and syntax, e.g. 8.3, 128 characters.

Rationale: *File System characteristics and constraints shall be known by users.*

Comment: *The locking of a file is required by ECSS-E-ST-70-41C §6.23.4.3.*

Verification Method: RoD, T

Parent: SAVOIR-DSS-ORG-170

7.3.4 Directories

SAVOIR.MMS.ORG.0240

Directories

The FMS shall support organizing files into directories.

Rationale: *To allow either storing files without structuration (all files stored in root directory and organization supported by a naming convention) or gathering/sorting them into logical containers called directories.*

Comment: *Note that if FMS supports different File Systems, some may not have the capability to manage directories. In that case, related services will return an error when executed.*

Verification Method: RoD, T

Parent: SAVOIR-DSS-ORG-210

SAVOIR.MMS.ORG.0250

Directory depth

The FMS shall support a directory depth of <DIRECTORY_MAX_DEPTH>.

Rationale: *The depth refers to the number of subdirectories levels from the root, which may be 0 if no directory creation is foreseen, and thus not supported.*

Comment: *The maximum depth is dependent on the mission and is always greater or equal to zero. A depth of zero means that only root directory can be used and no other directory can be created (flat file system). A depth greater than zero means that a least one level of directory can be created (hierarchical file system).*

Verification Method: RoD, T

Parent: SAVOIR-DSS-ORG-210

SAVOIR.MMS.ORG.0260

Root directory

A Store root directory shall always exist in a File Store managed by a FMS.

Rationale: *This is the ancestor of any file and directory created into the Store, and the entry point to the File System.*

Comment: *It is accessed through the Store identifier attributed at FMS level. This requirement implies that the Store root directory cannot be deleted.*

Verification Method: RoD, T

Parent: SAVOIR-DSS-ORG-200

SAVOIR.MMS.ORG.0270

Directory identification

A directory shall be uniquely identified by its path, consisting in the gathering of:
- The identifier of the File Store in which it is contained,
- The tree of parent directories identifiers up to the Store root directory,
- Its own unique identifier within its parent directory.

Rationale: *A directory identifier is unique among all other entities stored within the parent (other directories and files).*

Comment: *The exact representation of the path (e.g. separator, representation and order of the identifiers) is implementation dependent and defined in the FMS characteristics (SAVOIR.MMS.ORG.0230)*

Verification Method: RoD, T

Parent: SAVOIR-DSS-ORG-220

7.3.5 Packet Management System

SAVOIR.MMS.ORG.0280

Packet Management System

The MMS shall support zero or more Packet Management System (PMS) providing high-level services for CCSDS Packet storage into Packet Stores.

Rationale: *Support of Packet Store may still be required in some missions.*

Comment: *A PMS is a specialization of a Data Management System which abstracts the data organization by the mean of Packet Stores.*

Verification Method: RoD, T

Parent: SAVOIR-DSS-ORG-740

SAVOIR.MMS.ORG.0290

PMS independence from storage technology

The PMS shall be independent from the low-level storage technology and logical Packets organization.

Rationale: *The PMS shall not have to deal with the underlying memory characteristics and FS organization specificities. The BAS layer handle this task.*

Verification Method: RoD, T

Parent: SAVOIR-DSS-ORG-750

SAVOIR.MMS.ORG.0300

PMS Services for PUS

The PMS shall comply with the packet management services required by the on-board software for supporting implementations of PUS service 15.

Rationale: *The PUS service 15 is the standard operational service to access packets.*

Verification Method: RoD, T

Parent: SAVOIR-DSS-ORG-760

8 DATA STORAGE SERVICES

As for previous section, the concept of Mass Memory System (MMS) is used in order to avoid overlapping with the concept of Data Storage System that is the subject of [SRD]. The definition of Mass Memory System is that a Data Storage System is made of one or several Mass Memory Systems.

8.1 Block Access System Service

The overall Data Storage memory is organised in Block Access Units (BAUs). The blocks can have different characteristics (e.g. size, writing and locking capability) that depends on the physical media and on the configuration selected at System level. This section provides requirements related to the Block Access System Service.

8.1.1 Service definition

SAVOIR.MMS.BAS.0100

Block Access Unit

A BAU shall be composed by the list of SAUs being part of the BAU.

Rationale: *As per definition, a BAU is a list of SAUs.*

Comment: *Each SAU that is part of the list is identified by the Storage Media and the address of the SAU within that Storage Media. For simplifying accesses, BAUs are usually made of contiguous number of SAUs having identical size. In that case, the BAU is identified by the Storage Media, the address of the first SAU and a number of SAUs.*

Verification Method: T

SAVOIR.MMS.BAS.0110

Block Access Unit identification

A BAU shall be uniquely identified within the MMS.

Rationale: *To avoid any ambiguity on the BAU.*

Comment: *The BAU identifiers can be defined statically (e.g. through configuration parameters of the MMS or implicit rule) or dynamically by a User Entity.*

Verification Method: T

Parent: SAVOIR-DSS-ORG-030

SAVOIR.MMS.BAS.0120

List of Block Access Units

A list of BAUs shall identify all the BAUs that are part of the list.

Rationale: *List of BAUs are used to perform operations on more than one BAU.*

Comment: *As matter of simplification, it is possible to only operate on contiguous list of BAUs that can identified by the first BAU and a number of BAUs.*

Verification Method: T

SAVOIR.MMS.BAS.0130

Block Access Unit read

A Block Access System shall support read access to all BAUs.

Rationale: *All BAUs shall be accessible in read mode.*

Comment: *The read can be performed at level of each individual BAU or at level of a list of BAUs.*

Verification Method: T

Parent: SAVOIR-DSS-ORG-030

SAVOIR.MMS.BAS.0140

Block Access Unit write

A Block Access System shall support write access to BAUs not identified as read-only.

Rationale: *All writable BAUs can be accessible in write mode by any User Entity.*

Comment: *The write can be performed at level of each individual BAU or at level of a list of BAUs.*

Verification Method: T

Parent: SAVOIR-DSS-ORG-030

SAVOIR.MMS.BAS.0150

Block Access Unit erase

A Block Access System shall support erasing the content of BAUs not identified as read-only.

Rationale: *All writable BAUs can be erased.*

Comment: *Erasing a BAU consists in setting its content to a particular value that is usually 0 but any other value or pattern can be used.*

Verification Method: T

SAVOIR.MMS.BAS.0160

Block Access Unit disable

A Block Access System shall support disabling BAUs.

Rationale: *All BAUs can be disabled, e.g. when they are faulty.*

Comment: *When disabled a BAU cannot be used as part of a new Store. If a disabled BAU is part of an existing Store, the Data Management System cannot access it anymore.*

Verification Method: T

SAVOIR.MMS.BAS.0170

Block Access Unit enable

A Block Access System shall support enabling BAUs.

Rationale: *All BAUs can be enabled, e.g. after having been checked and found safe to be used.*

Comment: *When enabled a BAU can be used as part of a Store (e.g. included in a new Store or added to an existing one when extending it or accessed if already part of an existing Store).*

Verification Method: T

SAVOIR.MMS.BAS.0180

Block Access Unit lock

A Block Access System shall support locking BAUs by a particular User Entity in:

- Exclusive_Read_Only: The User Entity can only Read the BAUs. Other User Entities cannot read or write the BAU.
- Read_Only: All User Entities can only read from the BAU.
- Exclusive_Access: The User Entity can read from and write to the BAU. Other User Entities cannot read from or write to the BAU;
- Single_Writer: The User Entity can read from and write to the BAU. Other User Entities can only read from the BAU.

Rationale: *BAUs can be locked in order to protect them against concurrent accesses.*

Comment: *Locking is usually managed by the Data Management System in order to protect BAUs against concurrent accesses, in particular BAUs that are used to store the structures used by the Data Management System itself.*

Verification Method: T

Parent: SAVOIR-DSS-ORG-045

SAVOIR.MMS.BAS.0190

Block Access Unit unlock

A Block Access System shall support unlocking BAUs by a particular User Entity.

Rationale: *BAUs can be unlocked after having been locked.*

Comment: *After being unlocked, the BAU can be used by any User Entity, i.e. read and write accesses.*

Verification Method: T

Parent: SAVOIR-DSS-ORG-045

8.1.2 Service parameters

SAVOIR.MMS.BAS.0200

BAU Identifier

The BAU Identifier parameter shall uniquely identify a BAU.

Rationale: *To avoid any ambiguity on the BAU to be accessed.*

Verification Method: T

SAVOIR.MMS.BAS.0210

BAU List

The BAU List parameter shall be made of a list of BAUs.

Rationale: *To execute an operation on several BAUs*

Verification Method: T

SAVOIR.MMS.BAS.0220

BAU Erase Pattern

The BAU Erase Pattern parameter shall define the pattern to be used when erasing a BAU.

Rationale: *To set the content of a BAU to any predefined pattern.*

Verification Method: T

SAVOIR.MMS.BAS.0230

BAU Lock Type

The BAU Lock Type parameter shall specify a lock action on a BAU. Possible values are:

- *Exclusive_Read_Only*: The lock-owner can only read from the BAU. Other user entities cannot read from or write to the BAU;
- *Read_Only*: The lock-owner can only read from the BAU. Other user entities can only read from the BAU (i.e. everyone can read);
- *Exclusive_Access*: The lock-owner can read from and write to the BAU. Other user entities cannot read from or write to the BAU;
- *Single_Writer*: The lock-owner can read from and write to the BAU. Other user entities can only read from the BAU.

Comment: *No lock applied on a BAU means all User Entities can read from and write to the BAU.*

Verification Method: T

SAVOIR.MMS.BAS.0240

BAU Transaction Identifier

The BAU Transaction Identifier shall be a value, assigned by the invoking User Entity, which is subsequently used to associate indication primitives with the causal request primitives.

Rationale: *The transaction Identifier ensures the correct management of requests in a multi-User Entity system and the User Entity is able to correlate all indications to a service request.*

Verification Method: T

SAVOIR.MMS.BAS.0250

BAU List Content

The BAU List Content shall be the data contained in all BAUs from a BAU list.

Rationale: *The Block Access Unit List Content contains the data that are read from a BAU list or that will be written into the Storage Media.*

Verification Method: T

SAVOIR.MMS.BAS.0260

BAU Result Metadata

The BAU Result Metadata parameter shall be used to provide information generated by the Block Access System provider to the User Entity regarding result of the execution of a request.

Comment: As an example it could indicate that the specified request cannot be serviced due to Block Access being disabled of locked by another User Entity.

Verification Method: T

8.1.3 Service interface

SAVOIR.MMS.BAS.0270

Block Access System primitives

The Bloc Access System shall provide the following primitives:
- READ_BAU.request, READ_BAU.indication
- WRITE_BAU.request, WRITE_BAU.indication
- ERASE_BAU.request, ERASE_BAU.indication
- DISABLE_BAU.request, DISABLE_BAU.indication
- ENABLE_BAU.request, ENABLE_BAU.indication
- LOCK_BAU.request, LOCK_BAU.indication
- UNLOCK_BAU.request, UNLOCK_BAU.indication

Rationale: These primitives are required to manage the BAUs.

Verification Method: T

Parent: SAVOIR-DSS-ORG-030, SAVOIR-DSS-ORG-040, SAVOIR-DSS-ORG-045

8.1.3.1 READ_BAU.request

8.1.3.1.1 Function

SAVOIR.MMS.BAS.0280

READ_BAU Function

The READ_BAU.request primitive shall be passed to the Block Access System provider to request in-order reading of a list of BAUs.

Verification Method: T

8.1.3.1.2 Semantics

SAVOIR.MMS.BAS.0290

READ_BAU Semantics

The READ_BAU.request primitive shall use the following semantics:
READ_BAU.request(BAU Transaction Identifier, BAU List).

Verification Method: T

8.1.3.1.3 When generated

SAVOIR.MMS.BAS.0300

READ_BAU When generated

The READ_BAU.request primitive shall be passed to the Block Access System provider to request the read of the content of all the BAUs identified in the BAU List.

Verification Method: T

8.1.3.1.4 Effect on receipt

SAVOIR.MMS.BAS.0310

READ_BAU effect on receipt

Receipt of the READ_BAU.request primitive shall cause the Block Access System provider to read the content of all the BAUs identified in the BAU List provided as parameter from the Storage Media.

Verification Method: T

8.1.3.1.5 Additional constraints

None

8.1.3.2 READ_BAU.indication

8.1.3.2.1 Function

SAVOIR.MMS.BAS.0320

READ_BAU.indication function

The READ_BAU.indication primitive shall be used to pass the data read from BAUs to the User Entity.

Rationale: The function returns the data read from the Storage Media.

Verification Method: T

8.1.3.2.2 Semantics

SAVOIR.MMS.BAS.0330

READ_BAU.indication semantics

The READ_BAU.indication primitive shall use the following semantics:
READ_BAU.indication(BAU Transaction Identifier, BAU List Content, BAU Result Metadata).

Verification Method: T

8.1.3.2.3 When generated

SAVOIR.MMS.BAS.0340

READ_BAU.indication When generated

The READ_BAU.indication primitive shall be passed by the Block Access System provider to the requesting User Entity in response to the READ_BAU.request with BAU Result Metadata indicating if the request was executed successfully or not.

Verification Method: T

SAVOIR.MMS.BAS.0350

READ_BAU.indication When generated success

When READ_BAU.request is successful, the BAU List Content shall provide the data read from BAU List.

Verification Method: T

SAVOIR.MMS.BAS.0360

READ_BAU.indication When generated failure

When READ_BAU.request is unsuccessful, the BAU Result Metadata shall provide the reason of the failure.

Comment: The behaviour of the BAS is not specified so the User Entity shall consider the complete BAU List Content as invalid.

Verification Method: T

8.1.3.2.4 Effect on receipt

The response of the User Entity to a READ_BAU.indication is unspecified.

8.1.3.2.5 Additional constraints

SAVOIR.MMS.BAS.0370

READ_BAU.indication BAU Invalid

The READ_BAU.indication Result Metadata shall report a failure if any of the requested BAU is not existing in the MMS.

Rationale: Reading a BAU that is not in the any Storage Media is not possible.

Verification Method: T

SAVOIR.MMS.BAS.0380

READ_BAU.indication BAU Disabled

The READ_BAU.indication Result Metadata shall report a failure if any of the requested BAU is disabled.

Rationale: Reading a disabled BAU is not authorized.

Verification Method: T

SAVOIR.MMS.BAS.0390

READ_BAU.indication BAU Locked

The READ_BAU.indication Result Metadata shall report a failure if any of the requested BAU is locked by another User Entity in Exclusive_Read_Only or Exclusive_Access modes.

Rationale: Reading a BAU locked by another User Entity is not authorized.

Verification Method: T

8.1.3.3 WRITE_BAU.request

8.1.3.3.1 Function

SAVOIR.MMS.BAS.0400

WRITE_BAU Function

The WRITE_BAU.request primitive shall be passed to the Block Access System provider to request in-order writing of a list of BAUs.

Verification Method: T

8.1.3.3.2 Semantics

SAVOIR.MMS.BAS.0410

WRITE_BAU Semantics

The WRITE_BAU.request primitive shall use the following semantics:
WRITE_BAU.request(BAU Transaction Identifier, BAU List, BAU List Content).

Verification Method: T

8.1.3.3.3 When generated

SAVOIR.MMS.BAS.0420

WRITE_BAU When generated

The WRITE_BAU.request primitive shall be passed to the Block Access System provider to request the write data contained into the BAU List Content into the BAUs identified in the BAU List.

Verification Method: T

8.1.3.3.4 Effect on receipt

SAVOIR.MMS.BAS.0430

WRITE_BAU effect on receipt

Receipt of the WRITE_BAU.request primitive shall cause the Block Access System provider to write on the Storage Media the data contained into the BAU List Content considering BAUs identified in the BAU List.

Verification Method: T

8.1.3.3.5 Additional constraints

8.1.3.4 WRITE_BAU.indication

8.1.3.4.1 Function

SAVOIR.MMS.BAS.0440

WRITE_BAU.indication function

The WRITE_BAU.indication primitive shall be used to indicate the outcome of the data writing on the Storage Media.

Rationale: The function returns the status of the write operation on the Storage Media.

Verification Method: T

8.1.3.4.2 Semantics

SAVOIR.MMS.BAS.0450

WRITE_BAU.indication semantics

The WRITE_BAU.indication primitive shall use the following semantics:
WRITE_BAU.indication(BAU Transaction Identifier, BAU Result Metadata).

Verification Method: T

8.1.3.4.3 When generated

SAVOIR.MMS.BAS.0460

WRITE_BAU.indication When generated

The WRITE_BAU.indication primitive shall be passed by the Block Access System provider to the requesting User Entity in response to the WRITE_BAU.request with BAU Result Metadata indicating if the request was executed successfully or not.

Verification Method: T

SAVOIR.MMS.BAS.0470

WRITE_BAU.indication When generated failure

When WRITE_BAU.request is unsuccessful, the BAU Result Metadata shall provide the reason of the failure.

Comment: The behaviour of the BAS is not specified so the User Entity shall consider that the complete operation is unsuccessful and that data related to the BAU List are unknown.

Verification Method: T

8.1.3.4.4 Effect on receipt

The response of the User Entity to a WRITE_BAU.indication is unspecified.

8.1.3.4.5 Additional constraints

SAVOIR.MMS.BAS.0480

WRITE_BAU.indication BAU Invalid

The WRITE_BAU.indication BAU Result Metadata shall report a failure if any of the requested BAU is not existing in the MMS.

Rationale: Writing a BAU that is not in the any Storage Media is not possible.

Verification Method: T

SAVOIR.MMS.BAS.0490

WRITE_BAU.indication BAU disabled

The WRITE_BAU.indication BAU Result Metadata shall report a failure if any of the requested BAU is disabled.

Rationale: Writing into a disabled BAU is not authorized.

Verification Method: T

SAVOIR.MMS.BAS.0500

WRITE_BAU.indication BAU locked

The WRITE_BAU.indication BAU Result Metadata shall report a failure if any of the requested BAU is locked by another User Entity in any mode.

Rationale: Writing into a BAU by another User Entity is not authorised.

Verification Method: T

8.1.3.5 ERASE_BAU.request

8.1.3.5.1 Function

SAVOIR.MMS.BAS.0510

ERASE_BAU.request Function

The ERASE_BAU.request primitive shall be passed to the Block Access System provider to request in-order erasing of a list of BAUs.

Verification Method: T

8.1.3.5.2 Semantics

SAVOIR.MMS.BAS.0520

ERASE_BAU.request Semantics

The ERASE_BAU.request primitive shall use the following semantics:
ERASE_BAU.request(BAU Transaction Identifier, BAU List, BAU Erase Pattern).

Verification Method: T

8.1.3.5.3 When generated

SAVOIR.MMS.BAS.0530

ERASE_BAU.request When generated

The ERASE_BAU.request primitive shall be passed to the Block Access System provider to request the erasing of the content of all the BAUs identified in the BAU List.

Verification Method: T

8.1.3.5.4 Effect on receipt

SAVOIR.MMS.BAS.0540

ERASE_BAU.request effect on receipt

Receipt of the ERASE_BAU.request primitive shall cause the Block Access System provider to write the provided BAU Erase Pattern parameter in the content of all the BAUs identified in the BAU list provided as parameter from the Storage Media.

Rationale: *The BAU Erase consists in filling the complete BAU with the provided BAU Erase Pattern.*

Comment: *The pattern is repeated in order to fill the each complete BAU that is part of the BAU List.*

Verification Method: T

8.1.3.5.5 Additional constraints

8.1.3.6 ERASE_BAU.indication

8.1.3.6.1 Function

SAVOIR.MMS.BAS.0550

ERASE_BAU.indication function

The ERASE_BAU.indication primitive shall be used to indicate the outcome of the erasing of the BAUs on the Storage Media.

Rationale: *The function returns the status of the operation.*

Verification Method: T

8.1.3.6.2 Semantics

SAVOIR.MMS.BAS.0560

ERASE_BAU.indication semantics

The ERASE_BAU.indication primitive shall use the following semantics:
ERASE_BAU.indication(BAU Transaction Identifier, BAU Result Metadata).

Verification Method: T

8.1.3.6.3 When generated

SAVOIR.MMS.BAS.0570

ERASE_BAU.indication When generated

The ERASE_BAU.indication primitive shall be passed by the Block Access System provider to the requesting User Entity in response to the ERASE_BAU.request with BAU Result Metadata indicating if the request was executed successfully or not.

Verification Method: T

SAVOIR.MMS.BAS.0580

ERASE_BAU.indication When generated failure

When ERASE_BAU.request is unsuccessful, the BAU Result Metadata shall provide the reason of the failure.

Comment: The behaviour of the BAS is not specified so the User Entity shall consider the data contained in the BAUs identified in the BAU List as unknown.

Verification Method: T

8.1.3.6.4 Effect on receipt

The response of the User Entity to a ERASE_BAU.indication is unspecified.

8.1.3.6.5 Additional constraints

SAVOIR.MMS.BAS.0590

ERASE_BAU.indication BAU Invalid

The ERASE_BAU.indication BAU Result Metadata shall report a failure if any of the requested BAU is not existing in the MMS.

Rationale: Erasing a BAU that is not in the any Storage Media is not possible.

Verification Method: T

SAVOIR.MMS.BAS.0600

ERASE_BAU.indication BAU disabled

The ERASE_BAU.indication ERASE Result Metadata shall report a failure if any of the requested BAU is disabled.

Rationale: Erasing a disabled BAU is not authorized.

Verification Method: T

SAVOIR.MMS.BAS.0610

ERASE_BAU.indication BAU locked

The ERASE_BAU.indication ERASE Result Metadata shall report a failure if any of the requested BAU is locked by another User Entity in any mode.

Rationale: Erasing a BAU locked by another User Entity is not authorized.

Verification Method: T

8.1.3.7 DISABLE_BAU.request

8.1.3.7.1 Function

SAVOIR.MMS.BAS.0620

DISABLE_BAU.request Function

The DISABLE_BAU.request primitive shall be passed to the Block Access System provider to request disabling a list of BAUs.

Verification Method: T

8.1.3.7.2 Semantics

SAVOIR.MMS.BAS.0630

DISABLE_BAU.request Semantics

The DISABLE_BAU.request primitive shall use the following semantics:
DISABLE_BAU.request(BAU Transaction Identifier, BAU List).

Verification Method: T

8.1.3.7.3 When generated

SAVOIR.MMS.BAS.0640

DISABLE_BAU.request When generated

The DISABLE_BAU.request primitive shall be passed to the Block Access System provider to request the disabling of all the BAUs identified in the BAU List.

Verification Method: T

8.1.3.7.4 Effect on receipt

SAVOIR.MMS.BAS.0650

DISABLE_BAU.request effect on receipt

Receipt of the DISABLE_BAU.request primitive shall cause the Block Access System provider to disable all the BAUs identified in the BAU list provided as parameter.

Verification Method: T

8.1.3.7.5 Additional constraints

8.1.3.8 DISABLE_BAU.indication

8.1.3.8.1 Function

SAVOIR.MMS.BAS.0660

DISABLE_BAU.indication Function

The DISABLE_BAU.indication primitive shall be used to indicate the outcome of the disabling of the BAUs identified in the BAU List provided as parameter.

Rationale: The function returns the status of the operation.

Verification Method: T

8.1.3.8.2 Semantics

SAVOIR.MMS.BAS.0670

DISABLE_BAU.indication Semantics

The DISABLE_BAU.indication primitive shall use the following semantics:
DISABLE_BAU.indication(BAU Transaction Identifier, BAU Result Metadata).

Verification Method: T

8.1.3.8.3 When generated

SAVOIR.MMS.BAS.0680

DISABLE_BAU.indication When generated

The DISABLE_BAU.indication primitive shall be passed by the Block Access System provider to the requesting User Entity in response to the DISABLE_BAU.request with BAU Result Metadata indicating if the request was executed successfully or not.

Verification Method: T

SAVOIR.MMS.BAS.0690

DISABLE_BAU.indication When generated failure

When DISABLE_BAU.request is unsuccessful, the BAU Result Metadata shall provide the reason of the failure.

Comment: The behaviour of the BAS is not specified so the User Entity shall consider the status of all BAUs identified in the BAU List as unknown. It is possible to disable any existing BAU even if locked by any User Entity.

Verification Method: T

8.1.3.8.4 Effect on receipt

The response of the User Entity to a DISABLE_BAU.indication is unspecified.

8.1.3.8.5 Additional constraints

SAVOIR.MMS.BAS.0700

DISABLE_BAU.indication BAU Invalid

The DISABLE_BAU.indication Result Metadata shall report a failure if any of the requested BAU is not existing in the MMS.

Rationale: Disabling a BAU that is not in the any Storage Media is not possible.

Verification Method: T

8.1.3.9 ENABLE_BAU.request

8.1.3.9.1 Function

SAVOIR.MMS.BAS.0710

ENABLE_BAU.request Function

The ENABLE_BAU.request primitive shall be passed to the Block Access System provider to request enabling a list of BAUs.

Verification Method: T

8.1.3.9.2 Semantics

SAVOIR.MMS.BAS.0720

ENABLE_BAU.request Semantics

The ENABLE_BAU.request primitive shall use the following semantics:
ENABLE_BAU.request(BAU Transaction Identifier, BAU List).

Verification Method: T

8.1.3.9.3 When generated

SAVOIR.MMS.BAS.0730

ENABLE_BAU.request When generated

The ENABLE_BAU.request primitive shall be passed to the Block Access System provider to request the enabling of all the BAUs identified in the BAU List.

Verification Method: T

SAVOIR.MMS.BAS.0740

ENABLE_BAU.request Lock reset

When enabling a BAU, the Block Access System provider shall reset any information related to the locking status of the BAU.

Verification Method: T

8.1.3.9.4 Effect on receipt

SAVOIR.MMS.BAS.0750

ENABLE_BAU.request effect on receipt

Receipt of the ENABLE_BAU.request primitive shall cause the Block Access System provider to enable all the BAUs identified in the BAU list provided as parameter.

Verification Method: T

8.1.3.9.5 Additional constraints

8.1.3.10 ENABLE_BAU.indication

8.1.3.10.1 Function

SAVOIR.MMS.BAS.0760

ENABLE_BAU.indication function

The ENABLE_BAU.indication primitive shall be used to indicate the outcome of the enabling of the BAUs identified in the BAU List provided as parameter.

Rationale: The function returns the status of the operation.

Verification Method: T

8.1.3.10.2 Semantics

SAVOIR.MMS.BAS.0770

ENABLE_BAU.indication semantics

The ENABLE_BAU.indication primitive shall use the following semantics:
ENABLE_BAU.indication(BAU Transaction Identifier, BAU Result Metadata).

Verification Method: T

8.1.3.10.3 When generated

SAVOIR.MMS.BAS.0780

ENABLE_BAU.indication when generated

The ENABLE_BAU.indication primitive shall be passed by the Block Access System provider to the requesting User Entity in response to the ENABLE_BAU.request with BAU Result Metadata indicating if the request was executed successfully or not.

Verification Method: T

SAVOIR.MMS.BAS.0790

ENABLE_BAU.indication When generated failure

When ENABLE_BAU.request is unsuccessful, the BAU Result Metadata shall provide the reason of the failure.

Comment: The behaviour of the BAS is not specified so the User Entity shall consider the status of all BAUs identified in the BAU List as unknown. Enabling a BAU that is already enabled in not considered as a failure.

Verification Method: T

8.1.3.10.4 Effect on receipt

The response of the User Entity to a ENABLE_BAU.indication is unspecified.

8.1.3.10.5 Additional constraints

SAVOIR.MMS.BAS.0800

ENABLE_BAU.indication BAU Invalid

The ENABLE_BAU.indication Result Metadata shall report a failure if any of the requested BAU is not existing in the MMS.

Rationale: Enabling a BAU that is not in the any Storage Media is not possible.

Verification Method: T

8.1.3.11 LOCK_BAU.request

8.1.3.11.1 Function

SAVOIR.MMS.BAS.0810

LOCK_BAU.request Function

The LOCK_BAU.request primitive shall be passed to the Block Access System provider to request the locking of a list of BAUs.

Verification Method: T

8.1.3.11.2 Semantics

SAVOIR.MMS.BAS.0820

LOCK_BAU.request Semantics

The LOCK_BAU.request primitive shall use the following semantics:
LOCK_BAU.request(BAU Transaction Identifier, BAU List, BAU Lock Type, User Entity Identifier).

Comment: The User Entity Identifier is specified as it may be different from the User Entity that is calling the service (e.g. a MMS service locking the BAU for a particular User Entity.

Verification Method: T

8.1.3.11.3 When generated

SAVOIR.MMS.BAS.0830

LOCK_BAU.request When generated

The LOCK_BAU.request primitive shall be passed to the Block Access System provider to request the locking of all the BAUs identified in the BAU List.

Verification Method: T

8.1.3.11.4 Effect on receipt

SAVOIR.MMS.BAS.0840

LOCK_BAU.request effect on receipt

Receipt of the LOCK_BAU.request primitive shall cause the Block Access System provider to lock all the BAUs identified in the BAU list provided as parameter.

Verification Method: T

8.1.3.11.5 Additional constraints

8.1.3.12 LOCK_BAU.indication

8.1.3.12.1 Function

SAVOIR.MMS.BAS.0850

LOCK_BAU.indication function

The LOCK_BAU.indication primitive shall be used to indicate the outcome of the locking of the BAUs identified in the BAU list provided as parameter.

Rationale: The function returns the status of the operation.

Verification Method: T

8.1.3.12.2 Semantics

SAVOIR.MMS.BAS.0860

LOCK_BAU.indication semantics

The LOCK_BAU.indication primitive shall use the following semantics:
LOCK_BAU.indication(BAU Transaction Identifier, BAU Result Metadata).

Verification Method: T

8.1.3.12.3 When generated

SAVOIR.MMS.BAS.0870

LOCK_BAU.indication when generated

The LOCK_BAU.indication primitive shall be passed by the Block Access System provider to the requesting User Entity in response to the LOCK_BAU.request with BAU Result Metadata indicating if the request was executed successfully or not.

Verification Method: T

SAVOIR.MMS.BAS.0880

LOCK_BAU.indication When generated failure

When LOCK_BAU.request is unsuccessful, the BAU Result Metadata shall provide the reason of the failure.

Comment: The behaviour of the BAS is not specified so the User Entity shall consider the locking status of all BAUs identified in the BAU List as unknown.

Verification Method: T

8.1.3.12.4 Effect on receipt

The response of the User Entity to a LOCK_BAU.indication is unspecified.

8.1.3.12.5 Additional constraints

SAVOIR.MMS.BAS.0890

LOCK_BAU.indication BAU Invalid

The LOCK_BAU.indication BAU Result Metadata shall report a failure if any of the requested BAU is not existing in the MMS.

Rationale: Locking a BAU that is not in the any Storage Media is not possible.

Verification Method: T

SAVOIR.MMS.BAS.0900

LOCK_BAU.indication BAU disabled

The LOCK_BAU.indication BAU Result Metadata shall report a failure if any of the requested BAU is disabled.

Rationale: Locking a disabled BAU is not possible.

Verification Method: T

SAVOIR.MMS.BAS.0910

LOCK_BAU.indication BAU locked

The LOCK_BAU.indication BAU Result Metadata shall report a failure if any of the requested BAU is locked by another User Entity in any mode.

Rationale: Locking a BAU locked by another User Entity is not possible.

Comment: The behaviour of the MMS with respect to the different locking services (BAU, Store, FMS) is let free to implementer.

Verification Method: T

8.1.3.13 UNLOCK_BAU.request

8.1.3.13.1 Function

SAVOIR.MMS.BAS.0920

UNLOCK_BAU.request Function

The UNLOCK_BAU.request primitive shall be passed to the Block Access System provider to request the unlocking of a list of BAUs.

Verification Method: T

8.1.3.13.2 Semantics

SAVOIR.MMS.BAS.0930

UNLOCK_BAU.request Semantics

The UNLOCK_BAU.request primitive shall use the following semantics:
UNLOCK_BAU.request(BAU Transaction Identifier, BAU List, User Entity Identifier).

Comment: The User Entity Identifier is specified as it may be different from the User Entity that is calling the service (e.g. a MMS service unlocking the BAU for a particular User Entity.

Verification Method: T

8.1.3.13.3 When generated

SAVOIR.MMS.BAS.0940

UNLOCK_BAU.request When generated

The UNLOCK_BAU.request primitive shall be passed to the Block Access System provider to request the unlocking of all the BAUs identified in the BAU List.

Verification Method: T

8.1.3.13.4 Effect on receipt

SAVOIR.MMS.BAS.0950

UNLOCK_BAU.request effect on receipt

Receipt of the UNLOCK_BAU.request primitive shall cause the Block Access System provider to unlock all the BAUs identified in the BAU list provided as parameter.

Verification Method: T

8.1.3.13.5 Additional constraints

8.1.3.14 UNLOCK_BAU.indication

8.1.3.14.1 Function

SAVOIR.MMS.BAS.0960

UNLOCK_BAU.indication function

The UNLOCK_BAU.indication primitive shall be used to indicate the outcome of the unlocking of the BAUs identified in the BAU list provided as parameter.

Rationale: The function returns the status of the operation.

Verification Method: T

8.1.3.14.2 Semantics

SAVOIR.MMS.BAS.0970

UNLOCK_BAU.indication semantics

The UNLOCK_BAU.indication primitive shall use the following semantics:
UNLOCK_BAU.indication(BAU Transaction Identifier, BAU Result Metadata).

Verification Method: T

8.1.3.14.3 When generated

SAVOIR.MMS.BAS.0980

UNLOCK_BAU.indication when generated

The UNLOCK_BAU.indication primitive shall be passed by the Block Access System provider to the requesting User Entity in response to the UNLOCK_BAU.request with BAU Result Metadata indicating if the request was executed successfully or not.

Verification Method: T

SAVOIR.MMS.BAS.0990

UNLOCK_BAU.indication When generated failure

When UNLOCK_BAU.request is unsuccessful, the BAU Result Metadata shall provide the reason of the failure.

Comment: The behaviour of the BAS is not specified so the User Entity shall consider the locking status of all BAUs identified in the BAU List as unknown.

Verification Method: T

8.1.3.14.4 Effect on receipt

The response of the User Entity to a UNLOCK_BAU.indication is unspecified.

8.1.3.14.5 Additional constraints

SAVOIR.MMS.BAS.1000

UNLOCK_BAU.indication BAU Invalid

The UNLOCK_BAU.indication BAU Result Metadata shall report a failure if any of the requested BAU is not existing in the MMS.

Rationale: Unlocking a BAU that is not in the any Storage Media is not possible.

Verification Method: T

SAVOIR.MMS.BAS.1010

UNLOCK_BAU.indication BAU disabled

The UNLOCK_BAU.indication BAU Result Metadata shall report a failure if any of the requested BAU is disabled.

Rationale: Unlocking a disabled BAU is not possible.

Verification Method: T

SAVOIR.MMS.BAS.1020

UNLOCK_BAU.indication BAU locked by another User Entity

The UNLOCK_BAU.indication BAU Result Metadata shall report a failure if any of the requested BAU is locked by another User Entity in any mode.

Rationale: *Unlocking a BAU locked by another User Entity is not possible.*

Comment: *If found too restrictive, it is possible to implement a User Entity having all the rights on the MMS or being able to impersonate the User Entity that locked the BAU.*

Verification Method: T

8.2 Store Service

This section provides requirements related to the Store Service provided by the Memory Management System. It includes the definition of the services and the interfaces to those services.

8.2.1 Service definition

SAVOIR.MMS.STO.0100

Store create

The MMS shall provide a User Entity the ability to create a Store.

Rationale: *The creation of a store provides flexibility in the management of the available data storage area.*

Comment: *The creation of the Store provides the capability to dynamically manage the partitioning of the available memory. Stores can also be defined statically through configuration tables.*

Verification Method: T

Parent: SAVOIR-DSS-ORG-060

SAVOIR.MMS.STO.0110

Store delete

The MMS shall provide a User Entity the ability to delete a Store.

Rationale: *The deletion of a Store provides flexibility in the management of the available data storage area.*

Comment: *The deletion of the Store provides the capability to dynamically manage the partitioning of the available memory.*

Comment: *The BAUs that was part of the deleted Store can later be included in new Stores or used to expand an existing Store.*

Verification Method: T

Parent: SAVOIR-DSS-ORG-070

SAVOIR.MMS.STO.0120

Store expand

The MMS shall provide a User Entity the ability to expand a Store.

Rationale: *The expansion of a Store provides flexibility in the management of the available data storage area.*

Comment: *The expansion of the Store provides the capability to dynamically manage the partitioning of the available memory.*

Comment: *The expansion is performed by adding available BAUs (i.e. not disabled and not used by any other Store).*

Verification Method: T

Parent: SAVOIR-DSS-ORG-110

SAVOIR.MMS.STO.0130

Store reduce

The MMS shall provide a User Entity the ability to reduce a Store.

Rationale: *The reduction of a Store provides flexibility in the management of the available data storage area.*

Comment: *The reduction of the Store provides the capability to dynamically manage the partitioning of the available memory.*

Comment: *The reduction is performed by adding available BAUs (i.e. not disabled and not used by any other Store).*

Verification Method: T

Parent: SAVOIR-DSS-ORG-090

SAVOIR.MMS.STO.0140

Store lock

The MMS shall provide a User Entity the ability to lock a Store.

Rationale: *The locking of a Store ensure protection of the Store against unwanted concurrent accesses.*

Comment: *The relation with the locking at BAU level is not specified and let free to the implementer, i.e. can be fully independent or locking of a Store will lock all BAUs that are part of this Store. Locking and unlocking services have to be consistent.*

Verification Method: T

SAVOIR.MMS.STO.0150

Store unlock

The MMS shall provide a User Entity the ability to unlock a Store.

Rationale: *To release the lock on a Store.*

Comment: *The relation with the locking at BAU level is not specified and let free to the implementer, i.e. can be fully independent or unlocking of a Store will unlock all BAUs that are part of this Store. Locking and unlocking services have to be consistent.*

Verification Method: T

SAVOIR.MMS.STO.0155

Store erase

The MMS shall provide a User Entity the ability to erase a Store.

Rationale: *To reset the content of a Store, i.e. set the content of all the BAUs contained in the Store to default value.*

Comment: *The characteristics of the Store are not changed and the full capacity of the Store is made available at the end of the erase operation.*

Verification Method: T

Parent: SAVOIR-DSS-ORG-080

8.2.1 Service parameters

SAVOIR.MMS.STO.0160

Store Identifier

The Store Identifier parameter shall uniquely identify a Store.

Rationale: *To avoid any ambiguity on the Store to be accessed.*

Verification Method: T

SAVOIR.MMS.STO.0170

Store Lock Type

The Store Lock Type parameter shall specify a lock action on a Store. Possible values are:

- *Exclusive_Read_Only*: The lock-owner can only read from the Store. Other user entities cannot read from or write to the Store;
- *Read_Only*: The lock-owner can only read from the Store. Other user entities can only read from the Store (i.e. everyone can read);
- *Exclusive_Access*: The lock-owner can read from and write to the Store. Other user entities cannot read from or write to the Store;
- *Single_Writer*: The lock-owner can read from and write to the Store. Other user entities can only read from the Store.

Comment: *No lock applied on a Store means all User Entities can read from and write to the Store.*

Verification Method: T

SAVOIR.MMS.STO.0180

Store Transaction Identifier

The Store Transaction Identifier shall be a value, assigned by the invoking User Entity, which is subsequently used to associate indication primitives with the causal request primitives.

Rationale: *The Store Transaction Identifier ensures the correct management of requests in a multi-User Entity system and the User Entity is able to correlate all indications to a service request.*

Verification Method: T

SAVOIR.MMS.STO.0190

Store Result Metadata

The Store Result Metadata parameter shall be used to provide information generated by the MMS provider to the User Entity regarding result of the execution of a request.

Verification Method: T

8.2.1 Service interface

SAVOIR.MMS.STO.0200

Store primitives

The MMS shall provide the following primitives:
- CREATE_STORE.request, CREATE_STORE.indication
- DELETE_STORE.request, DELETE_STORE.indication
- EXPAND_STORE.request, EXPAND_STORE.indication
- REDUCE_STORE.request, REDUCE_STORE.indication
- LOCK_STORE.request, LOCK_STORE.indication
- UNLOCK_STORE.request, UNLOCK_STORE.indication
- ERASE_STORE.request, ERASE_STORE.indication

Rationale: These primitives are required to manage the Stores.

Verification Method: T

8.2.1.1 CREATE_STORE.request

8.2.1.1.1 Function

SAVOIR.MMS.STO.0210

CREATE_STORE.request Function

The CREATE_STORE.request primitive shall be passed to the MMS provider to request the creation of a Store.

Verification Method: T

8.2.1.1.2 Semantics

SAVOIR.MMS.STO.0220

CREATE_STORE.request Semantics

The CREATE_STORE.request primitive shall use the following semantics:
CREATE_STORE.request(Store Transaction Identifier, BAU List, Store Identifier).

Verification Method: T

8.2.1.1.3 When generated

SAVOIR.MMS.STO.0230

CREATE_STORE.request When generated

The CREATE_STORE.request primitive shall be passed to the MMS provider to request the creation of a Store made of all the BAUs identified in the BAU List.

Verification Method: T

8.2.1.1.4 Effect on receipt

SAVOIR.MMS.STO.0240

CREATE_STORE.request effect on receipt

Receipt of the CREATE_STORE.request primitive shall cause the MMS provider to create a Store and associate to this Store all the BAUs identified in the BAU list provided as parameter.

Comment: At creation of the Store, all BAUs are unlocked and the content of the BAUs is undefined.

Verification Method: T

8.2.1.1.5 Additional constraints

8.2.1.2 CREATE_STORE.indication

8.2.1.2.1 Function

SAVOIR.MMS.STO.0250

CREATE_STORE.indication Function

The CREATE_STORE.indication primitive shall be used to indicate the outcome of the creation of a store.

Rationale: The function returns the status of the operation.

Verification Method: T

8.2.1.2.2 Semantics

SAVOIR.MMS.STO.0260

CREATE_STORE.indication Semantics

The CREATE_STORE.indication primitive shall use the following semantics:
CREATE_STORE.indication(Store Transaction Identifier, Store Result Metadata).

Verification Method: T

8.2.1.2.3 When generated

SAVOIR.MMS.STO.0270

CREATE_STORE.indication When generated

The CREATE_STORE.indication primitive shall be passed by the MMS provider to the requesting User Entity in response to the CREATE_STORE.request with Store Result Metadata indicating if the request was executed successfully or not.

Verification Method: T

SAVOIR.MMS.STO.0280

CREATE_STORE.indication When generated failure

When CREATE_STORE.request is unsuccessful, the Store Result Metadata shall provide the reason of the failure.

Comment: The Store is not created and the BAUs identified in the BAU List remain available for being associated to any Store.

Verification Method: T

8.2.1.2.4 Effect on receipt

The response of the User Entity to a CREATE_STORE.indication is unspecified.

8.2.1.2.5 Additional constraints

SAVOIR.MMS.STO.0290

CREATE_STORE.indication BAU Invalid

The CREATE_STORE.indication Store Result Metadata shall report a failure if any of the requested BAU is not existing in the MMS.

Rationale: Associating to a Store a BAU that is not in the any Storage Media is not possible.

Verification Method: T

SAVOIR.MMS.STO.0300

CREATE_STORE.indication BAU Used

The CREATE_STORE.indication Store Result Metadata shall report a failure if any of the requested BAU is already associated to any other Store.

Rationale: It is not possible to associate a specific BAU to several Stores.

Comment: It is possible to add disabled BAUs to a Store but these BAUs will not be used until they are enabled.

Verification Method: T

8.2.1.3 DELETE_STORE.request

8.2.1.3.1 Function

SAVOIR.MMS.STO.0310

DELETE_STORE.request Function

The DELETE_STORE.request primitive shall be passed to the MMS provider to request the deletion of a Store.

Verification Method: T

8.2.1.3.2 Semantics

SAVOIR.MMS.STO.0320

DELETE_STORE.request Semantics

The DELETE_STORE.request primitive shall use the following semantics:
DELETE_STORE.request(Store Transaction Identifier, Store Identifier).

Verification Method: T

8.2.1.3.3 When generated

SAVOIR.MMS.STO.0330

DELETE_STORE.request When generated

The DELETE_STORE.request primitive shall be passed to the MMS provider to request the deletion of a Store and dissociate the BAUs identified in the BAU List from the Store.

Verification Method: T

8.2.1.3.4 Effect on receipt

SAVOIR.MMS.STO.0340

DELETE_STORE.request Effect on receipt

Receipt of the DELETE_STORE.request primitive shall cause the MMS provider to delete a Store and dissociate from the Store all the BAUs identified in the BAU list provided as parameter.

Comment: All the BAUs that were part of the Store can later be associated to any other Store newly created or expanded.

Verification Method: T

8.2.1.3.5 Additional constraints

8.2.1.4 DELETE_STORE.indication

8.2.1.4.1 Function

SAVOIR.MMS.STO.0350

DELETE_STORE.indication Function

The DELETE_STORE.indication primitive shall be used to indicate the outcome of the deletion of a Store.

Rationale: The function returns the status of the operation.

Verification Method: T

8.2.1.4.2 Semantics

SAVOIR.MMS.STO.0360

DELETE_STORE.indication Semantics

The DELETE_STORE.indication primitive shall use the following semantics:
DELETE_STORE.indication(Store Transaction Identifier, Store Result Metadata).

Verification Method: T

8.2.1.4.3 When generated

SAVOIR.MMS.STO.0370

DELETE_STORE.indication When generated

The DELETE_STORE.indication primitive shall be passed by the MMS provider to the requesting User Entity in response to the DELETE_STORE.request with Store Result Metadata indicating if the request was executed successfully or not.

Verification Method: T

SAVOIR.MMS.STO.0380

DELETE_STORE.indication When generated failure

When DELETE_STORE.request is unsuccessful, the Store Result Metadata shall provide the reason of the failure.

Comment: The Store is not deleted and the BAUs identified in the BAU List remain linked to the Store.

Verification Method: T

8.2.1.4.4 Effect on receipt

The response of the User Entity to a DELETE_STORE.indication is unspecified.

8.2.1.4.5 Additional constraints

SAVOIR.MMS.STO.0390

DELETE_STORE.indication BAU Locked

The DELETE_STORE.indication Store Result Metadata shall report a failure if any of the BAU associated to the Store is locked by any other User Entity.

Rationale: It is only possible to delete a Store that is not used.

Comment: During the deletion of the Store, the MMS has to ensure that no other User Entity is locking any BAU of the Store.

Verification Method: T

SAVOIR.MMS.STO.0400

DELETE_STORE.indication Store Locked

The DELETE_STORE.indication Store Result Metadata shall report a failure if the Store is locked by any other User Entity.

Rationale: It is not possible to delete a Store that is locked by another User Entity.

Verification Method: T

8.2.1.5 EXPAND_STORE.request

8.2.1.5.1 Function

SAVOIR.MMS.STO.0410

EXPAND_STORE.request Function

The EXPAND_STORE.request primitive shall be passed to the MMS provider to request the expansion of a Store.

Verification Method: T

8.2.1.5.2 Semantics

SAVOIR.MMS.STO.0420

EXPAND_STORE.request Semantics

The EXPAND_STORE.request primitive shall use the following semantics:
EXPAND_STORE.request(Store Transaction Identifier, BAU List, Store Identifier).

Verification Method: T

8.2.1.5.3 When generated

SAVOIR.MMS.STO.0430

EXPAND_STORE.request When generated

The EXPAND_STORE.request primitive shall be passed to the MMS provider to request the expansion of a Store by adding all the BAUs identified in the BAU List to the list of BAUs already associated to the Store.

Verification Method: T

8.2.1.5.4 Effect on receipt

SAVOIR.MMS.STO.0440

EXPAND_STORE.request Effect on receipt

Receipt of the EXPAND_STORE.request primitive shall cause the MMS provider to expand a Store and add to this Store all the BAUs identified in the BAU list provided as parameter.

Comment: All added BAUs are unlocked and the content of the BAUs is undefined.

Verification Method: T

8.2.1.5.5 Additional constraints

SAVOIR.MMS.STO.0445

EXPAND_STORE.request Store data preservation on expand

The MMS shall ensure that the data contained in the BAUs that are already part of a Store are preserved.

Rationale: The Store shall be expanded by addition of the BAU list to an existing Store that shall keep the data it contain unmodified.

Comment: The MMS preserves the content of the BAUs of the Store to extend unmodified. It is up to the DMS (e.g. FMS or PMS) to update its structure to reflect this additional space if necessary. In that case, the DMS may have to update the BAUs that are containing the information related to the management of the Store.

Verification Method: T

Parent: SAVOIR-DSS-ORG-100

8.2.1.6 EXPAND_STORE.indication

8.2.1.6.1 Function

SAVOIR.MMS.STO.0450

EXPAND_STORE.indication Function

The EXPAND_STORE.indication primitive shall be used to indicate the outcome of the expansion of a store.

Rationale: The function returns the status of the operation.

Verification Method: T

8.2.1.6.2 Semantics

SAVOIR.MMS.STO.0460

EXPAND_STORE.indication Semantics

The EXPAND_STORE.indication primitive shall use the following semantics:
EXPAND_STORE.indication(Store Transaction Identifier, Store Result Metadata).

Verification Method: T

8.2.1.6.3 When generated

SAVOIR.MMS.STO.0470

EXPAND_STORE.indication When generated

The EXPAND_STORE.indication primitive shall be passed by the MMS provider to the requesting User Entity in response to the EXPAND_STORE.request with Store Result Metadata indicating if the request was executed successfully or not.

Verification Method: T

SAVOIR.MMS.STO.0480

EXPAND_STORE.indication When generated failure

When CREATE_STORE.request is unsuccessful, the Store Result Metadata shall provide the reason of the failure.

Comment: The Store is not expanded and the BAUs identified in the BAU List remain available for being associated to any Store.

Verification Method: T

8.2.1.6.4 Effect on receipt

The response of the User Entity to a EXPAND_STORE.indication is unspecified.

8.2.1.6.5 Additional constraints

SAVOIR.MMS.STO.0490

EXPAND_STORE.indication BAU Invalid

The EXPAND_STORE.indication Store Result Metadata shall report a failure if any of the requested BAU is not existing in the MMS.

Rationale: Associating to a Store a BAU that is not in the any Storage Media is not possible.

Verification Method: T

SAVOIR.MMS.STO.0500

EXPAND_STORE.indication BAU Used

The EXPAND_STORE.indication StoreResult Metadata shall report a failure if any of the BAU identified in the BAU List is already associated to any other Store.

Rationale: It is only possible to associate a specific BAU to only one Store.

Verification Method: T

SAVOIR.MMS.STO.0510

EXPAND_STORE.indication Store Locked

The EXPAND_STORE.indication Store Result Metadata shall report a failure if the Store is locked by any other User Entity.

Rationale: It is not possible to expand a Store that is locked by another User Entity.

Verification Method: T

8.2.1.7 REDUCE_STORE.request

8.2.1.7.1 Function

SAVOIR.MMS.STO.0520

REDUCE_STORE.request Function

The REDUCE_STORE.request primitive shall be passed to the MMS provider to request the reduction of a Store.

Comment: The reduction of a Store can be used to remove disabled BAUs or BAUs that are in a Storage Media identified as faulty.

Verification Method: T

8.2.1.7.2 Semantics

SAVOIR.MMS.STO.0530

REDUCE_STORE.request Semantics

The REDUCE_STORE.request primitive shall use the following semantics:
REDUCE_STORE.request(Store Transaction Identifier, BAU List, Store Identifier).

Verification Method: T

8.2.1.7.3 When generated

SAVOIR.MMS.STO.0540

REDUCE_STORE.request When generated

The REDUCE_STORE.request primitive shall be passed to the MMS provider to request the reduction of a Store by dissociating all the BAUs identified in the BAU List from the list of BAUs already associated to the Store.

Verification Method: T

8.2.1.7.4 Effect on receipt

SAVOIR.MMS.STO.0550

REDUCE_STORE.request Effect on receipt

Receipt of the REDUCE_STORE.request primitive shall cause the MMS provider to reduce a Store and dissociate from the Store all the BAUs identified in the BAU list provided as parameter.

Comment: The BAUs that are dissociated from the Store can be later used to create a new Store or to expand an existing Store.

Comment: The DMS (e.g. FMS or PMS) has to update its internal structures to reflect the removal of the storage space.

Comment: The removal of BAUs that are used by the DMS (e.g. FMS or PMS) has potentially dramatic consequence on the data contained in the Store (e.g. loss of references to stored data or loss of data).

Verification Method: T

8.2.1.7.5 Additional constraints

8.2.1.8 REDUCE_STORE.indication

8.2.1.8.1 Function

SAVOIR.MMS.STO.0560

REDUCE_STORE.indication Function

The REDUCE_STORE.indication primitive shall be used to indicate the outcome of the reduction of a store.

Rationale: The function returns the status of the operation.

Verification Method: T

8.2.1.8.2 Semantics

SAVOIR.MMS.STO.0570

REDUCE_STORE.indication Semantics

The REDUCE_STORE.indication primitive shall use the following semantics:
REDUCE_STORE.indication(Store Transaction Identifier, Store Result Metadata).

Verification Method: T

8.2.1.8.3 When generated

SAVOIR.MMS.STO.0580

REDUCE_STORE.indication When generated

The REDUCE_STORE.indication primitive shall be passed by the MMS provider to the requesting User Entity in response to the REDUCE_STORE.request with Store Result Metadata indicating if the request was executed successfully or not.

Verification Method: T

SAVOIR.MMS.STO.0590

REDUCE_STORE.indication When generated failure

When REDUCE_STORE.request is unsuccessful, the Store Result Metadata shall provide the reason of the failure.

Comment: The Store is not reduced and the BAUs identified in the BAU List remain associated to the Store.

Verification Method: T

8.2.1.8.4 Effect on receipt

The response of the User Entity to a REDUCE_STORE.indication is unspecified.

8.2.1.8.5 Additional constraints

SAVOIR.MMS.STO.0600

REDUCE_STORE.indication BAU Invalid

The REDUCE_STORE.indication Store Result Metadata shall report a failure if any of the BAU identified in the BAU List is not associated to the Store.

Rationale: Dissociating a BAU that is not associated to the Store is not possible.

Comment: The BAUs that does not exist in the MMS cannot be part of a Store (verification is performed at creation and expansion).

Verification Method: T

SAVOIR.MMS.STO.0610

REDUCE_STORE.indication BAU Locked

The REDUCE_STORE.indication Store Result Metadata shall report a failure if any of the BAU identified in the BAU List is locked by another User Entity.

Rationale: It is not possible to associate to the Store a BAU that is locked by another User entity.

Verification Method: T

SAVOIR.MMS.STO.0620

REDUCE_STORE.indication Store Locked

The REDUCE_STORE.indication Store Result Metadata shall report a failure if the Store is locked by any other User Entity.

Rationale: It is not possible to reduce a Store that is locked by another User Entity.

Verification Method: T

8.2.1.9 LOCK_STORE.request

8.2.1.9.1 Function

SAVOIR.MMS.STO.0630

LOCK_STORE.request Function

The LOCK_STORE.request primitive shall be passed to the MMS provider to request the locking of a Store

Comment: Locking is managed at the level of the complete Store.

Verification Method: T

8.2.1.9.2 Semantics

SAVOIR.MMS.STO.0640

LOCK_STORE.request Semantics

The LOCK_STORE.request primitive shall use the following semantics:
LOCK_STORE.request(Store Transaction Identifier, Store Identifier, Store Lock Type, User Entity Identifier).

Comment: The User Entity Identifier is specified as it may be different from the User Entity that is calling the service (e.g. a MMS service locking the Store for a particular User Entity.

Verification Method: T

8.2.1.9.3 When generated

SAVOIR.MMS.STO.0650

LOCK_STORE.request When generated

The LOCK_STORE.request primitive shall be passed to the MMS provider to request the locking of the complete Store.

Comment: When locked, the Store can only be used by the User Entity owner of the lock. This includes all services related to Store, BAU and Data Management System (e.g. FMS and PMS).

Verification Method: T

8.2.1.9.4 Effect on receipt

SAVOIR.MMS.STO.0660

LOCK_STORE.request Effect on receipt

Receipt of the LOCK_STORE.request primitive shall cause the MMS provider to lock the complete Store identified by Store Identifier parameter.

Verification Method: T

8.2.1.9.5 Additional constraints

8.2.1.10 LOCK_STORE.indication

8.2.1.10.1 Function

SAVOIR.MMS.STO.0670

LOCK_STORE.indication Function

The LOCK_STORE.indication primitive shall be used to indicate the outcome of the locking of the Store identified by the Store Identifier provided as parameter.

Rationale: The function returns the status of the operation.

Verification Method: T

8.2.1.10.2 Semantics

SAVOIR.MMS.STO.0680

LOCK_STORE.indication Semantics

The LOCK_STORE indication primitive shall use the following semantics:
LOCK_STORE.indication(Store Transaction Identifier, Store Result Metadata).

Verification Method: T

8.2.1.10.3 When generated

SAVOIR.MMS.STO.0690

LOCK_STORE.indication When generated

The LOCK_STORE.indication primitive shall be passed by the MMS provider to the requesting User Entity in response to the LOCK_STORE.request with Store Result Metadata indicating if the request was executed successfully or not.

Verification Method: T

SAVOIR.MMS.STO.0700

LOCK_STORE.indication When generated failure

When LOCK_STORE.request is unsuccessful, the Store Result Metadata shall provide the reason of the failure.

Verification Method: T

8.2.1.10.4 Effect on receipt

The response of the User Entity to a LOCK_STORE.indication is unspecified.

8.2.1.10.5 Additional constraints

SAVOIR.MMS.STO.0710

LOCK_STORE.indication Store Invalid

The LOCK_STORE.indication Store Result Metadata shall report a failure if Store identified by the Store Identifier does not exist.

Verification Method: T

SAVOIR.MMS.STO.0720

LOCK_STORE.indication Store locked

The LOCK_STORE.indication Store Result Metadata shall report a failure if Store identified by the Store Identifier is locked by another User Entity in any mode.

Rationale: *Locking a Store locked by another User Entity is not possible.*

Comment: *The behaviour of the MMS with respect to the different locking services (BAU, Store, FMS) is let free to implementer.*

Verification Method: T

8.2.1.11 UNLOCK_STORE.request

8.2.1.11.1 Function

SAVOIR.MMS.STO.0730

UNLOCK_STORE.request Function

The UNLOCK_STORE.request primitive shall be passed to the MMS provider to request the unlocking of a Store.

Verification Method: T

8.2.1.11.2 Semantics

SAVOIR.MMS.STO.0740

UNLOCK_STORE.request Semantics

The UNLOCK_STORE.request primitive shall use the following semantics:
UNLOCK_STORE.request(Store Transaction Identifier, Store Identifier, User Entity Identifier).

Verification Method: T

8.2.1.11.3 When generated

SAVOIR.MMS.STO.0750

UNLOCK_STORE.request When generated

The UNLOCK_STORE.request primitive shall be passed to the MMS provider to request the unlocking of the Store identified by the Store Identifier provided as parameter.

Verification Method: T

8.2.1.11.4 Effect on receipt

SAVOIR.MMS.STO.0760

UNLOCK_STORE.request Effect on receipt

Receipt of the UNLOCK_STORE.request primitive shall cause the MMS provider to unlock the Store identified by the Store Identifier provided as parameter.

Verification Method: T

8.2.1.11.5 Additional constraints

8.2.1.12 UNLOCK_STORE.indication

8.2.1.12.1 Function

SAVOIR.MMS.STO.0770

UNLOCK_STORE.indication function

The UNLOCK_STORE.indication primitive shall be used to indicate the outcome of the unlocking of the Store identified by the Store Identifier provided as parameter.

Rationale: The function returns the status of the operation.

Verification Method: T

8.2.1.12.2 Semantics

SAVOIR.MMS.STO.0780

UNLOCK_STORE.indication semantics

The UNLOCK_STORE.indication primitive shall use the following semantics:
UNLOCK_STORE.indication(Store Transaction Identifier, Store Result Metadata).

Verification Method: T

8.2.1.12.3 When generated

SAVOIR.MMS.STO.0790

UNLOCK_STORE.indication when generated

The UNLOCK_STORE.indication primitive shall be passed by the MMS provider to the requesting User Entity in response to the UNLOCK_STORE.request with Store Result Metadata indicating if the request was executed successfully or not.

Verification Method: T

SAVOIR.MMS.STO.0800

UNLOCK_STORE.indication When generated failure

When UNLOCK_STORE.request is unsuccessful, the Store Result Metadata shall provide the reason of the failure.

Comment: The Store remains locked.

Verification Method: T

8.2.1.12.4 Effect on receipt

The response of the User Entity to a UNLOCK_STORE.indication is unspecified.

8.2.1.12.5 Additional constraints

SAVOIR.MMS.STO.0810

UNLOCK_STORE.indication Store Invalid

The UNLOCK_STORE.indication Store Result Metadata shall report a failure if any of the requested Store is not existing in the MMS.

Rationale: Unlocking a Store that is not created is not possible.

Verification Method: T

SAVOIR.MMS.STO.0820

UNLOCK_STORE.indication Store locked by another User Entity

The UNLOCK_STORE.indication Store Result Metadata shall report a failure if the Store to unlock is locked by another User Entity in any mode.

Rationale: Unlocking a Store locked by another User Entity is not possible.

Comment: If found too restrictive, it is possible to implement a User Entity having all the rights on the MMS or being able to impersonate the User Entity that locked the Store.

Verification Method: T

8.2.1.13 ERASE_STORE.request

8.2.1.13.1 Function

SAVOIR.MMS.STO.0830

ERASE_STORE.request Function

The ERASE_STORE.request primitive shall be passed to the MMS provider to request the erase of a Store.

Verification Method: T

8.2.1.13.2 Semantics

SAVOIR.MMS.STO.0840

ERASE_STORE.request Semantics

The ERASE_STORE.request primitive shall use the following semantics:
ERASE_STORE.request(Store Transaction Identifier, Store Identifier).

Verification Method: T

8.2.1.13.3 When generated

SAVOIR.MMS.STO.0850

ERASE_STORE.request When generated

The ERASE_STORE.request primitive shall be passed to the MMS provider to request the erase of the Store identified by the Store Identifier provided as parameter.

Verification Method: T

8.2.1.13.4 Effect on receipt

SAVOIR.MMS.STO.0860

ERASE_STORE.request Effect on receipt

Receipt of the ERASE_STORE.request primitive shall cause the MMS provider to erase the Store identified by the Store Identifier provided as parameter.

Comment: Erase of the store consists in setting the content of all the BAUs that are part of the Store to a default value, e.g. by using ERASE_BAU service.

Verification Method: T

8.2.1.13.5 Additional constraints

8.2.1.14 ERASE_STORE.indication

8.2.1.14.1 Function

SAVOIR.MMS.STO.0870

ERASE_STORE.indication function

The ERASE_STORE.indication primitive shall be used to indicate the outcome of the erasing of the Store identified by the Store Identifier provided as parameter.

Rationale: The function returns the status of the operation.

Verification Method: T

8.2.1.14.2 Semantics

SAVOIR.MMS.STO.0880

ERASE_STORE.indication semantics

The ERASE_STORE.indication primitive shall use the following semantics:
ERASE_STORE.indication(Store Transaction Identifier, Store Result Metadata).

Verification Method: T

8.2.1.14.3 When generated

SAVOIR.MMS.STO.0890

ERASE_STORE.indication when generated

The ERASE_STORE.indication primitive shall be passed by the MMS provider to the requesting User Entity in response to the ERASE_STORE.request with Store Result Metadata indicating if the request was executed successfully or not.

Verification Method: T

SAVOIR.MMS.STO.0900

ERASE_STORE.indication When generated failure

When ERASE_STORE.request is unsuccessful, the Store Result Metadata shall provide the reason of the failure.

Comment: The content of the Store may be in an unknown state if only part of the BAUs have been erased.

Verification Method: T

8.2.1.14.4 Effect on receipt

The response of the User Entity to an ERASE_STORE.indication is unspecified.

8.2.1.14.5 Additional constraints

SAVOIR.MMS.STO.0910

ERASE_STORE.indication Store Invalid

The ERASE_STORE.indication Store Result Metadata shall report a failure if any of the requested Store is not existing in the MMS.

Rationale: Erasing a Store that is not existing is not possible.

Verification Method: T

SAVOIR.MMS.STO.0920

ERASE_STORE.indication Store locked

The ERASE_STORE.indication Store Result Metadata shall report a failure if the Store to erase is locked by another User Entity in any mode.

Rationale: Erasing a Store locked by another User Entity is not possible.

Verification Method: T

8.3 File Management System Service

This section provides requirements related to the FMS service. It includes the definition of the services and the interfaces to those services. Those services expose a common interface to access File Systems selected for the mission to any user of the FMS (User Entity).

8.3.1 Service definition

SAVOIR.MMS.FMS.0100

File create

The FMS shall provide a User Entity the ability to create a File.

Rationale: *The creation of a File provides flexibility in the management of the available data storage area of a File Store.*

Comment: *The creation of the Files provides the capability to dynamically manage the available storage area provided by a File Store. File Stores can also be defined statically through configuration tables.*

Verification Method: T

Parent: SAVOIR-DSS-ORG-350

SAVOIR.MMS.FMS.0110

File open

The FMS shall provide a User Entity the ability to open an existing File.

Rationale: *To identify the File to be manipulated by the User Entity.*

Verification Method: T

Parent: SAVOIR-DSS-ORG-390

SAVOIR.MMS.FMS.0120

File close

The FMS shall provide a User Entity the ability to close an opened File.

Rationale: *To indicate that the File will be no more accessed by the User Entity.*

Verification Method: T

Parent: SAVOIR-DSS-ORG-410

SAVOIR.MMS.FMS.0130

File write

The FMS shall provide a User Entity the ability to write data into an opened File.

Rationale: *To store data into a File.*

Verification Method: T

Parent: SAVOIR-DSS-ORG-310, SAVOIR-DSS-ORG-430

SAVOIR.MMS.FMS.0140

File read

The FMS shall provide a User Entity the ability to read data from an opened File.

Rationale: *To retrieve data stored into a File.*

Verification Method: T

Parent: SAVOIR-DSS-ORG-440

SAVOIR.MMS.FMS.0150

File seek

The FMS shall provide a User Entity the ability to identify from which location of an opened File the data will be read.

Rationale: *To allow direct access into a File.*

Verification Method: T

Parent: SAVOIR-DSS-ORG-450, SAVOIR-DSS-ORG-460

SAVOIR.MMS.FMS.0160

File Status Get

The FMS shall provide a User Entity the ability to retrieve all the information related to an existing File.

Rationale: *To retrieve information related to a file as its size, its maximum size, etc.*

Verification Method: T

Parent: SAVOIR-DSS-ORG-482

SAVOIR.MMS.FMS.0170

File Delete

The FMS shall provide a User Entity the ability to delete an existing File.

Rationale: *To free the data storage area used by the File.*

Verification Method: T

Parent: SAVOIR-DSS-ORG-380

SAVOIR.MMS.FMS.0180

File Copy

The FMS shall provide a User Entity the ability to copy an existing File.

Rationale: *To duplicate the content of a File.*

Verification Method: T

Parent: SAVOIR-DSS-ORG-580, SAVOIR-DSS-ORG-590

SAVOIR.MMS.FMS.0190

File Move

The FMS shall provide a User Entity the ability to move an existing file.

Rationale: *To move a File to another Directory of the same File Store or to another File Store.*

Comment: *File Move can be implemented by a File Copy followed by a File Delete. File Move can be used to rename a File.*

Verification Method: T

Parent: SAVOIR-DSS-ORG-650, SAVOIR-DSS-ORG-510

SAVOIR.MMS.FMS.0200

Directory Create

The FMS shall provide a User Entity the ability to create a Directory.

Rationale: *To create a hierarchical organisation of Files.*

Verification Method: T

Parent: SAVOIR-DSS-ORG-230

SAVOIR.MMS.FMS.0210

Directory Delete

The FMS shall provide a User Entity the ability to delete an existing Directory.

Rationale: *To update the hierarchical organisation.*

Verification Method: T

Parent: SAVOIR-DSS-ORG-240

SAVOIR.MMS.FMS.0220

Directory Rename

The FMS shall provide a User Entity the ability to delete an existing Directory.

Rationale: *To update the hierarchical organisation.*

Verification Method: T

Parent: SAVOIR-DSS-ORG-250

SAVOIR.MMS.FMS.0221

Directory Attribute Set

The FMS shall provide a User Entity the ability to set attributes defined for a Directory.

Rationale: *To set additional information related to a Directory.*

Comment: *Some attributes may be managed by the FMS and cannot be set by the User Entity.*

Verification Method: T

Parent: SAVOIR-DSS-ORG-290

SAVOIR.MMS.FMS.0222

Directory Attribute Get

The FMS shall provide a User Entity the ability to retrieve the attributes associates to an existing Directory.

Rationale: *To retrieve specific information on a Directory.*

Verification Method: T

Parent: SAVOIR-DSS-ORG-300

SAVOIR.MMS.FMS.0230

File Lock

The FMS shall provide a User Entity the ability to lock the access to an existing File.

Rationale: To protect the File against concurrent accesses.

Verification Method: T

Parent: SAVOIR-DSS-ORG-700

SAVOIR.MMS.FMS.0240

File Unlock

The FMS shall provide a User Entity the ability to unlock the access to an existing File.

Rationale: To release the lock on a File.

Verification Method: T

Parent: SAVOIR-DSS-ORG-700

SAVOIR.MMS.FMS.0250

File Locked List

The FMS shall provide a User Entity the ability to retrieve the list of the locked Files.

Rationale: To get indication on the Files that are currently locked by User Entities.

Verification Method: T

Parent: SAVOIR-DSS-ORG-700

SAVOIR.MMS.FMS.0260

File Find

The FMS shall provide a User Entity the ability to find Files located in a Directory.

Rationale: To list the files that are currently existing in a Directory.

Verification Method: T

Parent: SAVOIR-DSS-ORG-260

SAVOIR.MMS.FMS.0270

File Map

The FMS shall provide a User Entity the ability to map an interface to an existing File.

Rationale: To automatically store the data received on an interface to a File.

Verification Method: T

Parent: SAVOIR-DSS-ORG-530

SAVOIR.MMS.FMS.0280

File Attribute Set

The FMS shall provide a User Entity the ability to set attributes defined for a File.

Rationale: *To set additional information related to a File.*

Comment: *Some attributes may be managed by the FMS and cannot be set by the User Entity.*

Verification Method: T

Parent: SAVOIR-DSS-ORG-480

SAVOIR.MMS.FMS.0290

File Attribute Get

The FMS shall provide a User Entity the ability to retrieve the attributes associates to an existing File.

Rationale: *To retrieve specific information on a File.*

Verification Method: T

Parent: SAVOIR-DSS-ORG-490

SAVOIR.MMS.FMS.0300

File Synchronisation

The FMS shall provide a User Entity the ability to ensure that all the data are currently stored into an opened File.

Rationale: *To avoid any data loss.*

Comment: *This service ensures that all the data that could be located in temporary buffers within the FMS are stored into the File.*

Verification Method: T

SAVOIR.MMS.FMS.0310

File Shrink

The FMS shall provide a User Entity the ability to reduce the size of an existing File.

Rationale: *To reduce the size of an existing File.*

Verification Method: T

SAVOIR.MMS.FMS.0320

File Checksum Get

The FMS shall provide a User Entity the ability to get the checksum of a File.

Rationale: *To identify any corruption of the data contained in a File.*

Verification Method: T

Parent: SAVOIR-DSS-ORG-500

SAVOIR.MMS.FMS.0330

File Copy Suspend

The FMS shall provide a User Entity the ability to suspend the copy of a File.

Rationale: *Copying a File can be a long operation that may need to be suspended.*

Verification Method: T

Parent: SAVOIR-DSS-ORG-620, SAVOIR-DSS-ORG-672

SAVOIR.MMS.FMS.0340

File Copy Resume

The FMS shall provide a User Entity the ability to resume the copy of a File that has been previously suspended.

Rationale: *To resume the operation in order to complete it.*

Verification Method: T

Parent: SAVOIR-DSS-ORG-630, SAVOIR-DSS-ORG-673

SAVOIR.MMS.FMS.0350

File Copy Abort

The FMS shall provide a User Entity the ability to abort a File Copy operation.

Rationale: *To cancel the copy of a File.*

Comment: *If the File Copy operation is aborted before completion, the destination File is deleted.*

Verification Method: T

Parent: SAVOIR-DSS-ORG-640, SAVOIR-DSS-ORG-674

SAVOIR.MMS.FMS.0360

File Copy Status Get

The FMS shall provide a User Entity the ability to get the current status of a File Copy operation.

Rationale: To get information on the current status of the operation.

Verification Method: T

Parent: SAVOIR-DSS-ORG-610, SAVOIR-DSS-ORG-671

SAVOIR.MMS.FMS.0370

File Copy Event reception

The FMS shall provide a User Entity the ability to receive events related to the progress of a File Copy operations.

Rationale: To follow the progression of a File Copy operation.

Verification Method: T

SAVOIR.MMS.FMS.0380

File Event Register

The FMS shall provide a User Entity the ability to register to FMS Events.

Rationale: To receive events on particular activities related to the FMS.

Verification Method: T

Parent: SAVOIR-DSS-ORG-680

SAVOIR.MMS.FMS.0390

File Event Unregister

The FMS shall provide a User Entity the ability to unregister to FMS Events.

Rationale: To stop receiving events on particular activities related to the FMS.

Verification Method: T

Parent: SAVOIR-DSS-ORG-680

SAVOIR.MMS.FMS.0400

File Event reception

The FMS shall provide a User Entity the ability to receive FMS Events.

Rationale: To receive information of registered FMS Events.

Verification Method: T

Parent: SAVOIR-DSS-ORG-680

SAVOIR.MMS.FMS.0401

File BAU List Get

The FMS shall provide a User Entity the ability to retrieve the BAU list that is used by the FMS to store the data stored.

Rationale: *To identify the BAUs that are used to store the data of a file (e.g. to perform dump operations).*

Verification Method: T

Parent: SAVOIR-DSS-ORG-330

SAVOIR.MMS.FMS.0402

File Defragmentation

The FMS shall provide a User Entity the ability to defragment the files.

Rationale: *To increase the reliability of the File Management System.*

Comment: *Defragmentaton is dependent on the way the File System organizes the data stored in files that may be linked to Data Storage and mission constraints. This maybe a complex and specific operation.*

Verification Method: T

Parent: SAVOIR-DSS-ORG-730

SAVOIR.MMS.FMS.0403

File Map Split

The FMS shall provide a User Entity the ability to force the segmentation of a file mapped to an input interface.

Rationale: *To give the possibility to split received data at a certain time (e.g. before critical operations on the spacecraft.*

Verification Method: T

Parent: SAVOIR-DSS-ORG-570

8.3.2 Service parameters

SAVOIR.MMS.FMS.0410

Absolute File Offset

The Absolute File Offset parameter shall indicate the desired octet offset from the start of that file at which data is to be read from or written to. It can be either positive or null.

Comment: Negative value are prohibited, since it would intentionally place the file's descriptor outside the boundaries of the file.

Verification Method: RoD

SAVOIR.MMS.FMS.0420

Attribute

The Attribute parameter is identified by an Attribute Identifier and defined by an Attribute Type, an Attribute Size and an Attribute Value.

Rationale: Attributes makes possible to attach specific data to an entry (file or directory) of the FMS.

Verification Method: RoD

Parent: SAVOIR-DSS-ORG-280, SAVOIR-DSS-ORG-470

SAVOIR.MMS.FMS.0430

Attribute Identifier

The Attribute Identifier parameter shall identify one specific Attribute of a file.

Rationale: To uniquely identify an Attribute associated to a file.

Comment: Identification is unique at the level of each file.

Verification Method: RoD

Parent: SAVOIR-DSS-ORG-280, SAVOIR-DSS-ORG-470

SAVOIR.MMS.FMS.0440

Attribute List

The Attribute List parameter shall identify a list of Attribute.

Rationale: To retrieve all attributes associated to a file.

Verification Method: RoD

SAVOIR.MMS.FMS.0450

Attribute Size

The Attribute Size parameter shall be the size of the Attribute Value.

Rationale: *To manage value of attributes with variable size.*

Comment: *The value of Attribute Size is dependent on the File System. It can be independently set for each Attribute or fixed explicitly or implicitly (e.g. when no String or Binary Types are used).*

Verification Method: RoD

Parent: SAVOIR-DSS-ORG-280, SAVOIR-DSS-ORG-470

SAVOIR.MMS.FMS.0460

Attribute Type

The Attribute Type parameter shall correspond to the type of an Attribute Value currently associated or to be associated to an Attribute, i.e.:

- INT64: an integer represented over 64 bits,
- DOUBLE: a floating-point value, represented over 64 bits,
- STRING: an array of ASCII characters of 'Attribute Size' maximum length,
- BINARY: an array of bytes, of 'Attribute Size' maximum size.

Rationale: *The Attribute Type indicates how the Attribute Value is to be interpreted.*

Comment: *STRING and BINARY attributes may not be natively supported by all File Systems.*

Verification Method: RoD

Parent: SAVOIR-DSS-ORG-280, SAVOIR-DSS-ORG-470

SAVOIR.MMS.FMS.0470

Attribute Value

The Attribute Value parameter shall be the value currently associated or to be associated to an Attribute.

Verification Method: RoD

Parent: SAVOIR-DSS-ORG-280, SAVOIR-DSS-ORG-470

SAVOIR.MMS.FMS.0480

Base Directory Full Path

The Base Directory Full Path parameter shall be the Directory Full Path identifying the directory from which the search operation is to be performed.

Rationale: *To specify where the search shall start in the directory hierarchy.*

Comment: *The Directory Full Path is defined in section 8.1.2.*

Verification Method: RoD

SAVOIR.MMS.FMS.0490

Destination File Full Path

The Destination File Full Path parameter shall be the File Full Path of the copied or moved file.

Verification Method: RoD

SAVOIR.MMS.FMS.0500

Directory Full Path

The Directory Full Path parameter shall uniquely identify a directory at FMS level by gathering its Directory Path and Directory Name

Comment: *There is no Directory Path for the Root Directory.*

Verification Method: RoD

SAVOIR.MMS.FMS.0510

Directory Listing

The Directory Listing parameter shall be used to list the files and subdirectories contained within a directory, providing at least the following information:
- Directory Full Path (only once),
- And for each child entity:
  - Name of the child entity (i.e. Directory Name or File Name),
  - Type of the child entity (among FILE and DIRECTORY).

Comment: *As the Directory Full Path is common, it is not repeated for each listed child entity.*

Verification Method: RoD

Parent: SAVOIR-DSS-ORG-270

SAVOIR.MMS.FMS.0520

Directory Name

The Directory Name parameter shall be a string or integer used to uniquely identify a directory within its parent.

Rationale: *Directory Name is required to identify a directory.*

Verification Method: RoD

SAVOIR.MMS.FMS.0530

Root Directory Name

A specific Directory Name identified as Root Directory Name shall identify the top-level directory of a File Store.

Rationale: *Root Directory Name identifies the starting point of the directory hierarchy of a File Store.*

Comment: *When a File System is used to manage several File Store, the Root Directory Name shall include the identification of the File Store.*

Verification Method: RoD

SAVOIR.MMS.FMS.0540

File Store Root

Each File Store shall be identified by a unique Directory Name giving access to its root directory.

Comment: *The Directory Name referring to a Store must be unique under the FMS root.*

Verification Method: RoD

SAVOIR.MMS.FMS.0550

Directory Path

The Directory Path parameter shall be an identifier used to locate a directory within the FMS organization.

Rationale: *Directory Path represents the directory hierarchy to follow to access the directory from its Root Directory.*

Comment: *Usually, the identifier is a string at FMS level in order to harmonize accesses to different File Systems. At File System level, the Directory Path could be different, e.g. a simple integer value. The FMS shall be able to translate the Directory Path into a format that is understood by underlying File Systems.*

Verification Method: RoD

SAVOIR.MMS.FMS.0560

Directory Path Type

The Directory hierarchy shall be a string made of Directory Name(s) and separator(s).

Rationale: *String is able to cope with Directory Path representation of different File Systems.*

Comment: *FMS Separator is part of the File System characteristics. A File System may not need to define a separator if fixed length names are used for Directory Names (e.g. 8 characters).*

Verification Method: RoD

SAVOIR.MMS.FMS.0570

Directory Attributes

The Directory Attributes parameter shall be the list of Attributes attached to a directory.

Comment: *A FMS implementation may define some mandatory attributes required by user entities (e.g. starting date of recording, data source identification, etc.).*
The attributes defined are not necessarily the same for all directories.

Verification Method: RoD

SAVOIR.MMS.FMS.0580

File Access Type

The File Access Type defined the type of access authorised on a file when opened:
- Read-Only: the User Entity can only read from the file;
- Read-Write: the User Entity can read from and write to the file;
- Append: the User Entity can only write at the end of file (no seek allowed).

Rationale: *To restrict the access to a file when opened.*

Comment: *The restriction is provided by the User Entity when opening a file. Other restriction may apply, e.g. at BAU level.*

Verification Method: RoD

SAVOIR.MMS.FMS.0590

File Action When Full

The File Action When Full Type shall define the action to be performed when a file becomes full:

- Close File: when the file gets full, it is closed, and no more writing to the file is possible (unless re-opening it);
- Close File & Create Next File: when the file gets full, it is closed, and a new file is created (if not existing) to continue data storage without losing any data.

Rationale: To identify the action to be automatically performed by the FMS when the maximum size of a file is reached.

Verification Method: RoD

SAVOIR.MMS.FMS.0600

File Attributes

The File Attributes parameter shall be the list of Attributes attached to a file, complementary to its content.

Comment: A FMS can define attributes for its own use i.e. that must not be modified by User Entities.

Comment: A FMS implementation may define some mandatory attributes required by User Entities (for example to manage a download priority, etc.). Moreover, a User entity can declare a limited number of custom attributes. The attributes defined are not necessarily the same for all files.

Verification Method: RoD

SAVOIR.MMS.FMS.0610

File Checksum

The File Checksum parameter shall be the computed checksum of a file.

Rationale: The checksum is used to verify data integrity and detect errors (alteration / bit flip) which may occur during transmission.

Comment: It consists in a small-size datum obtained by computation on file content.

Verification Method: RoD

SAVOIR.MMS.FMS.0620

File Checksum function

The Checksum function shall be defined at mission design level and identified at FMS level.

Rationale: There are many ways to compute a checksum. Therefore, the method or algorithm has to be fixed so there is no risk of ambiguity and systematic rejection of data that are actually valid.

Verification Method: RoD

SAVOIR.MMS.FMS.0630

File Closed

The File Closed parameter shall indicate if the file was closed or not by a write operation (WRITE_TO_FILE.request).

Verification Method: RoD

SAVOIR.MMS.FMS.0640

File Copy Identifier

The Copy Identifier parameter shall be used to logically identify an initiated file copy operation managed by a FMS.

Verification Method: RoD

SAVOIR.MMS.FMS.0650

File Copy Status

The Copy Status parameter shall be used to report information about an initiated file copy, including:

- Copy Identifier – identifies the copy operation.
- State – indicates the current state of the copy operation among STARTED, SUSPENDED, COMPLETED, and ABORTED.
- Overall Progress – indicates the progression percentage (copied / source file's size * 100).

Comment: If the source file's size increases during the copy operation (esp. new data appended by another User Entity), the Overall Progress may decrease in time. The term "on-going" further appearing in requirements, refers to a copy operation in STARTED or SUSPENDED state.

SAVOIR.MMS.FMS.0651

File Copy State transitions

The File Copy States transitions parameter shall be:

- STARTED to COMPLETED when end of Source file is reached.
- STARTED to SUSPENDED on SUSPEND_FILE_COPY request
- SUSPENDED to STARTED on RESUME_FILE_COPY request
- STARTED to ABORTED on ABORT_FILE_COPY request
- SUSPENDED to ABORTED on ABORT_FILE_COPY request

Comment: Transitions are illustrated in Figure 4.

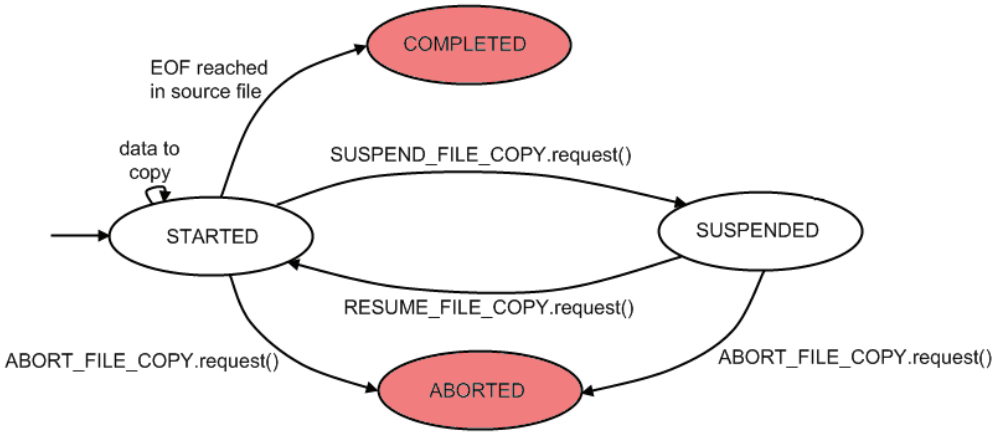


Figure 4: File Copy State Machine

Verification Method: RoD

SAVOIR.MMS.FMS.0660

File Descriptor

The File Descriptor parameter shall be used to logically identify an opened file among all the File Stores managed by a FMS.

Rationale: File Descriptors are used to work on files that have been opened.

Verification Method: RoD

SAVOIR.MMS.FMS.0670

File Full Path

The File Full Path parameter shall uniquely identify a file at FMS level by gathering its File Path and File Name.

Rationale: The File Full Path is used to uniquely identify a file.

Verification Method: RoD

Parent: SAVOIR-DSS-ORG-340

SAVOIR.MMS.FMS.0680

File Mapping Type

The File Mapping Type shall identify the type of mapping applied to a file:

- Exclusive: all data received on a single input interface has to be stored into the file,
- Protocol Address: all data received on a set of input interfaces with a given Data Link layer address has to be stored into the file,
- Packet APID: all packets received on a set of input interfaces with a given APID have to be stored into the file,
- Custom Packet Field: all packets received on a set of input interfaces with a given packet field value have to be stored into the file,
- Packet PUS Service: all PUS packets received on a set of input interfaces with a given service have to be stored into the file,
- Packet PUS Service & Subservice: all PUS packets received on a set of input interfaces with a given (service, subservice) have to be stored into the file,
- Packet APID, PUS Service & Subservice: all PUS packets received on a set of input interfaces with a given triplet (APID, service, subservice) have to be stored into the file.

Rationale: To identify how data are stored within dedicated files.

Verification Method: RoD, T

SAVOIR.MMS.FMS.0690

File Mapping Configuration

The File Mapping Configuration shall identify the information required to perform the mapping of data to a file. This includes:

- Protocol Address: identifies the Data Link layer address (e.g. SpaceWire address, 1553 address and sub-address, etc.) when Protocol Address mapping is requested.
- Packet APID: identifies the Packet Application Process Identifier when either Packet APID or Packet APID, PUS Service & Subservice mapping is requested.
- Custom Packet Field: identifies the characteristics of the custom field when Custom Packet Field mapping is requested:
    - Offset: identifies the field offset from the beginning of the Packet
    - Value: identifies the reference value used for field comparison
    - Size: identifies the field size
    - Comparison operator: identifies the operator (e.g. >, <, =) to be applied to check packet field matching against the reference value.
- Packet PUS Service: identifies the PUS service when either Packet PUS Service, Packet PUS Service & Subservice or Packet APID, PUS Service & Subservice mapping is requested.
- Packet PUS Subservice: identifies the PUS subservice when either Packet PUS Service &Subservice or Packet APID, PUS Service & Subservice mapping is requested.

Rationale: *To identify the information that need to be extracted in order to perform the mapping*.

Verification Method: RoD

SAVOIR.MMS.FMS.0700

File Name

The File Name parameter shall be a string or integer used to uniquely identify a file within a directory (in a File Store).

Verification Method: RoD

SAVOIR.MMS.FMS.0710

File Lock Type

The File Lock Type parameter shall specify a lock action on a file. Possible values are:
- Exclusive_Read_Only: The lock-owner can only read from the file. Other user entities cannot read from or write to the file;
- Read_Only: The lock-owner can only read from the file. Other user entities can only read from the file (i.e. everyone can read);
- Exclusive_Access: The lock-owner can read from and write to the file. Other user entities cannot read from or write to the file;
- Single_Writer: The lock-owner can read from and write to the file. Other user entities can only read from the file.

Comment: No lock applied means all entities can read from and write to the file.

Parent: SAVOIR-DSS-ORG-700, SAVOIR-DSS-ORG-710, SAVOIR-DSS-ORG-720

Verification Method: RoD

SAVOIR.MMS.FMS.0720

File Opening Criteria

The File Opening Criteria parameter shall indicate the criteria to be applied to a file when opened:
- File Access Type: define the authorisation of User Entity opening the file.
- Create mode flag: if specified and the file does not exist it will be created.
- File Lock Type (optionally): if specified, the file is locked atomically when opened.
- File Action When Full (optionally): Applicable only when Access Type is “Append”.
- File Packet Storage flag (optionally): Applicable only when Access Type is “Append”, when specified, the file is used exclusively to store complete Packets (i.e. that shall not be split), available only to File Stores supporting the definition of Maximum Size.
- Other specific fields may be defined depending on the mission

Rationale: To identify how the file has to be managed at the time it is opened.

Comment: “Full File Action” and “Packet Storage” criteria are not considered when Access Type is different from “Append”. Implementing the “Full File Action” criterion requires it to be specified when “Append” Access Type is requested.

Verification Method: RoD

SAVOIR.MMS.FMS.0730

File Path

The File Path parameter shall be an identifier used to locate a file within the FMS organization.

Comment: It represents the directory hierarchy to follow from the FMS root to access the file.

Verification Method: RoD

SAVOIR.MMS.FMS.0740

File Result Metadata

The Result Metadata parameter shall be used to provide information generated by the FMS provider to the User Entity regarding successful or failed result of a primitive.

Comment: As an example it could indicate that the specified request cannot be serviced within the managed timeout period or the FS or BAS layers are malfunctioning.

Verification Method: RoD

SAVOIR.MMS.FMS.0750

File Segment

The File Segment parameter shall be the data segment read from a file or to be written to a file.

Verification Method: RoD

SAVOIR.MMS.FMS.0760

File Segment Length

The File Segment Length parameter shall indicate the length in octets of the data to be read or written for a request, and the effective length or data read or written in an indication primitive.

Verification Method: RoD

SAVOIR.MMS.FMS.0770

File Selection Pattern

The File Selection Pattern parameter shall be used to specify the pattern to be applied on any information/characteristic available in File Status in order to search for specific file(s).

Comment: This may include wildcards and string pattern matching expressions applied to strings. Complex pattern can be defined to support logical operators between the attributes of the files. At File System level, the format of wildcards and string pattern matching expressions is implementation dependent.

Verification Method: RoD

SAVOIR.MMS.FMS.0780

File Size

The File Size parameter shall indicate the number of data octets forming the file content.

Verification Method: RoD

SAVOIR.MMS.FMS.0790

File Size Allocation

The File Size Allocation parameter shall indicate if the allocation of the data storage area required to store data in the File shall be done at the time of creation of the File.

Rationale: To pre-allocate the storage area required to store the content of a File at its creation.

Comment: The allocation can only be done if the Maximum File Size is provided at the creation of the File.

Verification Method: RoD

SAVOIR.MMS.FMS.0800

File Status

The File Status parameter shall be used to list the information related to the file current state:

- File Name identifies the file.
- Creation Time (optionally) is the time at which the file was created.
- Last Read Time (optionally) is the time at which the file was last read.
- Last Write Time (optionally) is the time at which the file was last written to.
- Lock Identifier (optionally) identifies a lock on the file (if locked).
- Lock Type (optionally) identifies the type of lock (if locked).
- Lock Owner (optionally) identifies the User Entity owning the lock (if locked).
- Size is the current size of content (data) in octets.
- Allocated Size (optionally) is the effective size occupied by the file within the Store, in octets. It is always a multiple of the Store BAU size.
- Maximum Size (optionally) indicates the size that cannot be exceeded by content if specified at creation.
- Mapping Criteria (optionally) identifies all mapping criteria currently associated to the file.
- Opened indicates whether the file is currently open by at least one User Entity or not.

Comment: *Some of the information may be implemented as attributes, but not directly modifiable by User Entities.*

Verification Method: RoD

SAVOIR.MMS.FMS.0810

FMS Event

The FMS Event parameter shall be used to list the characteristics of a reported event:

- Type identifies the kind of operation reported (creation, auto-closing, deletion).
- Store identifier
- Source File Path is the full path to the source file at the origin of this event.
- Destination File Path (optional) is the full path to the destination file at the origin of this event (for operations involving a destination file identification).
- Event Sequence Counter is a counter incremented for each generated event (for time ordering).

Comment: *It is possible to implement other types of event. Those given in the above requirement are mandatory. A lock / unlock event can be implemented so User Entity waiting for a file access are notified.*

Verification Method: RoD

Parent: SAVOIR-DSS-ORG-690

SAVOIR.MMS.FMS.0820

Force

The Force parameter shall be used to force deletion of a currently open file (whatever the User Entity).

Verification Method: RoD

SAVOIR.MMS.FMS.0830

Found Files List

The Found Files List parameter shall contain the list of the full path of all files that matched a specific File Selection Pattern

Verification Method: RoD

SAVOIR.MMS.FMS.0840

Locked Files List

The Locked Files List parameter references all currently locked files that shall be maintain and referred as Locked Files List parameter.

Verification Method: RoD

SAVOIR.MMS.FMS.0850

Locked Files List information

The Locked Files List parameter shall contain lock characteristics of all locked files:
- File Path identifies the full path to the file upon which the lock is applied;
- Lock Identifier identifies a lock on a file;
- Lock Type identifies the type of lock applied;
- Lock Owner identifies the User Entity owning the lock.

- *Verification Method: RoD*

SAVOIR.MMS.FMS.0860

Lock Identifier

The Lock identifier parameter shall be used to logically identify a lock applied on a file.

Verification Method: RoD

SAVOIR.MMS.FMS.0870

Mapping Criteria

The Mapping Criteria shall indicate the criteria to be associated to a file for configuration of autonomous data storage:

- File Mapping Type: identifies how data have to be associated to a file.
- Interface Identifier: identifies the input interface concerned by the mapping.
- File Mapping Configuration: identifies the filtering configuration.
- Maximum File Size: identifies the amount of data to stop into mapped files.
- Continuous Storage: if specified, the storage must continue in new files (successively created, and up to the File Store capacity) when mapped file gets full.
- Proxy Mapping: if specified, the input interfaces used for acquisition must be those of the remote MM (only applicable when the File is located into a remote File Store).

Comment: *The Proxy Mapping parameter allows supporting the two following scenarios:*
- When Proxy Mapping is False, data is collected from OBC inputs and stored into a remote MM File Store (via a Remote File Management Protocol). In this case the OBC FMS manages the acquisition and commands the remote FMS to store data (performing the file opening and writing operations).
- When Proxy Mapping is True, data is collected from the remote MM inputs and stored into the remote MM File Store. In this case the OBC FMS acts only as a proxy and transfers the mapping request to the remote FMS.

Parent: SAVOIR-DSS-ORG-560

Verification Method: RoD

SAVOIR.MMS.FMS.0880

Maximum File Size

The Maximum File Size parameter shall indicate the size in octets that cannot be exceeded by file content.

Rationale: *To limit the data written within the file.*

Verification Method: RoD

SAVOIR.MMS.FMS.0890

Monitored Events

The Monitored Events parameter shall indicate the event or group of events to be registered with a directory or unregistered from it, among:

- FILE_CREATE: report of any file creation, either by direct request from a User Entity or indirectly performed by the FMS (e.g. in the frame of a file copy, autonomous storage, etc.).
- FILE_AUTOCLOSE: report of any file closing by the FMS as the result of a Full File Action (direct file closing by a User Entity is excluded).
- FILE_DELETE: report of any file deletion, either by direct request from a User Entity or indirectly performed by the FMS (e.g. in the frame of a file move).
- Any Combination of the above events (including the three ones).

Comment: File renaming corresponds to a move file operation and is thus covered by FILE_CREATE and FILE_DELETE events. An event is generated at completion of the action it gives the status of.

Verification Method: RoD

SAVOIR.MMS.FMS.0900

New Directory Name

The New Directory Name parameter shall be the resulting Directory Name once renamed.

Verification Method: RoD

SAVOIR.MMS.FMS.0910

Old Directory Full Path

The Old Directory Full Path parameter shall be the Directory Full Path identifying the directory to be renamed.

Verification Method: RoD

SAVOIR.MMS.FMS.0920

Recursive

The Recursive parameter shall specify that the registered event(s) also have to be monitored in all sub-directories of the specified directory.

Comment: The recursive monitoring is dynamic, which means that any sub-directory created after the event(s) registration is automatically monitored.

Verification Method: RoD

SAVOIR.MMS.FMS.0930

Relative File Offset

The Relative File Offset parameter shall indicate the desired octet offset from a reference position within a file (start of file, current position, end of file) at which data is to be read from or written to. It can be either positive, negative or null.

Verification Method: RoD

Verification Method: RoD

SAVOIR.MMS.FMS.0940

Resulting File Offset

The Resulting File Offset parameter shall indicate the byte offset from the beginning of file (i.e. new value of File Descriptor's current position). It can be either positive or null.

Comment: There is a single offset (current position) per File Descriptor used for both read and write operations.

Verification Method: RoD

SAVOIR.MMS.FMS.0950

Source File Full Path

The Source File Full Path parameter shall be the File Full Path of the file to be copied or moved.

Verification Method: RoD

SAVOIR.MMS.FMS.0960

Transaction Identifier

The Transaction Identifier parameter shall be a value, assigned by the invoking User Entity, which is subsequently used to associate indication primitives with the causal request primitives.

Rationale: To correlate all indications and confirmations with the originating service request.

Verification Method: RoD

SAVOIR.MMS.FMS.0970

Transaction Identifier unicity

The Transaction Identifier shall be unique within the invoking User Entity.

Comment: This is the responsibility of the User Entity to provided unique identifiers. The FMS can complement this identifier to make it unique within the system.

Verification Method: RoD

SAVOIR.MMS.FMS.0980

Whence

The Whence parameter shall specify the reference position to be considered within a file among:

- SEEK_SET: start of file,
- SEEK_END: end of file (highest written position),
- SEEK_CUR: File Descriptor's current position.

- *Verification Method: RoD*

8.3.3 Service interface

SAVOIR.MMS.FMS.0990

File Management System primitives

The following primitives defined in [AD06] shall be supported:

- CREATE_FILE.request, CREATE_FILE.indication,
- OPEN_FILE.request, OPEN_FILE.indication,
- MAP_FILE.request, MAP_FILE.indication,
- CLOSE_FILE.request, CLOSE_FILE.indication,
- WRITE_TO_FILE.request, WRITE_TO_FILE.indication,
- READ_FROM_FILE.request, READ_FROM_FILE.indication,
- SEEK_FILE.request, SEEK_FILE.indication,
- SHRINK_FILE.request, SHRINK_FILE.indication,
- GET_FILE_STATUS.request, GET_FILE_STATUS.indication,
- SET_FILE_ATTRIBUTE.request, SET_FILE_ATTRIBUTE.indication,
- GET_FILE_ATTRIBUTES.request, GET_FILE_ATTRIBUTES.indication,
- DELETE_FILE.request, DELETE_FILE.indication,
- COPY_FILE.request, COPY_FILE.indication,
- MOVE_FILE.request, MOVE_FILE.indication,
- CREATE_DIR.request, CREATE_DIR.indication,
- LIST_DIR.request, LIST_DIR.indication,
- DELETE_DIR.request, DELETE_DIR.indication,
- RENAME_DIR.request, RENAME_DIR.indication,
- LOCK_FILE.request, LOCK_FILE.indication,
- UNLOCK_FILE.request, UNLOCK_FILE.indication,
- LIST_LOCKED_FILES.request, LIST_LOCKED_FILES.indication,
- FIND_FILES.request, FIND_FILES.indication.
- FORCE_FILE_SYNCH.request, FORCE_FILE_SYNCH.indication,
- GET_FILE_CHECKSUM.request, GET_FILE_CHECKSUM.indication,
- SUSPEND_FILE_COPY.request, SUSPEND_FILE_COPY.indication,
- RESUME_FILE_COPY.request, RESUME_FILE_COPY.indication,
- ABORT_FILE_COPY.request, ABORT_FILE_COPY.indication,
- GET_FILE_COPY_STATUS.request, GET_FILE_COPY_STATUS.indication,
- FILE_COPY_EVENT.indication,
- REGISTER_FMS_EVENT.request, REGISTER_FMS_EVENT.indication,
- UNREGISTER_FMS_EVENT.request, UNREGISTER_FMS_EVENT.indication,
- FMS_EVENT.indication,
- SET_DIR_ATTRIBUTE.request, SET_DIR_ATTRIBUTE.indication,
- GET_DIR_ATTRIBUTES.request, GET_DIR_ATTRIBUTES.indication,
- GET_BAU_LIST.request, GET_BAU_LIST.indication,
- FILE_DEFRAGMENTATION.request, FILE_DEFRAGMENTATION.indication
- MAP_FILE_SPLIT.request, MAP_FILE_SPLIT.indication

Rationale: *These are the services required to manage Files and Directories.*

Comment: *The list of services can be reduced to match the mission needs, e.g. directory management services may be tailored out.*

Verification Method: I

SAVOIR.MMS.FMS.1000

File Management System parameters

The services shall exclusively use the parameters with the meaning defined in the present specification.

Rationale: *To ensure consistency of implementations.*

Comment: *The meaning of the parameters of the FMS is specified in 8.3.2.*

Verification Method: RoD

8.3.3.1 CREATE_FILE.request

8.3.3.1.1 Function

SAVOIR.MMS.FMS.1010

CREATE_FILE.request Function

The CREATE_FILE.request primitive shall be passed to the FS provider to request the creation of a new file in a File Store.

Rationale: *To create a File that will be later used to store data.*

Verification Method: T

8.3.3.1.2 Semantics

SAVOIR.MMS.FMS.1020

CREATE_FILE.request Semantics

The CREATE_FILE.request primitive shall use the following semantics:
CREATE_FILE.request (File Transaction Identifier, File Full Path, Maximum File Size (optional), File Size Allocation (optional))

Verification Method: T

Parent: SAVOIR-DSS-ORG-370

8.3.3.1.3 When generated

SAVOIR.MMS.FMS.1030

CREATE_FILE.request When generated

The CREATE_FILE.request primitive shall be passed to the FS provider to request the creation of the specified file. Optionally, it may be requested to limit its maximum size and allocate the full capacity at creation.

Verification Method: T

8.3.3.1.4 Effect on receipt

SAVOIR.MMS.FMS.1040

CREATE_FILE.request effect on receipt

The CREATE_FILE.request primitive shall cause the FS provider to create the specified file.

Comment: If File Size Allocation is set, the FS creates the File with the Maximum File Size (i.e. allocate all necessary BAUs to store the specified amount of data).

Verification Method: T

8.3.3.1.5 Additional constraints

SAVOIR.MMS.FMS.1050

CREATE_FILE.request File unicity

In CREATE_FILE.request primitive, the File Name of the File Full Path parameter shall be unique within the parent directory.

Verification Method: T

SAVOIR.MMS.FMS.1060

CREATE_FILE.request File Size limit

In CREATE_FILE.request primitive, the Maximum File Size parameter shall be lower than or equal to the limit of the File Store FS.

Comment: Anyway, the file size could never exceed the FS limit (which may be different per File Store).

Verification Method: T

SAVOIR.MMS.FMS.1070

CREATE_FILE.request File Full Path validity

In CREATE_FILE.request primitive, the File Full Path parameter shall only contain authorized characters (including separators) when string identifiers are used and integers in the allowed range when integer identifiers are used.

Verification Method: T

SAVOIR.MMS.FMS.1080

CREATE_FILE.request Directory existence

In CREATE_FILE.request primitive, the File Path of the File Full Path parameter shall refer to an existing directory.

Comment: The primitive does not automatically create inexistent directories appearing in the path.

Verification Method: T

8.3.3.2 CREATE_FILE.indication

8.3.3.2.1 Function

SAVOIR.MMS.FMS.1090

CREATE_FILE.indication Function

The CREATE_FILE.indication primitive shall be used to pass the outcome of creating a file to the User Entity.

Verification Method: T

8.3.3.2.2 Semantics

SAVOIR.MMS.FMS.1100

CREATE_FILE.indication Semantics

The CREATE_FILE.indication primitive shall use the following semantics:
CREATE_FILE.indication (File Transaction Identifier, File Result Metadata)

Verification Method: T

8.3.3.2.3 When generated

SAVOIR.MMS.FMS.1110

CREATE_FILE.indication When generated

The CREATE_FILE.indication primitive shall be passed by the FS provider to the receiving User Entity in response to a CREATE_FILE.request with File Result Metadata indicating if the request was executed successfully or not, and the reason in case of failure.

Verification Method: T

8.3.3.2.4 Effect on Receipt

The response of the User Entity to a CREATE_FILE.indication is unspecified.

8.3.3.2.5 Additional constraints

SAVOIR.MMS.FMS.1120

CREATE_FILE.indication Not Enough Space

The CREATE_FILE.indication File Result Metadata shall report a failure if the requested Maximum File Size is greater than the current space available in the File Store.

Comment: The current space available in the File Store refers to all the BAU (and indirectly underlying SAU) that can be used by the File System to store new data (i.e. capacity of the File Store minus already reserved BAUs).

Verification Method: T

SAVOIR.MMS.FMS.1130

CREATE_FILE.indication Invalid Size

The CREATE_FILE.indication File Result Metadata shall report a failure if the requested Maximum File Size is negative or null.

Verification Method: T

SAVOIR.MMS.FMS.1140

CREATE_FILE.indication File already exists

The CREATE_FILE.indication File Result Metadata shall report a failure if the requested File Name already exists within the parent directory (i.e. last directory of the File Full Path).

Parent: SAVOIR-DSS-ORG-360

Verification Method: T

SAVOIR.MMS.FMS.1150

CREATE_FILE.indication File Full Path Invalid

The CREATE_FILE.indication File Result Metadata shall report a failure if the requested File Full Path is invalid.

Comment: Invalid means the file path or file name contain unauthorized characters or out of the allowed range integers, depending on the kind of identifier used.

Verification Method: T

SAVOIR.MMS.FMS.1160

CREATE_FILE.indication File Path does not exist

The CREATE_FILE.indication File Result Metadata shall report a failure if the requested File Path refers to an inexistent directory.

Comment: This is the case if at least one directory of the path does not exist.

Verification Method: T

8.3.3.3 OPEN_FILE.request

8.3.3.3.1 Function

SAVOIR.MMS.FMS.1170

OPEN_FILE.request Function

The OPEN_FILE.request primitive shall be passed to the FS provider to request the opening of an existing file, or the creation and then opening of a new file, for access in a File Store with the specified opening criteria.

Rationale: To open a File

Verification Method: T

SAVOIR.MMS.FMS.1180

OPEN_FILE.request Open and Lock

The FMS shall support, as an atomic action, the application of a file lock while opening the file whenever requested by the User Entity.

Rationale: The FMS enable the possibilities described in the opening criteria.

Verification Method: T

SAVOIR.MMS.FMS.1190

OPEN_FILE.request Full file action

The FMS shall support the action defined by the User Entity when the file remaining space is not sufficient for appending data segment at end of file.

Rationale: *To support autonomous storage and delegate file closing to the FMS.*

Comment: *This capability is available only when requesting file opening with "Append" Access Type. The behaviour also depends on the Packet Storage parameter.*
The file remaining space corresponds to the Maximum File Size minus the current file size (i.e. highest written position in the file).

Verification Method: T

8.3.3.3.2 Semantics

SAVOIR.MMS.FMS.1200

OPEN_FILE.request Semantics

The OPEN_FILE.request primitive shall use the following semantics:
OPEN_FILE.request (Transaction Identifier, File Full Path, File Opening Criteria)

Parent: SAVOIR-DSS-ORG-400

Verification Method: T

8.3.3.3.3 When generated

SAVOIR.MMS.FMS.1210

OPEN_FILE.request When generated

The OPEN_FILE.request primitive shall be passed to the FS provider to request the file to be opened or created and then opened, and optionally be locked.

Verification Method: T

8.3.3.3.4 Effect on Receipt

SAVOIR.MMS.FMS.1220

OPEN_FILE.request Effect on receipt: open

Receipt of the OPEN_FILE.request primitive shall cause the FS provider to open, or create and open the specified file while granting the user's entity with the access rights defined in the opening criteria.

Comment: It is not possible to define the maximum file size when it is created by the OPEN_FILE.request primitive.
The underlying File System limit or global configuration applies.

Verification Method: T

SAVOIR.MMS.FMS.1230

OPEN_FILE.request Effect on receipt: File Descriptor

The File Descriptor's current position within the file shall be initialized to the end of file when selecting "Append" Access type in the File Opening Criteria and to the start of the file otherwise.

Rationale: To identify the location where the first byte will be written in the File.

Verification Method: T

8.3.3.3.5 Additional constraints

SAVOIR.MMS.FMS.1240

OPEN_FILE.request Additional constraints

When a file is autonomously created by the FS provider in answer to the Full File Action parameter, the new created file shall:

- inherit the maximum file size of its ancestor (i.e. the currently opened file which misses space for requested data appending),
- be opened with exactly the same parameters as its ancestor (File Opening Criteria parameter),
- inherit all the file attributes of its ancestor,
- inherit any mapping criteria (cf. MAP_FILE.request) of its ancestor,
- continue the autonomous data storage without any data loss.

Comment: The naming of the new created file is implementation dependent. It may rely on an incremented prefix or suffix added to the initial file name to identify the series (and creation order) among other files.

Verification Method: T

The table below presents a synthesis of file opening capabilities and associated behavior:

Primitive	Access Type	Full File Action	Packet Storage	Behavior
OPEN	Read-only	N/A	N/A	Pointer positioned at beginning of file. Read and seek operations supported in this opened file.
OPEN	Read-write	N/A	N/A	Pointer positioned at beginning of file. Read, write and seek operations supported in this opened file. Read and write are performed at current pointer position and automatically move it by the effective number of read or written bytes. When the file remaining space is not sufficient to write data, data is partially written until reaching Maximum File Size and the remaining part is discarded. No more data can be written at the end (but overwrite anywhere remains possible). Read also remains possible.
OPEN	Append	Close File	No	Pointer positioned at end of file. Only write operation supported in this opened file. Write appends data to the file from the last written position (i.e. previous content is not lost) and current pointer is automatically moved by the effective number of written bytes. When the file remaining space is not sufficient to append data at end of file, data is partially written until reaching Maximum File Size, the remaining part is discarded and the file is finally closed (potential mapping is disabled).
OPEN	Append	Close File & Create Next File	No	Same as opening with access type = “Append”, no Packet Storage and FFA= “Close File” except that when the file remaining space is not sufficient to append data at end of file, data is partially written until reaching Maximum File Size, then a new file is created, the remaining data part written to this new file, and the current file is closed. Potential mapping is inherited by the new created file. No data loss during the transition between files. Data can be appended to the new created file.

Primitive	Access Type	Full File Action	Packet Storage	Behavior
OPEN	Append	Close File	Yes	Pointer positioned at end of file. Only write operation supported in this opened file. Write appends data to the file from the last written position (i.e. previous content is not lost) and current pointer is automatically moved by the effective number of written bytes. When the file remaining space is not sufficient to append data at end of file, data is entirely discarded and the file is closed (potential mapping is disabled).
OPEN	Append	Close File & Create Next File	Yes	Same as opening with access type = “Append”, Packet Storage and FFA= “Close File” except that when the file remaining space is not sufficient to append data at end of file, a new file is created, data is entirely written to this new file, and current file is closed.

Table 1: Behaviour depending on File Opening Criteria

8.3.3.4 OPEN_FILE.indication

8.3.3.4.1 Function

SAVOIR.MMS.FMS.1250

OPEN_FILE.indication Function

The OPEN_FILE.indication primitive shall be used to indicate the outcome of opening a file to the User Entity.

Verification Method: T

8.3.3.4.2 Semantics

SAVOIR.MMS.FMS.1260

OPEN_FILE.indication Semantics

The OPEN_FILE.indication primitive shall use the following semantics:
OPEN_FILE.indication(Transaction Identifier, File Descriptor, File Result Metadata)

Verification Method: T

8.3.3.4.3 When generated

SAVOIR.MMS.FMS.1270

OPEN_FILE.indication When generated

The OPEN_FILE.indication primitive shall be passed by the FMS provider to the receiving User Entity in response to an OPEN_FILE.request with File Result Metadata indicating if the request was executed successfully or if not

Verification Method: T

SAVOIR.MMS.FMS.1280

OPEN_FILE.indication When generated success

When OPEN_FILE.request is successful, the File Result Metadata shall provide the File Descriptor allocated to the opened file (to logically identify it at FMS level).

Verification Method: T

SAVOIR.MMS.FMS.1290

OPEN_FILE.indication When generated failure

When OPEN_FILE.request is unsuccessful, the File Result Metadata shall provide the reason of the failure.

Verification Method: T

8.3.3.4.4 Effect on Receipt

The response of the User Entity to an OPEN_FILE.indication is unspecified.

8.3.3.4.5 Additional constraints

SAVOIR.MMS.FMS.1300

OPEN_FILE.indication Access not granted

The OPEN_FILE.indication File Result Metadata shall report a failure if the requested access rights in file opening criteria cannot be granted to the User Entity.

Comment: For example, when requesting the application of a "Read_Only" lock together with a "Read-Write" Access Type in the File Opening Criteria. However requesting opening with application of a "Single_Writer" lock, and "Read-Only" Access Type is possible.

Verification Method: T

SAVOIR.MMS.FMS.1310

OPEN_FILE.indication File locked

The OPEN_FILE.indication File Result Metadata shall report a failure if the application of a lock was requested whereas a lock was already applied to the file.

Verification Method: T

The next table provides indicates when it is possible to Open a File.

Access Type (in File Opening Criteria)	Lock Type (in File Opening Criteria)	Lock already applied?			Opening possible?
		Yes/ No	Type	Lock owner	
Read-Only	*Exclusive_Read_Only*	No			Yes
Read-Only	Read_Only	No			Yes
Read-Only	Exclusive_Access	No			Yes
Read-Only	Single_Writer	No			Yes
Read-Only	None (no lock requested)	Yes	Exclusive_Read_Only	Requesting User Entity	Yes
Read-Only	None (no lock requested)	Yes	Exclusive_Read_Only	Other User Entity	No
Read-Only	None(no lock requested)	Yes	Read_Only	Any	Yes
Read-Only	None(no lock requested)	Yes	Exclusive_Access	Requesting User Entity	Yes
Read-Only	None(no lock requested)	Yes	Exclusive_Access	Other User Entity	No
Read-Only	None(no lock requested)	Yes	Single_Writer	Any	Yes
Read-Write or Append	Exclusive_Read_Only	No			No
Read-Write or Append	Read_Only	No			No

Access Type (in File Opening Criteria)	Lock Type (in File Opening Criteria)	Lock already applied?			Opening possible?
		Yes/ No	Type	Lock owner	
Read-Write or Append	Exclusive_Access	No			Yes
Read-Write or Append	Single_Writer	No			Yes
Read-Write or Append	None(no lock requested)	Yes	Exclusive_Read_ Only	Any	No
Read-Write or Append	None(no lock requested)	Yes	Read_Only	Any	No
Read-Write or Append	None(no lock requested)	Yes	Exclusive_Access	Requesting User Entity	Yes
Read-Write or Append	None(no lock requested)	Yes	Exclusive_Access	Other User Entity	No
Read-Write or Append	None(no lock requested)	Yes	Single_Writer	Requesting User Entity	Yes
Read-Write or Append	None(no lock requested)	Yes	Single_Writer	Other User Entity	No

Table 2: Conditions for granting files access

SAVOIR.MMS.FMS.1320

OPEN_FILE.indication File Full Path invalid

The OPEN_FILE.indication File Result Metadata shall report a failure if the requested File Full Path refers to an inexistent file and the "Create" parameter was not set.

Comment: When a failure is returned, the file is not opened for the requesting user's entity.

Verification Method: T

SAVOIR.MMS.FMS.1330

OPEN_FILE.indication Not Enough Space

The OPEN_FILE.indication File Result Metadata shall report a failure if file creation is requested ("Create" parameter is set while file to be opened does not exist) but the Maximum File Size applicable by default to the considered File Store is greater than the current space available in this File Store.

Comment: The File Store remaining space would not be sufficient to allow fulfilling the file up to the Maximum File Size. Note that the Maximum File Size can be independently configured per file with the CREATE_FILE.request primitive.

Verification Method: T

8.3.3.5 MAP_FILE.request

8.3.3.5.1 Function

SAVOIR.MMS.FMS.1340

MAP_FILE.request Function

The MAP_FILE.request primitive shall be passed to the FS provider to request autonomous data storage into an existing file, or a succession of files (if Continuous Storage is selected) with respect to the specified mapping criteria.

Verification Method: T

8.3.3.5.2 Semantics

SAVOIR.MMS.FMS.1350

MAP_FILE.request Semantics

The MAP_FILE.request primitive shall use the following semantics:
MAP_FILE.request(Transaction Identifier, File Full Path, Mapping Criteria)

Verification Method: T

8.3.3.5.3 When generated

SAVOIR.MMS.FMS.1360

MAP_FILE.request When generated

The MAP_FILE.request primitive shall be passed to the FS provider to request the file to be opened and data received on MM inputs that match the specified Mapping Criteria to be appended to the file.

Verification Method: T

8.3.3.5.4 Effect on Receipt

SAVOIR.MMS.FMS.1370

MAP_FILE.request Effect on Receipt: open and map

Receipt of the MAP_FILE.request primitive shall cause the FMS provider to open the specified file, associate it with the Mapping Criteria and begin autonomous storage with respect to these criteria.

Verification Method: T

SAVOIR.MMS.FMS.1380

MAP_FILE.request Effect on Receipt: autonomous storage

The autonomous data storage shall consist in appending to the file (once successfully opened) all the data received on MM input interfaces that match the associated Mapping Criteria.

Rationale: *Data written to the file have to comply with the File Opening Criteria provided when the file was opened by the service.*

Comment: *The Access Type parameter in the File Opening Criteria has to be set to "Append" when the file is being opened by this service.*

Comment: *Requesting file locking and setting the Packet Storage parameter is implementation dependent (and may be related to the type of mapping).*

Verification Method: T

8.3.3.5.5 Additional constraints

SAVOIR.MMS.FMS.1390

MAP_FILE.request Additional constraints: closing file

Closing the initial file opened by the FMS when requesting autonomous storage shall cause the Mapping Criteria to be de-associated from the file and no more data matching these Mapping Criteria to be appended to the file.

Comment: Refer to CLOSE_FILE.request primitive for details on file closing.

Verification Method: T

SAVOIR.MMS.FMS.1400

MAP_FILE.request Additional constraints: multiple Mapping Criteria

It shall be possible to associate multiple Mapping Criteria (Exclusive mapping excluded) to a single file, each mapping criteria being considered independently.

Comment: It is possible to call the MAP_FILE.request primitive as many times as mapping criteria have to be associated to a file for autonomous storage. Each "mapping" request returns a File Descriptor which can be further used to de-associate the specific Mapping Criteria(without affecting the other ones) by calling the CLOSE_FILE.request. As an example it would be possible to request autonomous storage of all packets received on MM inputs with a PUS service equal to 3 or 5 into the same file.

Verification Method: T

SAVOIR.MMS.FMS.1410

MAP_FILE.request Additional constraints: Continuous Storage

Setting the Continuous Storage parameter in the Mapping Criteria shall request the FMS provider to continue the autonomous storage into a new file without data loss when the currently opened one gets full.

Comment: The Full File Action parameter in the File Opening Criteria must be set to "Close File & Create Next File" when the file is being opened by this service and the Continuous Storage parameter is set.
Refer to OPEN_FILE.request primitive for characteristics of the new created file.

Parent: SAVOIR-DSS-ORG-550

Verification Method: T

SAVOIR.MMS.FMS.1420

MAP_FILE.request Additional constraints: Non Continuous Storage

Not setting the Continuous Storage parameter in the Mapping Criteria shall request the FMS provider to close the file when it gets full.

Rationale: *As the service autonomously opens the file to be mapped, it is consistent to close it without external involvement (e.g. command from OBC).*

Comment: *The Full File Action parameter in the File Opening Criteria must be set to "Close File" when the file is being opened by the service and the Continuous Storage parameter is not set.*

Parent: SAVOIR-DSS-ORG-420

Verification Method: T

SAVOIR.MMS.FMS.1430

MAP_FILE.request Additional constraints: Proxy Mapping

Setting the Proxy Mapping parameter in the Mapping Criteria when the File Full Path refers to a remote File Store shall request the FMS provider to delegate the mapping operation to the remote FMS (proxy).

Comment: *This is done by forwarding the MAP_FILE.request primitive via a Remote File Management Protocol.*
The Proxy Mapping parameter shall be ignored when the File Full Path refers to a local File Store (local from the "point of view" of the FMS receiving the primitive).

Verification Method: T

SAVOIR.MMS.FMS.1440

MAP_FILE.request Additional constraints: Delegation

All operations directly related to the mapping criteria shall be performed by the FMS provider on behalf of the user's entity at the origin of the MAP_FILE.request.

Comment: *Especially for Continuous Storage which involves file creation, or lock at opening.*

Verification Method: T

SAVOIR.MMS.FMS.1450

MAP_FILE.request Additional constraints: Data does not match

The FMS shall discard data received over any of its input interfaces if this data does not match any mapping criteria currently associated to a file.

Rationale: The data shall be stored within an identified file or ignored, e.g. to protect the system from 'babbling idiot'.

Verification Method: T

8.3.3.6 MAP_FILE.indication

8.3.3.6.1 Function

SAVOIR.MMS.FMS.1460

MAP_FILE.indication Function

The MAP_FILE.indication primitive shall be used to indicate to the User Entity the outcome of opening a File for autonomous storage.

Verification Method: T

8.3.3.6.2 Semantics

SAVOIR.MMS.FMS.1470

MAP_FILE.indication Semantics

The MAP_FILE.indication primitive shall use the following semantics:
MAP_FILE.indication(Transaction Identifier, File Descriptor, File Result Metadata)

Verification Method: T

8.3.3.6.3 When generated

SAVOIR.MMS.FMS.1480

MAP_FILE.indication When generated

The MAP_FILE.indication primitive shall be passed by the FS provider to the receiving User Entity in response to an MAP_FILE.request with File Result Metadata indicating if the request was executed successfully or not.

Verification Method: T

SAVOIR.MMS.FMS.1490

MAP_FILE.indication When generated success

When MAP_FILE.request is successful, the File Result Metadata shall provide the File Descriptor allocated to the opened file (to logically identify it at FS level).

Comment: The File Descriptor is returned (rather than a Mapping Identifier for example) to allow the User Entity to append its own data at any time (i.e. interleaving with autonomously stored data). This can be required to periodically insert synchronization markers for example.

Verification Method: T

SAVOIR.MMS.FMS.1500

MAP_FILE.indication When generated failure

When MAP_FILE.request is unsuccessful, the File Result Metadata shall provide the reason of the failure.

Verification Method: T

8.3.3.6.4 Effect on Receipt

The response of the User Entity to a MAP_FILE.indication is unspecified.

8.3.3.6.5 Additional constraints

SAVOIR.MMS.FMS.1510

MAP_FILE.indication Additional constraints: File Full Path invalid

The MAP_FILE.indication File Result Metadata shall report a failure if the requested File Full Path refers to an inexistent file.

Rationale: It is only possible to map data to an existing file.

Verification Method: T

SAVOIR.MMS.FMS.1520

MAP_FILE.indication Additional constraints: Exclusive mapping invalid

The MAP_FILE.indication File Result Metadata shall report a failure if the Exclusive mapping of an input interface was requested whereas this interface:

- is invalid (not referring to a known interface identifier),
- is not configured as RAW type,
- is already mapped to another file.

- *Verification Method: T*

SAVOIR.MMS.FMS.1530

MAP_FILE.indication Additional constraints: Interface invalid

The MAP_FILE.indication File Result Metadata shall report a failure if a Protocol Address mapping was requested whereas no input interface is configured with LINK_PROTOCOL type (and thus the condition could never be met).

Comment: Protocol Address mapping requires handling input data at the level of Data Link layer entity (SpaceWire packet, 1553 Frame, etc.). The FMS is not responsible for data reception through hardware interface (i.e. at low level). This is performed by a dedicated MM subsystem which provides reassembled data to the FMS based on the configured input interface type (interface between the subsystem and FMS is implementation dependent).

Verification Method: T

SAVOIR.MMS.FMS.1540

MAP_FILE.indication Additional constraints: No Packet Interface

The MAP_FILE.indication File Result Metadata shall report a failure if the requested mapping applies to a Packet field (i.e. Mapping Type is either *Packet APID*; *Packet PUS Service*; *Packet PUS Service &Subservice*; *Packet APID, PUS Service & Subservice* or *Custom Packet Field*) whereas no input interface is configured with PACKET type.

Comment: Mapping on Packet field requires handling input data at the level of CCSDS Packets (which are provided by a dedicated MM subsystem, responsible for data reception through hardware interfaces, and reassembly).

Verification Method: T

8.3.3.7 CLOSE_FILE.request

8.3.3.7.1 Function

SAVOIR.MMS.FMS.1550

CLOSE_FILE.request Function

The CLOSE_FILE.request primitive shall be passed to the FS provider to request the closing of the specified file in the specified File Store and stop any autonomous storage associated to the File Descriptor.

Rationale: To prevent access (read, write) to the file content when it is no longer needed and stop storage when a mapping was established.

Verification Method: T

8.3.3.7.2 Semantics

SAVOIR.MMS.FMS.1560

CLOSE_FILE.request Semantics

The CLOSE_FILE.request primitive shall use the following semantics:
CLOSE_FILE.request(Transaction Identifier, File Descriptor)

Verification Method: T

8.3.3.7.3 When generated

SAVOIR.MMS.FMS.1570

CLOSE_FILE.request When generated

The CLOSE_FILE.request primitive shall be passed to the FS provider to request the file to be closed.

Verification Method: T

8.3.3.7.4 Effect on Receipt

SAVOIR.MMS.FMS.1580

CLOSE_FILE.request Effect on Receipt

Receipt of the CLOSE_FILE.request primitive shall cause the FS provider to close the specified file if opened.

Comment: FMS and FS have to ensure that all the data have been transferred to the File before closing it.
FMS and FS have to release all the resources that were used to manage the File.

Verification Method: T

8.3.3.7.5 Additional constraints

SAVOIR.MMS.FMS.1590

CLOSE_FILE.request Additional constraints for Full File Action

Closing a file that was opened with the Full File Action parameter set to "Close File & Create Next File" shall result in closing the last created file of the series.

Rationale: *When requesting autonomous data storage, the OBC gets the File Descriptor of the initially opened file and then lets the MM manage the storage. It can be assumed that the OBC has not tracked any potential notification regarding file closing and creation, and thus has no knowledge of the currently used file for storage. It should however support easily stopping the storage process by requesting to close the initial file and letting the FMS close the proper file.*

Comment: *The FMS has to keep an association between the initial File Descriptor and the one of last automatically opened file.*

Verification Method: T

SAVOIR.MMS.FMS.1600

CLOSE_FILE.request Additional constraints for File opened multiple times

Closing a file (through the File Descriptor obtained at opening) shall only affect the User Entity at the origin of the request.

Comment: *This means that the User Entity having closed the file will no longer be able to read from or write data into it, but the file remains open for all other User Entities through their respective File Descriptors (which are not impacted). There is not a single open/close status per file.*

Verification Method: T

8.3.3.8 CLOSE_FILE.indication

8.3.3.8.1 Function

SAVOIR.MMS.FMS.1610

CLOSE_FILE.indication Function

The CLOSE_FILE.indication primitive shall be used to indicate the outcome of closing a file to the User Entity.

Verification Method: T

8.3.3.8.2 Semantics

SAVOIR.MMS.FMS.1620

CLOSE_FILE.indication Semantics

The CLOSE_FILE.indication primitive shall use the following semantics:
CLOSE_FILE.indication(Transaction Identifier, File Result Metadata)

Verification Method: T

8.3.3.8.3 When generated

SAVOIR.MMS.FMS.1630

CLOSE_FILE.indication When generated

The CLOSE_FILE.indication primitive shall be passed by the FS provider to the receiving User Entity in response to a CLOSE_FILE.request with File Result Metadata indicating if the request was executed successfully or not, and the reason in case of failure.

Verification Method: T

8.3.3.8.4 Effect on Receipt

The response of the User Entity to a CLOSE_FILE.indication is unspecified.

8.3.3.8.5 Additional constraints

SAVOIR.MMS.FMS.1640

CLOSE_FILE.indication Additional constraints on File Descriptor

The File Descriptor of the closed file shall become invalid as soon as the CLOSE_FILE.indication primitive is passed by the FS provider to the receiving User Entity if no error has been reported in File Result Metadata.

Comment: An invalid File Descriptor corresponds to a descriptor which is not attached to any opened file, and thus can no longer be used when calling FMS primitives.

Verification Method: T

SAVOIR.MMS.FMS.1650

CLOSE_FILE.indication Additional constraints File not opened

The CLOSE_FILE.indication File Result Metadata shall report a failure if the requested File Descriptor does not correspond to a currently opened file.

Verification Method: T

8.3.3.9 WRITE_TO_FILE.request

8.3.3.9.1 Function

SAVOIR.MMS.FMS.1660

WRITE_TO_FILE.request Function

The WRITE_TO_FILE.request primitive shall be passed to the FS provider to request in-order writing of a data segment of specified length in the specified file, starting at the File Descriptor's current position after its potential repositioning.

Comment: The File Descriptor's current position can be initialized differently, either by being placed at the specifically requested location or kept at current position. This is determined by the use of the appropriate parameter and depends on the File Opening Criteria (which allows or prohibits the User Entity to force a repositioning).

Comment: The current position is relative to the file content and not the BAUs that are used to store the data (the FS is in charge of the mapping).

Parent: SAVOIR-DSS-ORG-320

Verification Method: T

8.3.3.9.2 Semantics

SAVOIR.MMS.FMS.1670

WRITE_TO_FILE.request Semantics

The WRITE_TO_FILE.request primitive shall use the following semantics: WRITE_TO_FILE.request(Transaction Identifier, File Descriptor, File Segment, File Segment Length, Absolute File Offset (optional))

Verification Method: T

8.3.3.9.3 When generated

SAVOIR.MMS.FMS.1680

WRITE_TO_FILE.request When generated

The WRITE_TO_FILE.request primitive shall be passed to the FMS provider to request data segment to be written in-order to an opened file at the requested File Descriptor's position.

Verification Method: T

8.3.3.9.4 Effect on Receipt

SAVOIR.MMS.FMS.1690

WRITE_TO_FILE.request Effect on receipt: Write location

Receipt of the WRITE_TO_FILE.request primitive shall cause the FMS provider to evaluate the File's descriptor position according to the parameters and write the data segment to the specified opened file.

Verification Method: T

SAVOIR.MMS.FMS.1700

WRITE_TO_FILE.request Effect on receipt: Overwrite

If the evaluated File Descriptor's position is not at the end of file (highest written position) the file's original data shall be overwritten with the new data

Verification Method: T

SAVOIR.MMS.FMS.1710

WRITE_TO_FILE.request Effect on receipt: Append

If the evaluated File Descriptor's position is at the end of file (highest written position), the data shall be appended to the end of file.

Verification Method: T

SAVOIR.MMS.FMS.1720

WRITE_TO_FILE.request Effect on receipt: Current position update

The File Descriptor's current position shall be subsequently advanced through the file by the length of data effectively written.

Verification Method: T

8.3.3.9.5 Additional constraints

SAVOIR.MMS.FMS.1730

WRITE_TO_FILE.request Additional constraints: Data truncation

When the length of the data segment to be written (File Segment Length parameter) is bigger than the Maximum File Size minus File Descriptor's current position, data shall be written at most up to the Maximum File Size.

Rationale: *File content cannot exceed the configured maximum file size (used-defined or global), so only a partial writing may be performed and the effective amount of data written shall be returned to the User Entity (cf. indication primitive).*

Comment: *Depending on file opening criteria, the data may not be written/appended at all to this opened file.*

Verification Method: T

SAVOIR.MMS.FMS.1740

WRITE_TO_FILE.request Additional constraints: Can only append

When an absolute file offset is given as parameter of the request, if the file was opened in Append Mode, the WRITE_TO_FILE.request must be rejected.

Rationale *In Append mode the User's entity is not allowed to change the File's Descriptor current position, as it is handled automatically.*

Verification Method: T

SAVOIR.MMS.FMS.1750

WRITE_TO_FILE.request Additional constraints: Invalid location

When an absolute file offset is given as parameter of the request, if the position targets a memory location not associated to the file specified in this same request, the WRITE_TO_FILE.request must be rejected.

Rationale: *WRITE_TO_FILE.request does not allow starting a write action outside the targeted file boundaries.*

Comment: *In some FS, they are mechanisms to change the size of a file (up to the maximum file size) by changing the end of file position. At a macroscopic level, it can appear as writing data in locations that were within the original boundaries, but this is handled automatically by the FS and the memory accessed this way is known as available. When using the absolute position, this forces the repositioning of the File's Descriptor current position, thus it bypasses some controls and could lead to the point to other files and therefore corrupting them.*

Verification Method: T

SAVOIR.MMS.FMS.1760

WRITE_TO_FILE.request Additional constraints: Insufficient space

If the file was opened for appending with the Packet Storage parameter selected, and the remaining space is not sufficient to store the integral data segment, no data shall be appended to the file.

Rationale: *To ensure that the data segment (packet) is not truncated.*

Comment: *Opened for appending indicates that the "Append" Access Type was selected.*

Verification Method: T

SAVOIR.MMS.FMS.1770

WRITE_TO_FILE.request Additional constraints: Maximum Size reached

If the file was opened for appending and the Full File Action parameter was set to "Close File", data not appended to the file shall be discarded, and the file closed.

Comment: *If Packet Storage was selected, the integral data segment is discarded, otherwise only last part of data remaining when the Maximum File Size was reached (i.e. data part not written to the file).*

Verification Method: T

SAVOIR.MMS.FMS.1780

WRITE_TO_FILE.request Additional constraints: Continue writing

If the file was opened for appending and the Full File Action parameter was set to "Close File& Create Next File", data not appended to the file shall be written to the next file (without any data loss).

Comment: *If Packet Storage was selected, the integral data segment is written to the next file, otherwise only last part of data remaining when the Maximum File Size was reached (i.e. data part not written to the current file) is written to this next file.*

Verification Method: T

SAVOIR.MMS.FMS.1790

WRITE_TO_FILE.request Additional constraints: Next file creation

When data is to be written to the next file and this one already exists, it shall be opened, otherwise it shall be created prior opening.

Comment: *The next file is opened to try writing the remaining part of data segment. This mechanism shall recursively open the (next) files until being able to append the remaining data (i.e. up to the last file of the series).*

Verification Method: T

SAVOIR.MMS.FMS.1800

WRITE_TO_FILE.request Additional constraints: Too many data

When Packet Storage parameter is selected at opening, a write operation shall be definitely rejected (no data written at all) if the File Segment Length is bigger than the Maximum File Size.

Rationale: *To prevent infinite file creation in case the Full File Action is "Close File & Create Next File".*

Verification Method: T

SAVOIR.MMS.FMS.1810

WRITE_TO_FILE.request Additional constraints: Concurrent append

When a file is opened for appending by more than one User Entity at a time (through respective File Descriptors), respective File Descriptor's current positions shall be synchronized between each other's (i.e. appending a data segment to the file automatically moves the current position of other user entities).

Rationale: *To provide the capability to append data to a single file from multiple user entities, in parallel, and without ever overwriting previously written content.*

Comment: *Be warned that synchronization of current positions only applies between user entities that have opened the file with "Append" access type. If another User Entity opens the file in parallel with e.g. "Read" access type, its position within the file will be completely independent and not automatically updated on writing by other user entities.*

Verification Method: T

SAVOIR.MMS.FMS.1820

WRITE_TO_FILE.request Additional constraints: End of File

When Packet Storage parameter is selected at opening, the Maximum File Size shall be set to the current end of file (i.e. highest written position) before creating the next file.

Rationale: To prevent more data to be appended to the file (whatever the User Entity), and guarantee time-ordering of stored data.

Verification Method: T

8.3.3.10 WRITE_TO_FILE.indication

8.3.3.10.1 Function

SAVOIR.MMS.FMS.1830

WRITE_TO_FILE.indication Function

The WRITE_TO_FILE.indication primitive shall be used to indicate the outcome of writing a data segment to a file to the User Entity (and information about potential automatically performed FMS operations).

Verification Method: T

8.3.3.10.2 Semantics

SAVOIR.MMS.FMS.1840

WRITE_TO_FILE.indication Semantics

The WRITE_TO_FILE.indication primitive shall use the following semantics: WRITE_TO_FILE.indication(Transaction Identifier, File Segment Length, File Descriptor, File Closed, File Result Metadata)

Verification Method: T

8.3.3.10.3 When generated

SAVOIR.MMS.FMS.1850

WRITE_TO_FILE.indication When generated

The WRITE_TO_FILE.indication primitive shall be passed by the FS provider to the receiving User Entity in response to a WRITE_TO_FILE.request with File Result Metadata indicating if the request was executed successfully or not.

Verification Method: T

SAVOIR.MMS.FMS.1860

WRITE_TO_FILE.indication When generated success

When WRITE_TO_FILE.request is successful, the File Result Metadata shall provide the length of data actually written into the file, an indication that the file was closed or not by the FS and the File Descriptor to use for next writing (relevant if creation).

Verification Method: T

SAVOIR.MMS.FMS.1870

WRITE_TO_FILE.indication When generated failure

When WRITE_TO_FILE.request is unsuccessful, the File Result Metadata shall provide the reason of the failure.

Verification Method: T

8.3.3.10.4 Effect on Receipt

The response of the User Entity to a WRITE_TO_FILE.indication is unspecified.

8.3.3.10.5 Additional constraints

SAVOIR.MMS.FMS.1880

WRITE_TO_FILE.indication Additional constraints: Returned length

When the next file is automatically created/opened by the FS in the frame of a write operation, the returned File Segment Length shall correspond to the sum of data lengths written to both current and next file.

Comment: The data length (File Segment Length parameter) written to the current file may be null when Packet Storage parameter was selected in the File Opening Criteria.

Verification Method: T

SAVOIR.MMS.FMS.1890

WRITE_TO_FILE.indication Additional constraints: File not opened

The WRITE_TO_FILE.indication File Result Metadata shall report a failure if the requested File Descriptor does not correspond to a currently opened file.

Verification Method: T

SAVOIR.MMS.FMS.1900

WRITE_TO_FILE.indication Additional constraints: File not writable

The WRITE_TO_FILE.indication File Result Metadata shall report a failure if the requested File Descriptor corresponds to a file that was not opened for writing.

Comment: Not opened for writing means that the file was opened with "Read-only" Access Type in the File Opening Criteria.

Verification Method: T

SAVOIR.MMS.FMS.1910

WRITE_TO_FILE.indication Additional constraints: Invalid Length

The WRITE_TO_FILE.indication File Result Metadata shall report a failure if the requested File Segment Length is negative.

Comment: Requesting data writing with a null File Segment Length is not reported as an error.

Verification Method: T

SAVOIR.MMS.FMS.1920

WRITE_TO_FILE.indication Additional constraints: Returned File Descriptor

The returned File Descriptor parameter shall correspond to the file descriptor to be used for next write operation. It is either the descriptor provided in the request or the descriptor of the next file (with recursive consideration) if it was automatically created/opened by the FS in the frame of the write operation.

Verification Method: T

SAVOIR.MMS.FMS.1930

WRITE_TO_FILE.indication Additional constraints: File closed

The returned File Closed parameter shall indicate if the file (identified by the File Descriptor provided in the request) was closed or not in the frame of the write operation.

Comment: The User Entity's application has to monitor the returned "File Closed" and "File Descriptor" parameters to determine if another writing operation is possible and with which file descriptor.
For a file opened with the Full File Action set to "Close File & Create Next File", the creation/opening of the next file shall be indicated by: "File Closed" parameter set to true, and "File Descriptor" different from the one provided in the request.

Verification Method: T

8.3.3.11 READ_FROM_FILE.request

8.3.3.11.1 Function

SAVOIR.MMS.FMS.1940

READ_FROM_FILE.request Function

The READ_FROM_FILE.request primitive shall be passed to the FS provider to request in-order reading of a data segment of specified length in the specified file, starting at the File Descriptor's current position.

Comment: The File Descriptor's current position is initialized differently and is allowed or not to be moved by the User Entity, depending on file opening criteria.

Comment: The current position is relative to the file content and not the BAUs that are used to store the data (the FS is in charge of the mapping).

Parent: SAVOIR-DSS-ORG-320

Verification Method: T

8.3.3.11.2 Semantics

SAVOIR.MMS.FMS.1950

READ_FROM_FILE.request Semantics

The READ_FROM_FILE.request primitive shall use the following semantics:
READ_FROM_FILE.request(Transaction Identifier, File Descriptor, File Segment Length, Absolute File Offset (optional))

Verification Method: T

8.3.3.11.3 When generated

SAVOIR.MMS.FMS.1960

READ_FROM_FILE.request When generated

The READ_FROM_FILE.request primitive shall be passed to the FS provider to request data segment to be read in-order from an opened file at the File Descriptor's current position.

Verification Method: T

8.3.3.11.4 Effect on Receipt

SAVOIR.MMS.FMS.1970

READ_FROM_FILE.request Effect on receipt: Read data

Receipt of the READ_FROM_FILE.request primitive shall cause the FS provider to evaluate the File's descriptor position according to the parameters and to read a data segment from the specified opened file at the evaluated File Descriptor's position.

Verification Method: T

Verification Method: T

SAVOIR.MMS.FMS.1980

READ_FROM_FILE.request Effect on receipt: Read at End of File

If the File Descriptor's position is at the end of file, no data is read.

Verification Method: T

SAVOIR.MMS.FMS.1990

READ_FROM_FILE.request Effect on receipt: Read available data

If there is less data than requested from the File Descriptor's current position to the end of file, only the available data is read.

Verification Method: T

SAVOIR.MMS.FMS.2000

READ_FROM_FILE.request Effect on receipt: File Descriptor update

The File Descriptor's current position shall be subsequently advanced through the file by the length of data effectively read.

Comment: Successive calls to READ_FROM_FILE.request primitive allows sequential reading of file content.

Verification Method: T

8.3.3.11.5 Additional constraints

SAVOIR.MMS.FMS.2010

READ_FROM_FILE.request Additional constraints: Invalid location

When an absolute file offset is given as parameter of the request, if the position targets a memory location not associated to the file specified in this same request, the READ_FROM_FILE.request must be rejected.

Rationale: *READ_FROM_FILE.request doesn't not allow reading from outside the targeted file boundaries.*

Comment: *If the absolute position does not belong to the file, so is the read data. Therefore the requesting entity shall not receive such data otherwise this could create discrepancies.*

Verification Method: T

8.3.3.12 READ_FROM_FILE.indication

8.3.3.12.1 Function

SAVOIR.MMS.FMS.2020

READ_FROM_FILE.indication Function

The READ_FROM_FILE.indication primitive shall be used to pass the data segment read from a file to the User Entity.

Verification Method: T

8.3.3.12.2 Semantics

SAVOIR.MMS.FMS.2030

READ_FROM_FILE.indication Semantics

The READ_FROM_FILE.indication primitive shall use the following semantics: READ_FROM_FILE.indication(Transaction Identifier, File Segment, File Segment Length, File Result Metadata)

Comment: *The File Segment parameter contains the data read from the file. The "File Segment Length" parameter may be lower than the value requested in READ_FROM_FILE.request primitive.*

Verification Method: T

8.3.3.12.3 When generated

SAVOIR.MMS.FMS.2040

READ_FROM_FILE.indication When generated

The READ_FROM_FILE.indication primitive shall be passed by the FS provider to the receiving User Entity in response to a READ_FROM_FILE.request with File Result Metadata indicating if the operation was successful or not.

Verification Method: T

SAVOIR.MMS.FMS.2050

READ_FROM_FILE.indication When generated success

When READ_FROM_FILE.request is successful, the service shall return the information related to the read data:

- File Segment parameter contains the data segment actually read from the file
- File Segment Length parameter contains the length of the data segment read.

Comment: The File Segment Length may be different from the value provided in the READ_FROM_FILE.request, e.g. when the file does not contain enough data (e.g. end of file is reached).

Verification Method: T

SAVOIR.MMS.FMS.2060

READ_FROM_FILE.indication When generated failure

When READ_FROM_FILE.request is unsuccessful, the File Result Metadata shall provide the reason of the failure.

Verification Method: T

8.3.3.12.4 Effect on Receipt

The response of the User Entity to a READ_FROM_FILE.indication is unspecified.

8.3.3.12.5 Additional constraints

SAVOIR.MMS.FMS.2070

READ_FROM_FILE.indication Additional constraints: File not opened

The READ_FROM_FILE.indication Result Metadata shall report a failure if the requested File Descriptor does not correspond to a currently opened file for reading.

Verification Method: T

SAVOIR.MMS.FMS.2080

READ_FROM_FILE.indication Additional constraints: Invalid length

The READ_FROM_FILE.indication Result Metadata shall report a failure if the requested File Segment Length is negative.

Comment: Requesting data reading with a null File Segment Length shall not be reported as an error.

Verification Method: T

8.3.3.13 SEEK_FILE.request

8.3.3.13.1 Function

SAVOIR.MMS.FMS.2090

SEEK_FILE.request Function

The SEEK_FILE.request primitive shall be passed to the FS provider to request the repositioning of the File Descriptor's current position within a specified open file to a new position computed from a reference position and offset.

Comment: The File Descriptor's current position defines the start position for next read or write operation.

Comment: The current position is relative to the file content and not the BAUs that are used to store the data (the FS is in charge of the mapping).

Parent: SAVOIR-DSS-ORG-320

Verification Method: T

8.3.3.13.2 Semantics

SAVOIR.MMS.FMS.2100

SEEK_FILE.request Semantics

The SEEK_FILE.request primitive shall use the following semantics:
SEEK_FILE.request(File Transaction Identifier, File Descriptor, Relative File Offset, Whence)

Verification Method: T

8.3.3.13.3 When generated

SAVOIR.MMS.FMS.2110

SEEK_FILE.request When generated

The SEEK_FILE.request primitive shall be passed to the FS provider to request repositioning the File Descriptor's current position within the specified open file for the next read or write operation.

Verification Method: T

8.3.3.13.4 Effect on Receipt

SAVOIR.MMS.FMS.2120

SEEK_FILE.request Effect on receipt: Current position updated

Receipt of the SEEK_FILE.request primitive shall cause the FMS provider to reposition the File Descriptor's current position within the specified open file to the requested offset.

Verification Method: T

SAVOIR.MMS.FMS.2130

SEEK_FILE.request Effect on receipt: Relative offset

The Relative File Offset shall be interpreted depending on the reference position ("Whence" parameter):
- *SEEK_SET* means the offset is relative to the beginning of the file,
- *SEEK_CUR* to the File Descriptor's current position,
- *SEEK_END* to the end of file.

Verification Method: T

SAVOIR.MMS.FMS.2140

SEEK_FILE.request Effect on receipt: Offset value

The offset shall be positive to move forward the specified reference position, negative to move backward and null to refer to the reference position.

Comment: To move backward from the end of file, the reference position shall be set to SEEK_END, and the offset be negative. To append, the reference position shall be set to SEEK_END, and the offset be zero.

Verification Method: T

8.3.3.13.5 Additional constraints

SAVOIR.MMS.FMS.2150

SEEK_FILE.request Additional constraints: Outside range

The operation shall be rejected if the requested position, resulting from reference position and relative offset, is not within the range of file content (i.e. between beginning and end of file, both included).

Comment: Position 0 refers to the beginning of file (i.e. first byte when file size is not null). End of file refers to the position immediately following the highest written position (i.e. equal to current file size).

Verification Method: T

8.3.3.14 SEEK_FILE.indication

8.3.3.14.1 Function

SAVOIR.MMS.FMS.2160

SEEK_FILE.indication Function

The SEEK_FILE.indication primitive shall be used to pass the outcome of repositioning the File Descriptor's current position within the specified file to the User Entity.

Verification Method: T

8.3.3.14.2 Semantics

SAVOIR.MMS.FMS.2170

SEEK_FILE.indication Semantics

The SEEK_FILE.indication primitive shall use the following semantics:
SEEK_FILE.indication(File Transaction Identifier, Resulting File Offset, File Result Metadata)

Verification Method: T

8.3.3.14.3 When generated

SAVOIR.MMS.FMS.2180

SEEK_FILE.indication When generated

The SEEK_FILE.indication primitive shall be passed by the FS provider to the receiving User Entity in response to a SEEK_FILE.request and File Result Metadata indicating if the request was executed successfully or not.

Verification Method: T

SAVOIR.MMS.FMS.2190

SEEK_FILE.indication When generated success

When SEEK_FILE.request is successful, the File Result Metadata shall provide the resulting file offset –corresponding to the File Descriptor's current position – within the file.

Verification Method: T

SAVOIR.MMS.FMS.2200

SEEK_FILE.indication When generated failure

When SEEK_FILE.request is unsuccessful, the File Result Metadata shall provide the reason of the failure.

Verification Method: T

8.3.3.14.4 Effect on Receipt

The response of the User Entity to a SEEK_FILE.indication is unspecified.

8.3.3.14.5 Additional constraints

SAVOIR.MMS.FMS.2210

SEEK_FILE.indication Additional constraints: File not opened

The SEEK_FILE.indication File Result Metadata shall report a failure if the requested File Descriptor does not correspond to a currently opened file.

Verification Method: T

SAVOIR.MMS.FMS.2220

SEEK_FILE.indication Additional constraints: Invalid reference

The SEEK_FILE.indication File Result Metadata shall report a failure if the requested reference position ("Whence" parameter) is invalid.

Comment: Invalid means different from SEEK_SET, SEEK_CUR or SEEK_END.

Verification Method: T

SAVOIR.MMS.FMS.2230

SEEK_FILE.indication Additional constraints: File only for appending

The SEEK_FILE.indication File Result Metadata shall report a failure if the requested File Descriptor corresponds to a file that was opened for appending data only.

Comment: Refers to a file that was opened with "Append" Access Type in the File Opening Criteria.

Verification Method: T

SAVOIR.MMS.FMS.2240

SEEK_FILE.indication Additional constraints: File invalid position

The SEEK_FILE.indication File Result Metadata shall report a failure if the requested position is out of file content range.

Comment: Seek is only possible between beginning and end-of-file, both included.

Verification Method: T

8.3.3.15 SHRINK_FILE.request

8.3.3.15.1 Function

SAVOIR.MMS.FMS.2250

SHRINK_FILE.request Function

The SHRINK_FILE.request primitive shall be passed to the FS provider to request truncating the content of an opened file to the requested file size.

Comment: Truncating means discarding data bytes exceeding the requested file size from file content, so that they can no longer be read.

Verification Method: T

8.3.3.15.2 Semantics

SAVOIR.MMS.FMS.2260

SHRINK_FILE.request Semantics

The SHRINK_FILE.request primitive shall use the following semantics:
SHRINK_FILE.request(File Transaction Identifier, File Descriptor, File Size)

Verification Method: T

8.3.3.15.3 When generated

SAVOIR.MMS.FMS.2270

SHRINK_FILE.request When generated

The SHRINK_FILE.request primitive shall be passed to the FS provider to request data after the requested size to be discarded from file content and the file size reduced accordingly.

Verification Method: T

8.3.3.15.4 Effect on Receipt

SAVOIR.MMS.FMS.2280

SHRINK_FILE.request Effect on Receipt

Receipt of the SHRINK_FILE.request primitive shall cause the FS provider to redefine the end of file position to the requested size for an opened file.

Comment: Modification of the current position is recorded in the File Descriptor.

Verification Method: T

8.3.3.15.5 Additional constraints

SAVOIR.MMS.FMS.2290

SHRINK_FILE.request Additional constraints: Access

Truncation shall be supported only for files opened for reading and writing.

Rationale: The operation is a modification of the File and requires access in Writing.

Comment: Opened for reading and writing indicates that the "Read-Write" Access Type was selected in File Opening Criteria.

Verification Method: T

SAVOIR.MMS.FMS.2300

SHRINK_FILE.request Additional constraints: File in use

The operation shall be rejected if the current position of at least one User Entity having opened this file (i.e. its File Descriptor's current position) is beyond the new requested end of file, i.e. within the part to be truncated.

Rationale: It is not possible to truncate a file to a given position while a User Entity is currently operating beyond this position.

Verification Method: T

8.3.3.16 SHRINK_FILE.indication

8.3.3.16.1 Function

SAVOIR.MMS.FMS.2310

SHRINK_FILE.indication Function

The SHRINK_FILE.indication primitive shall be used to pass the outcome of truncating a file to the User Entity.

Verification Method: T

8.3.3.16.2 Semantics

SAVOIR.MMS.FMS.2320

SHRINK_FILE.indication Semantics

The SHRINK_FILE.indication primitive shall use the following semantics:
SHRINK_FILE.indication(File Transaction Identifier, File Result Metadata)

Verification Method: T

8.3.3.16.3 When generated

SAVOIR.MMS.FMS.2330

SHRINK_FILE.indication When generated

The SHRINK_FILE.indication primitive shall be passed by the FS provider to the receiving User Entity in response to a SHRINK_FILE.request with Result Metadata indicating if the request was executed successfully or not.

Verification Method: T

8.3.3.16.4 Effect on Receipt

The response of the User Entity to a SHRINK_FILE.indication is unspecified.

8.3.3.16.5 Additional constraints

SAVOIR.MMS.FMS.2340

SHRINK_FILE.indication Additional constraints: File not opened

The SHRINK_FILE.indication File Result Metadata shall report a failure if the requested File Descriptor does not correspond to a currently open file.

Verification Method: T

SAVOIR.MMS.FMS.2350

SHRINK_FILE.indication Additional constraints: Invalid Size

The SHRINK_FILE.indication File Result Metadata shall report a failure if the requested File Size is negative or bigger than the current file size.

Verification Method: T

SAVOIR.MMS.FMS.2360

SHRINK_FILE.indication Additional constraints: Invalid File Access Type

The SHRINK_FILE.indication File Result Metadata shall report a failure if the requested File Descriptor corresponds to a file that was opened for reading-only or appending.

Verification Method: T

SAVOIR.MMS.FMS.2370

Deleted

8.3.3.17 GET_FILE_STATUS.request

8.3.3.17.1 Function

SAVOIR.MMS.FMS.2380

GET_FILE_STATUS.request Function

The GET_FILE_STATUS.request primitive shall be passed to the FS provider to request the File Status of a specified File.

Verification Method: T

8.3.3.17.2 Semantics

SAVOIR.MMS.FMS.2390

GET_FILE_STATUS.request Semantics

The GET_FILE_STATUS.request primitive shall use the following semantics:
GET_FILE_STATUS.request(File Transaction Identifier, File Full Path)

Verification Method: T

8.3.3.17.3 When generated

SAVOIR.MMS.FMS.2400

GET_FILE_STATUS.request When generated

The GET_FILE_STATUS.request primitive shall be passed to the FS provider to request the File Status of a specific file.

Verification Method: T

8.3.3.17.4 Effect on Receipt

SAVOIR.MMS.FMS.2410

GET_FILE_STATUS.request Effect on Receipt

Receipt of the GET_FILE_STATUS.request primitive shall cause the FS provider to get information about the file corresponding to the provided File Full Path.

Comment: The status information are used (and managed) internally by the FS to provide its services. For this reason, users cannot directly interact with them, and this information is rather the result of operations performed when calling FS primitives (ex: file size, lock).

Verification Method: T

8.3.3.17.5 Additional constraints

None.

8.3.3.18 GET_FILE_STATUS.indication

8.3.3.18.1 Function

SAVOIR.MMS.FMS.2420

GET_FILE_STATUS.indication Function

The GET_FILE_STATUS.indication primitive shall be used to pass the File Status of the File identified File Full Path to the User Entity.

Verification Method: T

8.3.3.18.2 Semantics

SAVOIR.MMS.FMS.2430

GET_FILE_STATUS.indication Semantics

The GET_FILE_STATUS.indication primitive shall use the following semantics:
GET_FILE_STATUS.indication(File Transaction Identifier, File Status, File Result Metadata)

Verification Method: T

8.3.3.18.3 When generated

SAVOIR.MMS.FMS.2440

GET_FILE_STATUS.indication When generated

The GET_FILE_STATUS.indication primitive shall be passed by the FS provider to the receiving User Entity in response to a GET_FILE_STATUS.request with File Result Metadata indicating if the operation was successful or not.

Verification Method: T

SAVOIR.MMS.FMS.2450

GET_FILE_STATUS.indication When generated success

When GET_FILE_STATUS.request is successful, the File Status parameter shall contain all the information related to the File.

Comment: The operation can be done on file currently opened and used by other User Entity, i.e. some information in the File Status may change. To avoid this, the File has to be locked before requesting its Status.

Verification Method: T

SAVOIR.MMS.FMS.2460

GET_FILE_STATUS.indication When generated failure

When GET_FILE_STATUS.request is unsuccessful, the File Result Metadata shall provide the reason of the failure.

Verification Method: T

8.3.3.18.4 Effect on Receipt

The response of the User Entity to a GET_FILE_STATUS.indication is unspecified.

8.3.3.18.5 Additional constraints

SAVOIR.MMS.FMS.2470

GET_FILE_STATUS.indication Additional constraints: File does not exist

The GET_FILE_STATUS.indication File Result Metadata shall report a failure if the requested File Full Path refers to an inexistent file.

Verification Method: T

8.3.3.19 SET_FILE_ATTRIBUTE.request

8.3.3.19.1 Function

SAVOIR.MMS.FMS.2480

SET_FILE_ATTRIBUTE.request Function

The SET_FILE_ATTRIBUTE.request primitive shall be passed to the FS provider to request associating an Attribute to a specified file.

Comment: An attribute is an information complementary to the file content.

Verification Method: T

8.3.3.19.2 Semantics

SAVOIR.MMS.FMS.2490

SET_FILE_ATTRIBUTE.request Semantics

The SET_FILE_ATTRIBUTE.request primitive shall use the following semantics:
SET_FILE_ATTRIBUTE.request(File Transaction Identifier, File Full Path, Attribute)

Verification Method: T

8.3.3.19.3 When generated

SAVOIR.MMS.FMS.2500

SET_FILE_ATTRIBUTE.request When generated

The SET_FILE_ATTRIBUTE.request primitive shall be passed to the FS provider to request an attribute to be associated (created or updated) to the specified file.

Verification Method: T

8.3.3.19.4 Effect on Receipt

SAVOIR.MMS.FMS.2510

SET_FILE_ATTRIBUTE.request Effect on Receipt: Attribute verification

Receipt of the SET_FILE_ATTRIBUTE.request primitive shall cause the FMS provider to first check if the attribute is already associated to the specified file.

Verification Method: T

SAVOIR.MMS.FMS.2520

SET_FILE_ATTRIBUTE.request Effect on Receipt: Attribute creation

If the Attribute is not already associated to the file identified by its File Full Path, the attribute shall be created with the information provided in the Attribute parameter and associated to the file.

Verification Method: T

SAVOIR.MMS.FMS.2530

SET_FILE_ATTRIBUTE.request Effect on Receipt: Attribute update

If an attribute with same identifier is already associated to the file identified by its File Full Path, the attribute shall be updated with the information provided in the Attribute parameter.

Comment: Some restriction may apply in some implementations related the restriction of updates to Attribute with the same Attribute Type and Attribute Size.

Verification Method: T

8.3.3.19.5 Additional constraints

SAVOIR.MMS.FMS.2540

SET_FILE_ATTRIBUTE.request Additional constraints: Maximum number of attributes

The maximum number of Attributes that can be associated per file shall be configured per File system.

Comment: This is mainly related to the used File System limitations and defined in the File System characteristics.

Verification Method: T

8.3.3.20 SET_FILE_ATTRIBUTE.indication

8.3.3.20.1 Function

SAVOIR.MMS.FMS.2550

SET_FILE_ATTRIBUTE.indication Function

The SET_FILE_ATTRIBUTE.indication primitive shall be used to pass the outcome of associating an attribute to a file to the User Entity.

Verification Method: T

8.3.3.20.2 Semantics

SAVOIR.MMS.FMS.2560

SET_FILE_ATTRIBUTE.indication Semantics

The SET_FILE_ATTRIBUTE.indication primitive shall use the following semantics:
SET_FILE_ATTRIBUTE.indication(File Transaction Identifier, File Result Metadata)

Verification Method: T

8.3.3.20.3 When generated

SAVOIR.MMS.FMS.2570

SET_FILE_ATTRIBUTE.indication When generated

The SET_FILE_ATTRIBUTE.indication primitive shall be passed by the FS provider to the receiving User Entity in response to a SET_FILE_ATTRIBUTE.request with File Result Metadata indicating if the operation was successful or not.

Verification Method: T

SAVOIR.MMS.FMS.2580

SET_FILE_ATTRIBUTE.indication When generated failure

When SET_FILE_ATTRIBUTE.request is unsuccessful, the File Result Metadata shall provide the reason of the failure.

Verification Method: T

8.3.3.20.4 Effect on Receipt

The response of the User Entity to a SET_FILE_ATTRIBUTE.indication is unspecified.

8.3.3.20.5 Additional constraints

SAVOIR.MMS.FMS.2590

SET_FILE_ATTRIBUTE.indication Additional constraints: File does not exist

The SET_FILE_ATTRIBUTE.indication File Result Metadata shall report a failure if the requested File Full Path refers to an inexistent file.

Verification Method: T

SAVOIR.MMS.FMS.2600

SET_FILE_ATTRIBUTE.indication Additional constraints: Too many attributes

The SET_FILE_ATTRIBUTE.indication File Result Metadata shall report a failure if the Attribute Name is not already associated to the file and the maximum number of attributes per file has been reached (i.e. insertion cannot be achieved).

Comment: Updating the value of an already existing/associated attribute remains possible.

Verification Method: T

SAVOIR.MMS.FMS.2610

SET_FILE_ATTRIBUTE.indication Additional constraints: Bad Type

The SET_FILE_ATTRIBUTE.indication Metadata shall report a failure if the requested Attribute Type is not supported.

Verification Method: T

8.3.3.21 GET_FILE_ATTRIBUTES.request

8.3.3.21.1 Function

SAVOIR.MMS.FMS.2620

GET_FILE_ATTRIBUTES.request Function

The GET_FILE_ATTRIBUTES.request primitive shall be passed to the FMS provider to request reporting the collection of attributes currently associated to the specified file.

Verification Method: T

8.3.3.21.2 Semantics

SAVOIR.MMS.FMS.2630

GET_FILE_ATTRIBUTES.request Semantics

The GET_FILE_ATTRIBUTES.request primitive shall use the following semantics:
GET_FILE_ATTRIBUTES.request(Transaction Identifier, File Full Path)

Verification Method: T

8.3.3.21.3 When generated

SAVOIR.MMS.FMS.2640

GET_FILE_ATTRIBUTES.request When generated

The GET_FILE_ATTRIBUTES.request primitive shall be passed to the FS provider to request listing the attributes (with their respective value and type) currently associated to the file identified by File Full Path.

Verification Method: T

8.3.3.21.4 Effect on Receipt

SAVOIR.MMS.FMS.2650

GET_FILE_ATTRIBUTES.request Effect on Receipt

Receipt of the GET_FILE_ATTRIBUTES.request primitive shall cause the FS provider to get all the attributes currently associated to the file identified by File Full Path.

Verification Method: T

8.3.3.21.5 Additional constraints

None.

8.3.3.22 GET_FILE_ATTRIBUTES.indication

8.3.3.22.1 Function

SAVOIR.MMS.FMS.2660

GET_FILE_ATTRIBUTES.indication Function

The GET_FILE_ATTRIBUTES.indication primitive shall be used to pass the collection of attributes associated to a file to the User Entity.

Verification Method: T

8.3.3.22.2 Semantics

SAVOIR.MMS.FMS.2670

GET_FILE_ATTRIBUTES.indication Semantics

The GET_FILE_ATTRIBUTES.indication primitive shall use the following semantics:
GET_FILE_ATTRIBUTES.indication(File Transaction Identifier, Attribute List, File Result Metadata)

Verification Method: T

8.3.3.22.3 When generated

SAVOIR.MMS.FMS.2680

GET_FILE_ATTRIBUTES.indication When generated

The GET_FILE_ATTRIBUTES.indication primitive shall be passed by the FS provider to the receiving User Entity in response to a GET_FILE_ATTRIBUTES.request with File Result Metadata indicating if the operation was successful or not.

Verification Method: T

SAVOIR.MMS.FMS.2690

GET_FILE_ATTRIBUTES.indication When generated success

When GET_FILE_ATTRIBUTES.request is successful, the Attribute List parameter shall contain the complete list of attributes associated to the file.

Verification Method: T

SAVOIR.MMS.FMS.2700

GET_FILE_ATTRIBUTES.indication When generated failure

When GET_FILE_ATTRIBUTES.request is unsuccessful, the File Result Metadata shall provide the reason of the failure.

Verification Method: T

8.3.3.22.4 Effect on Receipt

The response of the User Entity to a GET_FILE_ATTRIBUTES.indication is unspecified.

8.3.3.22.5 Additional constraints

SAVOIR.MMS.FMS.2710

GET_FILE_ATTRIBUTES.indication Additional constraints: File does not exist

The GET_FILE_ATTRIBUTES.indication File Result Metadata shall report a failure if the requested File Full Path refers to an inexistent file.

Verification Method: T

8.3.3.23 FORCE_FILE_SYNCH.request

8.3.3.23.1 Function

SAVOIR.MMS.FMS.2720

FORCE_FILE_SYNCH.request Function

The FORCE_FILE_SYNCH.request primitive shall be passed to the FS provider to request immediate synchronization of the specified file with the storage media.

Rationale: *To ensure that data are written on the Storage Media in case cache mechanisms have been introduced for performance reasons.*

Comment: *This primitive is always implicitly called when closing a file.*

Verification Method: T

8.3.3.23.2 Semantics

SAVOIR.MMS.FMS.2730

FORCE_FILE_SYNCH.request Semantics

The FORCE_FILE_SYNCH.request primitive shall use the following semantics:
FORCE_FILE_SYNCH.request(File Transaction Identifier, File Descriptor)

Verification Method: T

8.3.3.23.3 When generated

SAVOIR.MMS.FMS.2740

FORCE_FILE_SYNCH.request When generated

The FORCE_FILE_SYNCH.request primitive shall be passed to the FS provider to request immediately synchronizing content and status information of an opened file to the storage media.

Verification Method: T

8.3.3.23.4 Effect on Receipt

SAVOIR.MMS.FMS.2750

FORCE_FILE_SYNCH.request Effect on Receipt

Receipt of the FORCE_FILE_SYNCH.request primitive shall cause the FS provider to write to the storage media any modification applied to the specified opened file that have not yet been written.

Verification Method: T

8.3.3.23.5 Additional constraints

SAVOIR.MMS.FMS.2760

FORCE_FILE_SYNCH.request Additional constraints

File synchronization to the storage media shall be supported only for files opened for reading and writing.

Comment: Opened for reading and writing indicates that the "Read-Write" Access Type was selected in File Opening Criteria. "Read" Access Type is excluded as there is nothing to synchronize to the storage media when only reading. Forced synchronization with "Append" Access Type is voluntarily excluded as synchronization is part of the FS internal process to support concurrent accesses.

Verification Method: T

8.3.3.24 FORCE_FILE_SYNCH.indication

8.3.3.24.1 Function

SAVOIR.MMS.FMS.2770

FORCE_FILE_SYNCH.indication Function

The FORCE_FILE_SYNCH.indication primitive shall be used to pass the outcome of synchronizing the specified file to the User Entity.

Verification Method: T

8.3.3.24.2 Semantics

SAVOIR.MMS.FMS.2780

FORCE_FILE_SYNCH.indication Semantics

The FORCE_FILE_SYNCH.indication primitive shall use the following semantics:
FORCE_FILE_SYNCH.indication(File Transaction Identifier, File Result Metadata)

Verification Method: T

8.3.3.24.3 When generated

SAVOIR.MMS.FMS.2790

FORCE_FILE_SYNCH.indication When generated

The FORCE_FILE_SYNCH.indication primitive shall be passed by the FS provider to the receiving User Entity in response to a FORCE_FILE_SYNCH.request with File Result Metadata indicating if the request was executed successfully or not, and the reason in case of failure.

Verification Method: T

8.3.3.24.4 Effect on Receipt

The response of the User Entity to a FORCE_FILE_SYNCH.indication is unspecified.

8.3.3.24.5 Additional constraints

SAVOIR.MMS.FMS.2800

FORCE_FILE_SYNCH.indication Additional constraints: File Descriptor not valid

The FORCE_FILE_SYNCH.indication Result Metadata shall report a failure if the requested File Descriptor is not valid.

Verification Method: T

SAVOIR.MMS.FMS.2810

FORCE_FILE_SYNCH.indication Additional constraints: File access error

The FORCE_FILE_SYNCH.indication Result Metadata shall report a failure if the requested File Descriptor does not correspond to a file open for writing (Read-Write access type).

Verification Method: T

8.3.3.25 GET_FILE_CHECKSUM.request

8.3.3.25.1 Function

SAVOIR.MMS.FMS.2820

GET_FILE_CHECKSUM Function

The GET_FILE_CHECKSUM.request primitive shall be passed to the FS provider to request computing and reporting the checksum of the specified file.

Comment: The checksum function to be used is implementation dependent (for example it can be a CRC - Cyclic Redundancy Check).

Verification Method: T

8.3.3.25.2 Semantics

SAVOIR.MMS.FMS.2830

GET_FILE_CHECKSUM Semantics

The GET_FILE_CHECKSUM.request primitive shall use the following semantics:
GET_FILE_CHECKSUM.request(File Transaction Identifier, File Full Path)

Verification Method: T

8.3.3.25.3 When generated

SAVOIR.MMS.FMS.2840

GET_FILE_CHECKSUM When generated

The GET_FILE_CHECKSUM.request primitive shall be passed to the FS provider to request file checksum to be computed.

Verification Method: T

8.3.3.25.4 Effect on Receipt

SAVOIR.MMS.FMS.2850

GET_FILE_CHECKSUM Effect on Receipt

Receipt of the GET_FILE_CHECKSUM.request primitive shall cause the FS provider to compute the checksum of the specified file by applying a checksum function on its content.

Comment: The checksum function is defined at mission level.

Verification Method: T

8.3.3.25.5 Additional constraints

SAVOIR.MMS.FMS.2860

GET_FILE_CHECKSUM Additional constraints: File not modified

The GET_FILE_CHECKSUM.request primitive shall be rejected if the file is currently opened with File Access Type Read-Write or Append by at least one User Entity.

Rationale: *To ensure that content cannot be modified during the checksum computation.*

Verification Method: T

SAVOIR.MMS.FMS.2870

GET_FILE_CHECKSUM Additional constraints: Checksum abort

An in-progress checksum computation shall be immediately aborted and the User Entity notified when the file for which checksum is computed is opened with File Access Type Read-Write or Appen by any User Entity.

Rationale: *Opening the file for writing has a higher precedence and interrupts the computation process.*

Comment: *Read opening of the file has no impact on checksum computation and is thus allowed.*

Verification Method: T

8.3.3.26 GET_FILE_CHECKSUM.indication

8.3.3.26.1 Function

SAVOIR.MMS.FMS.2880

GET_FILE_CHECKSUM.indication Function

The GET_FILE_CHECKSUM.indication primitive shall be used to pass the computed file checksum or notify the User Entity about an abort.

Verification Method: T

8.3.3.26.2 Semantics

SAVOIR.MMS.FMS.2890

GET_FILE_CHECKSUM.indication Semantics

The GET_FILE_CHECKSUM.indication primitive shall use the following semantics:
GET_FILE_CHECKSUM.indication(File Transaction Identifier, File Checksum, File Result Metadata)

Verification Method: T

8.3.3.26.3 When generated

SAVOIR.MMS.FMS.2900

GET_FILE_CHECKSUM.indication When generated

The GET_FILE_CHECKSUM.indication primitive shall be passed by the FS provider to the receiving User Entity in response to a GET_FILE_CHECKSUM.request with File Result Metadata indicating if the request was executed successfully or not.

Comment: The indication is not immediately generated, and the delay depends on the file size and the checksum function.

Verification Method: T

SAVOIR.MMS.FMS.2910

GET_FILE_CHECKSUM.indication When generated success

When GET_FILE_CHECKSUM.indication primitive is successful, the service returned the result of the checksum computation in the File Checksum parameter.

Verification Method: T

SAVOIR.MMS.FMS.2920

GET_FILE_CHECKSUM.indication When generated failure

When GET_FILE_CHECKSUM.indication primitive is unsuccessful, the File Result Metadata shall provide the reason of the failure..

Verification Method: T

8.3.3.26.4 Effect on Receipt

The response of the User Entity to a GET_FILE_CHECKSUM.indication is unspecified.

8.3.3.26.5 Additional constraints

SAVOIR.MMS.FMS.2930

GET_FILE_CHECKSUM.indication Additional constraints: File does not exist

The GET_FILE_CHECKSUM.indication File Result Metadata shall report a failure if the requested File Full Path refers to an inexistent file.

Verification Method: T

SAVOIR.MMS.FMS.2940

GET_FILE_CHECKSUM.indication Additional constraints: File already opened for writing

The GET_FILE_CHECKSUM.indication File Result Metadata shall report a failure if the requested file is currently open for writing by at least one User Entity.

Verification Method: T

SAVOIR.MMS.FMS.2950

GET_FILE_CHECKSUM.indication Additional constraints: File opened for writing

The GET_FILE_CHECKSUM.indication File Result Metadata shall report a failure in case the file is opened for writing (by any User Entity) while the checksum is being computed.

Verification Method: T

8.3.3.27 DELETE_FILE.request

8.3.3.27.1 Function

SAVOIR.MMS.FMS.2960

DELETE_FILE.request Function

The DELETE.request primitive shall be passed to the FS provider to request deletion of an existing file in a File Store.

Verification Method: T

8.3.3.27.2 Semantics

SAVOIR.MMS.FMS.2970

DELETE_FILE.request Semantics

The DELETE_FILE.request primitive shall use the following semantics:
DELETE_FILE.request(File Transaction Identifier, File Full Path, Force)

Verification Method: T

8.3.3.27.3 When generated

SAVOIR.MMS.FMS.2980

DELETE_FILE.request When generated

The DELETE_FILE.request primitive shall be passed to the FS provider to request the deletion of the specified file.

Verification Method: T

8.3.3.27.4 Effect on Receipt

SAVOIR.MMS.FMS.2990

DELETE_FILE.request Effect on Receipt

Receipt of the DELETE_FILE.request primitive shall cause the FS provider to delete the specified file.

Verification Method: T

8.3.3.27.5 Additional constraints

SAVOIR.MMS.FMS.3000

DELETE_FILE.request Additional constraints: File opened

The deletion shall be rejected in case the specified file is currently opened by at least one User Entity, unless Force parameter is set.

Comment: The Force parameter allows bypassing the check on file open status. The User Entity forcing deletion of an open file must be aware that it may cause troubles to user entities that were using it (i.e. owning a File Descriptor to it).

Verification Method: T

SAVOIR.MMS.FMS.3010

DELETE_FILE.request Additional constraints: File locked by other

The deletion shall be rejected in case a lock is applied on the specified file and the owner of the lock is not the User Entity at the origin of the deletion request.

Comment: A file can be locked without being currently opened

Verification Method: T

SAVOIR.MMS.FMS.3020

DELETE_FILE.request Additional constraints: File locked read-only

The deletion shall be rejected in case a read-only lock is applied on the requested file.

Comment: This covers the case where the User Entity requesting the deletion is the locker-owner. Locking a file in read-only restricts the access to its content to reading operations, and thus prevents alteration. However, if the file is locked for writing (either Exclusive_Access or Single_Writer Lock Type), and the lock-owner is the User Entity at the origin of the request deletion, the file can be deleted.

Verification Method: T

SAVOIR.MMS.FMS.3030

DELETE_FILE.request Additional constraints: File protection

No operation shall be allowed on the file during the deletion operation.

Comment: Any operation requested on the file shall be rejected (opening for reading, listing attributes, etc). This applies to all User Entities.

Verification Method: T

8.3.3.28 DELETE_FILE.indication

8.3.3.28.1 Function

SAVOIR.MMS.FMS.3040

DELETE_FILE.indication Function

The DELETE_FILE.indication primitive shall be used to pass the outcome of deleting a file to the User Entity.

Verification Method: T

8.3.3.28.2 Semantics

SAVOIR.MMS.FMS.3050

DELETE_FILE.indication Semantics

The DELETE_FILE.indication primitive shall use the following semantics:
DELETE_FILE.indication(Transaction Identifier, Result Metadata)

Verification Method: T

8.3.3.28.3 When generated

SAVOIR.MMS.FMS.3060

DELETE_FILE.indication When generated

The DELETE_FILE.indication primitive shall be passed by the FS provider to the receiving User Entity in response to a DELETE_FILE.request with File Result Metadata indicating if the operation was successful or not..

Comment: The primitive provides File Result Metadata indicating if the request was executed successfully or not, and the reason in case of failure.

Verification Method: T

SAVOIR.MMS.FMS.3070

DELETE_FILE.indication When generated failure

When DELETE_FILE.request is unsuccessful, the File Result Metadata shall provide the reason of the failure.

Verification Method: T

8.3.3.28.4 Effect on Receipt

The response of the User Entity to a DELETE_FILE.indication is unspecified.

8.3.3.28.5 Additional constraints

SAVOIR.MMS.FMS.3080

DELETE_FILE.indication Additional constraints: File does not exist

The DELETE_FILE.indication File Result Metadata shall report a failure if the requested Path refers to an inexistent file.

Verification Method: T

SAVOIR.MMS.FMS.3090

DELETE_FILE.indication Additional constraints: File opened

The DELETE_FILE.indication Result Metadata shall report a failure if the file is currently open and Force parameter was not set.

Verification Method: T

SAVOIR.MMS.FMS.3100

DELETE_FILE.indication Additional constraints: File locked by other

The DELETE_FILE.indication Result Metadata shall report a failure if a lock is applied on the file and the lock-owner is not the User Entity requesting deletion.

Verification Method: T

SAVOIR.MMS.FMS.3110

DELETE_FILE.indication Additional constraints: File locked

The DELETE_FILE.indication Result Metadata shall report a failure if a read-only lock is applied on the file.

Verification Method: T

8.3.3.29 COPY_FILE.request

8.3.3.29.1 Function

SAVOIR.MMS.FMS.3120

COPY_FILE.request Function

The COPY_FILE.request primitive shall be passed to the FS provider to request copying an existing file from one specified directory to another specified one, either within the same File Store, between local File Stores, or between local and remote File Stores on-board.

Comment: This requirement covers file copy in both "directions" (e.g. it is possible to request a file to be copied to or from a remote File Store).
The file may be copied within the same directory, given that a different name is provided for the destination one.

Verification Method: T

8.3.3.29.2 Semantics

SAVOIR.MMS.FMS.3130

COPY_FILE.request Semantics

The COPY_FILE.request primitive shall use the following semantics:
COPY_FILE.request(File Transaction Identifier, Source File Full Path, Destination File Full Path, Maximum File Size (optional))

Verification Method: T

8.3.3.29.3 When generated

SAVOIR.MMS.FMS.3140

COPY_FILE.request When generated

The COPY_FILE.request primitive shall be passed to the FS provider to request the copy of the specified existing source file to the specified destination one to be created.

Verification Method: T

8.3.3.29.4 Effect on Receipt

SAVOIR.MMS.FMS.3150

COPY_FILE.request Effect on Receipt: Destination File Creation

Receipt of the COPY_FILE.request primitive shall cause the FS provider to first create the destination file with the requested Maximum File Size (optional) or the one inherited from the source file.

Comment: This can be implemented by CREATE_FILE.request and related requirements apply.

Verification Method: T

SAVOIR.MMS.FMS.3160

COPY_FILE.request Effect on Receipt: File Attribute Copy

The attributes of the source file shall be duplicated to the destination file.

Comment: The information related to File Status of the source file (lock information, mapping, etc.) are not inherited by the destination file.

Verification Method: T

SAVOIR.MMS.FMS.3170

COPY_FILE.request Effect on Receipt File content Copy

The content of the source file shall be copied until end of source file.

Comment: Copy of the content stops when all the data contained in the source file are copied.

Verification Method: T

8.3.3.29.5 Additional constraints

SAVOIR.MMS.FMS.3180

COPY_FILE.request Additional constraints: Source file access

The Source File Full Path parameter shall refer to an existing file currently accessible for reading.

Comment: Reading access is mandatory to copy the content.

Verification Method: T

SAVOIR.MMS.FMS.3190

COPY_FILE.request Additional constraints: Source and Destination different

The Source File Full Path and Destination File Full Path parameters shall not be identical.

Comment: The File Name of source and destination may be identical, as long as they have different parent directories (file name unique within a directory).

Verification Method: T

SAVOIR.MMS.FMS.3200

COPY_FILE.request Additional constraints: File already exists

The Destination File Full Path parameter shall not refer to an already existing file.

Comment: No overwrite allowed for a copy operation, the destination file is being created during the procedure.

Verification Method: T

Parent: SAVOIR-DSS-ORG-600

SAVOIR.MMS.FMS.3210

COPY_FILE.request Additional constraints: Destination directory does not exist

The Destination File Full Path shall only contain existing directories.

Comment: The copy procedure shall not have to create any directory.

Verification Method: T

Parent: SAVOIR-DSS-ORG-600

SAVOIR.MMS.FMS.3220

COPY_FILE.request Additional constraints: Destination size smaller

When specified, the optional Maximum File Size parameter (applicable to the destination) shall be greater than or equal to the source file's Maximum File Size.

Comment: No File truncation that would lead to loss of data is supported.

Verification Method: T

SAVOIR.MMS.FMS.3230

COPY_FILE.request Additional constraints: Destination available space

The copy operation shall be rejected if the Maximum File Size to be applied to the destination file (either inherited from the source file or provided as parameter) is greater than the limit imposed by the destination File Store FS.

Comment: *At the very minimum, the size limit supported by the destination File Store FS must be equal to the source file's Maximum File Size for the copy to be possible.*

Verification Method: T

Parent: SAVOIR-DSS-ORG-600

SAVOIR.MMS.FMS.3240

COPY_FILE.request Additional constraints: File copy states

The file copy operation shall be managed by a State Machine including the following states: STARTED, SUSPENDED, COMPLETED, ABORTED.

Verification Method: T

SAVOIR.MMS.FMS.3250

COPY_FILE.request Additional constraints: File copy state STARTED

When initiated (i.e. not rejected by the FS), a copy operation, the File Copy State Machine shall immediately enter the STARTED state.

Rationale: *This is the entry point of the State Machine.*

Verification Method: T

SAVOIR.MMS.FMS.3260

COPY_FILE.request Additional constraints: File copy suspending

The File Copy State Machine shall only authorize suspending a copy currently in STARTED state.

Comment: *Requesting suspending a copy in any other state must be rejected.*

Verification Method: T

SAVOIR.MMS.FMS.3270

COPY_FILE.request Additional constraints: File copy resuming

The File Copy State Machine shall only authorize resuming a copy currently in SUSPENDED state.

Comment: *Requesting resuming a copy in any other state must be rejected.*

Verification Method: T

SAVOIR.MMS.FMS.3280

COPY_FILE.request Additional constraints: File copy aborting

The Copy State Machine shall only authorize aborting a copy currently in STARTED or SUSPENDED state.

Comment: Requesting aborting a copy in any other state must be rejected.

Verification Method: T

SAVOIR.MMS.FMS.3290

COPY_FILE.request Additional constraints: File copy aborted

The Copy State Machine shall move to ABORTED state when abort is authorized.

Verification Method: T

SAVOIR.MMS.FMS.3300

COPY_FILE.request Additional constraints: File copy completion

An initiated copy operation shall immediately enter the COMPLETED state when end-of file (EOF) of the source file or the File Maximum Size is reached.

Verification Method: T

8.3.3.30 COPY_FILE.indication

8.3.3.30.1 Function

SAVOIR.MMS.FMS.3310

COPY_FILE.indication Function

The COPY_FILE.indication primitive shall be used to pass the outcome of initiating a file copy to the User Entity.

Comment: This primitive is not intended to provide the result of the copy operation, this is achieved by the FILE_COPY_EVENT.indication primitive.

Verification Method: T

8.3.3.30.2 Semantics

SAVOIR.MMS.FMS.3320

COPY_FILE.indication Semantics

The COPY_FILE.indication primitive shall use the following semantics:
COPY_FILE.indication(File Transaction Identifier, File Copy Identifier, File Result Metadata)

Verification Method: T

8.3.3.30.3 When generated

SAVOIR.MMS.FMS.3330

COPY_FILE.indication When generated

The COPY_FILE.indication primitive shall be passed by the FS provider to the receiving User Entity in response to a COPY_FILE.request with File Result Metadata indicating if the request was executed successfully or not, and the reason in case of failure.

Comment: The primitive provides the unique identifier allocated to the copy operation (if initiated) and File Result Metadata indicating if the request was executed successfully or not, and the reason in case of failure.

Verification Method: T

8.3.3.30.4 Effect on Receipt

The response of the User Entity to a COPY_FILE.indication is unspecified.

8.3.3.30.5 Additional constraints

SAVOIR.MMS.FMS.3340

COPY_FILE.indication Additional constraints: Source does not exist

The COPY_FILE.indication File Result Metadata shall report a failure if the requested Source File Full Path refers to an inexistent file

Verification Method: T

SAVOIR.MMS.FMS.3350

COPY_FILE.indication Additional constraints: Source locked

The COPY_FILE.indication File Result Metadata shall report a failure if the requested Source File Full Path is not accessible for reading (locked).

Verification Method: T

SAVOIR.MMS.FMS.3360

COPY_FILE.indication Additional constraints: Source and destination identical

The COPY_FILE.indication File Result Metadata shall report a failure if the requested Source File Full Path and Destination File Full Path are identical.

Verification Method: T

SAVOIR.MMS.FMS.3370

COPY_FILE.indication Additional constraints: Destination directory does not exist

The COPY_FILE.indication File Result Metadata shall report a failure if the requested Destination File Full Path refers to an already existing file.

Verification Method: T

SAVOIR.MMS.FMS.3380

COPY_FILE.indication Additional constraints: Directory does not exist

The COPY_FILE.indication File Result Metadata shall report a failure if the Destination File Full Path contains one directory that does not exist.

Verification Method: T

SAVOIR.MMS.FMS.3390

COPY_FILE.indication Additional constraints: Maximum File Size too low

The COPY_FILE.indication File Result Metadata shall report a failure if the requested Maximum File Size is lower than the source file one.

Comment: Requirement

Verification Method: T

SAVOIR.MMS.FMS.3400

COPY_FILE.indication Additional constraints: Too many copy operations

The COPY_FILE.indication File Result Metadata shall report a failure if the threshold of maximum concurrent on-going file copy operations defined by <*FMS maximum File Copy Status*> has been reached.

Rationale: The management of a copy operation requires resources and the number of concurrent execution can be limited.

Comment: In this case the requesting User Entity has to wait for completion or abort of an on-going copy.
The threshold is to be defined to comply with mission needs.
File copies that may be triggered (internally) by MOVE_FILE.request (cf. 2.5.3.31.3) are also recorded (i.e. considered as on-going file copy operations).

Verification Method: T

8.3.3.31 SUSPEND_FILE_COPY.request

8.3.3.31.1 Function

SAVOIR.MMS.FMS.3410

SUSPEND_FILE_COPY.request Function

The SUSPEND_FILE_COPY.request primitive shall be passed to the FS provider to request the immediate suspension of an on-going file copy operation.

Comment: On-going file copy operation means that a valid Copy Identifierwas returned by the COPY_FILE.indication (copy initiated), and the copy has not already completed or been aborted.

Verification Method: T

8.3.3.31.2 Semantics

SAVOIR.MMS.FMS.3420

SUSPEND_FILE_COPY.request Semantics

The SUSPEND_FILE_COPY.request primitive shall use the following semantics:
SUSPEND_FILE_COPY.request(File Transaction Identifier, File Copy Identifier)

Verification Method: T

8.3.3.31.3 When generated

SAVOIR.MMS.FMS.3430

SUSPEND_FILE_COPY.request When generated

The SUSPEND_FILE_COPY.request primitive shall be passed to the FS provider to request an on-going file copy operation to be suspended immediately.

Comment: Suspension is for an undetermined duration.

Verification Method: T

8.3.3.31.4 Effect on Receipt

SAVOIR.MMS.FMS.3440

SUSPEND_FILE_COPY.request Effect on Receipt

Receipt of the SUSPEND_FILE_COPY.request primitive shall cause the FS provider to suspend the identified file copy operation and move corresponding state machine to SUSPENDED state.

Verification Method: T

8.3.3.31.5 Additional constraints

None.

8.3.3.32 SUSPEND_FILE_COPY.indication

8.3.3.32.1 Function

SAVOIR.MMS.FMS.3450

SUSPEND_FILE_COPY.indication Function

The SUSPEND_FILE_COPY.indication primitive shall be used to pass the outcome of suspending a file copy to the User Entity.

Verification Method: T

8.3.3.32.2 Semantics

SAVOIR.MMS.FMS.3460

SUSPEND_FILE_COPY.indication Semantics

The SUSPEND_FILE_COPY.indication primitive shall use the following semantics:
SUSPEND_FILE_COPY.indication(File Transaction Identifier, File Result Metadata)

Verification Method: T

8.3.3.32.3 When generated

SAVOIR.MMS.FMS.3470

SUSPEND_FILE_COPY.indication When generated

The SUSPEND_FILE_COPY.indication primitive shall be passed by the FS provider to the receiving User Entity in response to a SUSPEND_FILE_COPY.request with File Result Metadata indicating if the request was executed successfully or not.

Verification Method: T

SAVOIR.MMS.FMS.3480

SUSPEND_FILE_COPY.indication When generated failure

When SUSPEND_FILE_COPY.request is unsuccessful, the File Result Metadata shall provide the reason of the failure.

Verification Method: T

8.3.3.32.4 Effect on Receipt

The response of the User Entity to a SUSPEND_FILE_COPY.indication is unspecified.

8.3.3.32.5 Additional constraints

SAVOIR.MMS.FMS.3490

SUSPEND_FILE_COPY.indication Additional constraints: Invalid identifier

The SUSPEND_FILE_COPY.indication File Result Metadata shall report a failure if the requested File Copy Identifier is invalid (i.e. does not refer to an already initiated copy operation).

Verification Method: T

SAVOIR.MMS.FMS.3500

SUSPEND_FILE_COPY.indication Additional constraints: Copy not started

The SUSPEND_FILE_COPY.indication Result Metadata shall report a failure if the state of the requested Copy Identifier is not STARTED.

Verification Method: T

8.3.3.33 RESUME_FILE_COPY.request

8.3.3.33.1 Function

SAVOIR.MMS.FMS.3510

RESUME_FILE_ COPY.request Function

The RESUME_FILE_COPY.request primitive shall be passed to the FS provider to request the immediate resuming of a suspended file copy operation.

Comment: Suspended means that the copy operation is in SUSPENDED state.

Verification Method: T

8.3.3.33.2 Semantics

SAVOIR.MMS.FMS.3520

RESUME_FILE_ COPY.request Semantics

The RESUME_FILE_COPY.request primitive shall use the following semantics:
RESUME_FILE_COPY.request(File Transaction Identifier, File Copy Identifier)

Verification Method: T

8.3.3.33.3 When generated

SAVOIR.MMS.FMS.3530

RESUME_FILE_COPY.request When generated

The RESUME_FILE_COPY.request primitive shall be passed to the FS provider to request a suspended file copy operation to be resumed immediately.

Verification Method: T

8.3.3.33.4 Effect on Receipt

SAVOIR.MMS.FMS.3540

RESUME_FILE_COPY.request Effect on Receipt

Receipt of the RESUME_FILE_COPY.request primitive shall cause the FS provider to resume the identified file copy operation and move to STARTED state.

Verification Method: T

8.3.3.33.5 Additional constraints

None.

8.3.3.34 RESUME_FILE_COPY.indication

8.3.3.34.1 Function

SAVOIR.MMS.FMS.3550

RESUME_FILE_COPY.indication Function

The RESUME_FILE_COPY.indication primitive shall be used to pass the outcome of resuming a file copy to the User Entity.

Verification Method: T

8.3.3.34.2 Semantics

SAVOIR.MMS.FMS.3560

RESUME_FILE_COPY.indication Semantics

The RESUME_FILE_COPY.indication primitive shall use the following semantics:
RESUME_FILE_COPY.indication(File Transaction Identifier, File Result Metadata)

Verification Method: T

8.3.3.34.3 When generated

SAVOIR.MMS.FMS.3570

RESUME_FILE_COPY.indication When generated

The RESUME_FILE_COPY.indication primitive shall be passed by the FMS provider to the receiving User Entity in response to a RESUME_FILE_COPY.request with File Result Metadata indicating if the request was executed successfully or not.

Verification Method: T

8.3.3.34.4 Effect on Receipt

The response of the User Entity to a RESUME_FILE_COPY.indication is unspecified.

8.3.3.34.5 Additional constraints

SAVOIR.MMS.FMS.3580

RESUME_FILE_COPY.indication Additional constraints: Invalid identifier

The RESUME_FILE_COPY.indication File Result Metadata shall report a failure if the requested File Copy Identifier is invalid (i.e. does not refer to an already initiated copy operation).

Verification Method: T

SAVOIR.MMS.FMS.3590

RESUME_FILE_COPY.indication Additional constraints: Copy not suspended

The RESUME_FILE_COPY.indication File Result Metadata shall report a failure if the state of the requested Copy Identifier is not SUSPENDED.

Verification Method: T

8.3.3.35 ABORT_FILE_COPY.request

8.3.3.35.1 Function

SAVOIR.MMS.FMS.3600

ABORT_FILE_COPY.request Function

The ABORT_FILE_COPY.request primitive shall be passed to the FS provider to request the immediate abort of an on-going file copy operation.

Comment: On-going file copy operation means that a valid File Copy Identifier was returned by the COPY_FILE.indication (copy initiated), and the copy has not already completed or been aborted.

Verification Method: T

8.3.3.35.2 Semantics

SAVOIR.MMS.FMS.3610

ABORT_FILE_COPY.request Semantics

The ABORT_FILE_COPY.request primitive shall use the following semantics:
ABORT_FILE_COPY.request(File Transaction Identifier, File Copy Identifier)

Verification Method: RoD

8.3.3.35.3 When generated

SAVOIR.MMS.FMS.3620

ABORT_FILE_COPY.request When generated

The ABORT_FILE_COPY.request primitive shall be passed to the FS provider to request an on-going file copy operation to be aborted immediately.

Verification Method: T

8.3.3.35.4 Effect on Receipt

SAVOIR.MMS.FMS.3630

ABORT_FILE_COPY.request Effect on Receipt File copy abort

Receipt of the ABORT_FILE_COPY.request primitive shall cause the FS provider to abort the identified file copy operation and move to ABORTED state.

Comment: Abort means definitive stop without possible resuming.

Verification Method: T

SAVOIR.MMS.FMS.3640

ABORT_FILE_COPY.request Effect on Receipt Destination delete

Receipt of the ABORT_FILE_COPY.request primitive shall cause the FS provider to delete the destination.

Rationale: Destination file is incomplete.

Verification Method: T

8.3.3.35.5 Additional constraints

None.

8.3.3.36 ABORT_FILE_COPY.indication

SAVOIR.MMS.FMS.3650

ABORT_FILE_COPY.indication Function

The ABORT_FILE_COPY.indication primitive shall be used to pass the outcome of aborting a file copy to the User Entity.

Verification Method: T

8.3.3.36.1 Semantics

SAVOIR.MMS.FMS.3660

ABORT_FILE_COPY.indication Semantics

The ABORT_FILE_COPY.indication primitive shall use the following semantics:
ABORT_FILE_COPY.indication(File Transaction Identifier, File Result Metadata)

Verification Method: T

8.3.3.36.2 When generated

SAVOIR.MMS.FMS.3670

ABORT_FILE_COPY.indication When generated

The ABORT_FILE_COPY.indication primitive shall be passed by the FS provider to the receiving User Entity in response to an ABORT_FILE_COPY.request with File Result Metadata indicating if the request was executed successfully or not.

Verification Method: T

SAVOIR.MMS.FMS.3680

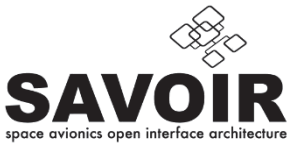
ABORT_FILE_COPY.indication When generated failure

When ABORT_FILE_COPY.request is unsuccessful, the File Result Metadata shall provide the reason of the failure.

Verification Method: T

8.3.3.36.3 Effect on Receipt

The response of the User Entity to an ABORT_FILE_COPY.indication is unspecified.



8.3.3.36.4 Additional constraints

SAVOIR.MMS.FMS.3690

ABORT_FILE_COPY.indication Additional constraints: Invalid identifier

The ABORT_FILE_COPY.indication File Result Metadata shall report a failure if the requested File Copy Identifier is invalid (i.e. does not refer to an already initiated copy operation).

Verification Method: T

SAVOIR.MMS.FMS.3700

ABORT_FILE_COPY.indication Additional constraints: Invalid state

The ABORT_FILE_COPY.indication File Result Metadata shall report a failure if the state of the requested File Copy Identifier is COMPLETED or ABORTED.

Comment: When a copy has completed or been aborted, no more operation is possible on it. The copy status history of an operation is maintained for a duration detailed in section 8.3.3.37.5.

Verification Method: T

8.3.3.37 GET_FILE_COPY_STATUS.request

8.3.3.37.1 Function

SAVOIR.MMS.FMS.3710

GET_FILE_COPY_STATUS.request Function

The GET_FILE_COPY_STATUS.request primitive shall be passed to the FS provider to request reporting the current status of a file copy operation.

Verification Method: T

8.3.3.37.2 Semantics

SAVOIR.MMS.FMS.3720

GET_FILE_COPY_STATUS.request Semantics

The GET_FILE_COPY_STATUS.request primitive shall use the following semantics:
GET_FILE_COPY_STATUS.request(FileTransaction Identifier, File Copy Identifier)

Verification Method: T

8.3.3.37.3 When generated

SAVOIR.MMS.FMS.3730

GET_FILE_COPY_STATUS.request When generated

The GET_FILE_COPY_STATUS.request primitive shall be passed to the FS provider to request the current status of an initiated file copy to be reported.

Verification Method: T

8.3.3.37.4 Effect on Receipt

SAVOIR.MMS.FMS.3740

GET_FILE_COPY_STATUS.request Effect on Receipt

Receipt of the GET_FILE_COPY_STATUS.request primitive shall cause the FS provider to get information about the specified file copy operation.

Verification Method: T

8.3.3.37.5 Additional constraints

SAVOIR.MMS.FMS.3750

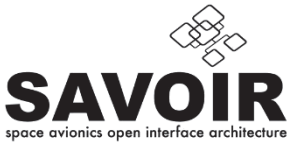
GET_FILE_COPY_STATUS.request Additional constraints: File Copy Status history

File copy statuses shall be retained for the last *<FMS maximum File Copy Status>* initiated copy operations.

Rationale: *Memory constraints prevent from indefinitely keeping file copy statuses.*

Comment: *As soon as a copy status stops being retained because of this limitation, its associated File Copy Identifier becomes invalid and its status can no longer be queried.*
<FMS maximum File Copy Status>is implementation-dependent but always greater than or equal to the maximum number of supported concurrent file copies <FMS maximum concurrent File Copy operations>.

Verification Method: T



8.3.3.38 GET_FILE_COPY_STATUS.indication

8.3.3.38.1 Function

SAVOIR.MMS.FMS.3760

GET_FILE_COPY_STATUS.indication Function

The GET_FILE_COPY_STATUS.indication primitive shall be used to pass the status of the specified file copy to the User Entity.

Verification Method: T

8.3.3.38.2 Semantics

SAVOIR.MMS.FMS.3770

GET_FILE_COPY_STATUS.indication Semantics

The GET_FILE_COPY_STATUS.indication primitive shall use the following semantics:
GET_FILE_COPY_STATUS.indication(File Transaction Identifier, File Copy Status, File Result Metadata)

Verification Method: T

8.3.3.38.3 When generated

SAVOIR.MMS.FMS.3780

GET_FILE_COPY_STATUS.indication When generated

The GET_FILE_COPY_STATUS.indication primitive shall be passed by the FS provider to the receiving User Entity in response to a GET_FILE_COPY_STATUS.request with File Result Metadata indicating if the request was executed successfully or not.

Verification Method: T

8.3.3.38.4 Effect on Receipt

The response of the User Entity to a GET_FILE_COPY_STATUS.indication is unspecified.

8.3.3.38.5 Additional constraints

SAVOIR.MMS.FMS.3790

GET_FILE_COPY_STATUS.indication Additional constraints: Invalid identifier

The GET_FILE_COPY_STATUS.indication Result Metadata shall report a failure if the requested Copy Identifier is invalid (i.e. does not refer to an already initiated copy operation).

Verification Method: T

8.3.3.39 FILE_COPY_EVENT.indication

8.3.3.39.1 Function

SAVOIR.MMS.FMS.3800

FILE_COPY_EVENT.indication Function

The FILE_COPY_EVENT.indication primitive shall be used to inform the User Entity about completion or abort of an initiated file copy.

Verification Method: T

8.3.3.39.2 Semantics

SAVOIR.MMS.FMS.3810

FILE_COPY_EVENT.indication Semantics

The FILE_COPY_EVENT.indication primitive shall use the following semantics:
FILE_COPY_EVENT.indication(File Transaction Identifier, File Copy Status)

Comment: The File Transaction Identifier shall correspond to the one provided by the User Entity when requesting the file copy (COPY_FILE.request primitive).

Verification Method: T

8.3.3.39.3 When generated

SAVOIR.MMS.FMS.3820

FILE_COPY_EVENT.indication When generated

The FILE_COPY_EVENT.indication primitive shall be passed by the FMS provider to the receiving User Entity at completion or abort of the file copy operation previously initiated with the COPY_FILE.request primitive.

Comment: The primitive provides a parameter that contains the status information of the file copy operation, but no particular Metadata as it is issued by the FMS itself.

Verification Method: T

8.3.3.39.4 Effect on Receipt

The response of the User Entity to a FILE_COPY_EVENT.indication is unspecified.

8.3.3.39.5 Additional constraints

None.

8.3.3.40 MOVE_FILE.request

8.3.3.40.1 Function

SAVOIR.MMS.FMS.3830

MOVE_FILE.request Function

The MOVE_FILE.request primitive shall be passed to the FS provider to request moving an existing file from one specified directory to another specified one, either within the same File Store, between local File Stores, or between local and remote File Stores on-board.

Comment: This requirement covers file move in both "directions" (e.g. it is possible to request a file to be moved to or from a remote File Store). The file may be moved within the same directory, which consists in renaming it.

Verification Method: T

8.3.3.40.2 Semantics

SAVOIR.MMS.FMS.3840

MOVE_FILE.request Semantics

The MOVE_FILE.request primitive shall use the following semantics:
MOVE_FILE.request(File Transaction Identifier, Source File Full Path, Destination File Full Path)

Verification Method: T

8.3.3.40.3 When generated

SAVOIR.MMS.FMS.3850

MOVE_FILE.request When generated

The MOVE_FILE.request primitive shall be passed to the FS provider to request the move (and possibly the attribution of a new name) of the specified existing source file from its current directory to a destination one.

Verification Method: T

Parent: SAVOIR-DSS-ORG-670

8.3.3.40.4 Effect on Receipt

SAVOIR.MMS.FMS.3860

MOVE_FILE.request Effect on Receipt: Move file

Receipt of the MOVE_FILE.request primitive shall cause the FS provider to move the specified source file to the specified destination directory and set its name with the Destination File Name parameter.

Verification Method: T

SAVOIR.MMS.FMS.3870

MOVE_FILE.request Effect on Receipt: File metadata

The data associated to a file by the FS (attributes, lock information, maximum file size) shall not be affected by the operation (e.g. if an Exclusive_Access lock is applied on the file, it shall still be locked once moved).

Comment: Depending on the implementation and concerned File Stores, the move operation may be achieved by only modifying FS internal data (e.g. attaching the file to a new parent directory within the same File Store), or copying the file and then deleting the source one (e.g. when distinct storage devices are involved, or between File Stores). In case the copy is involved, the requirements defined for COPY_FILE.request and COPY_FILE.indication also apply (the requesting User Entity is then the FS and indications shall be handled internally by the FS, not returned to the User Entity initiator for the MOVE_FILE.request).

Verification Method: T

8.3.3.40.5 Additional constraints

SAVOIR.MMS.FMS.3880

MOVE_FILE.request Additional constraints: Source file does not exist

The Source File Full Path parameter shall refer to an existing file.

Verification Method: T

SAVOIR.MMS.FMS.3890

MOVE_FILE.request Additional constraints: Move or rename

The Source File Full Path and Destination File Full Path shall not be identical.

Comment: However the name of source and destination files may be identical, as long as they have different parent directories (file name unity within a directory).

Verification Method: T

SAVOIR.MMS.FMS.3900

MOVE_FILE.request Additional constraints: Destination file already exists

The Destination File Full Path parameter shall not refer to an already existing file.

Comment: No overwrite allowed for a move operation.

Verification Method: T

SAVOIR.MMS.FMS.3910

MOVE_FILE.request Additional constraints: Destination path does not exist

The Destination File Full Path shall only contain existing directories.

Comment: The move procedure shall not have to create any directory.

Verification Method: T

SAVOIR.MMS.FMS.3920

MOVE_FILE.request Additional constraints: Source disappears

Once moved, the file shall only exist in the destination directory.

Comment: The file no longer exists in its source directory.

Verification Method: T

Parent: SAVOIR-DSS-ORG-660

SAVOIR.MMS.FMS.3930

MOVE_FILE.request Additional constraints: Source closed

The file to be moved shall not be opened (whatever the access type) by any User Entity.

Verification Method: T

SAVOIR.MMS.FMS.3940

MOVE_FILE.request Additional constraints: Source unused

The file to be moved shall not be involved in any on-going operation (move, copy, attribute modification, deletion, etc.) to start the moving.

Comment: These operations do not necessarily require the file to be opened.

Verification Method: T

SAVOIR.MMS.FMS.3950

MOVE_FILE.request Additional constraints: Source locking

No operation shall be allowed on the file during the moving operation (i.e. once move has been initiated).

Comment: No operation allowed means that any operation requested on the file shall be rejected (opening for reading, listing attributes, etc). This applies to all user entities.

Verification Method: T

8.3.3.41 MOVE_FILE.indication

8.3.3.41.1 Function

SAVOIR.MMS.FMS.3960

MOVE_FILE.indication Function

The MOVE_FILE.indication primitive shall be used to pass the outcome of a file move to the User Entity.

Comment: This primitive provides the result of the file move operation.

Verification Method: T

8.3.3.41.2 Semantics

SAVOIR.MMS.FMS.3970

MOVE_FILE.indication Semantics

The MOVE_FILE.indication primitive shall use the following semantics:
MOVE_FILE.indication(File Transaction Identifier, File Result Metadata)

Verification Method: T

8.3.3.41.3 When generated

SAVOIR.MMS.FMS.3980

MOVE_FILE.indication When generated

The MOVE_FILE.indication primitive shall be passed by the FS provider to the receiving User Entity in response to a MOVE_FILE.request with File Result Metadata indicating if the request was executed successfully or not.

Verification Method: T

SAVOIR.MMS.FMS.3990

MOVE_FILE.indication When generated failure

When MOVE_FILE.request is unsuccessful, the File Result Metadata shall provide the reason of the failure.

Verification Method: T

8.3.3.41.4 Effect on Receipt

The response of the User Entity to a MOVE_FILE.indication is unspecified.

8.3.3.41.5 Additional constraints

SAVOIR.MMS.FMS.4000

MOVE_FILE.indication Additional constraints: Source does not exist

The MOVE_FILE.indication File Result Metadata shall report a failure if the requested Source File Full Path refers to an inexistent file

Verification Method: T

SAVOIR.MMS.FMS.4010

MOVE_FILE.indication Additional constraints: Source locked for writing

The MOVE_FILE.indication File Result Metadata shall report a failure if the requested file identified by Source File Full Path cannot be written (lock).

Rationale: Moving a file can be considered as modifying it and write access is then required.

Verification Method: T

SAVOIR.MMS.FMS.4020

MOVE_FILE.indication Additional constraints: Source and Destination identical

The MOVE_FILE.indication File Result Metadata shall report a failure if the Source File Full Path and Destination File Full Path are identical.

Verification Method: T

SAVOIR.MMS.FMS.4030

MOVE_FILE.indication Additional constraints: Destination already exists

The MOVE_FILE.indication File Result Metadata shall report a failure if the requested Destination File Full Path refers to an already existing file.

Verification Method: T

SAVOIR.MMS.FMS.4040

MOVE_FILE.indication Additional constraints: Destination Path invalid

The MOVE_FILE.indication File Result Metadata shall report a failure if the Destination File Full Path contains one directory that does not exist.

Verification Method: T

SAVOIR.MMS.FMS.4050

MOVE_FILE.indication Additional constraints: Source opened

The MOVE_FILE.indication File Result Metadata shall report a failure if the file identified by Source File Full Path is opened by any User Entity.

Verification Method: T

SAVOIR.MMS.FMS.4060

MOVE_FILE.indication Additional constraints: Source used

The MOVE_FILE.indication File Result Metadata shall report a failure if the file identified by Source File Full Path is involved in any on-going operation (move, copy, attribute modification, deletion, etc).

Comment: The FMS can ensure this by locking the file before starting the move operation.

Verification Method: T

8.3.3.42 REGISTER_FMS_EVENT.request

8.3.3.42.1 Function

SAVOIR.MMS.FMS.4070

REGISTER_FMS_EVENT.request Function

The REGISTER_FMS_EVENT.request primitive shall be passed to the FS provider to request monitoring and further reporting of specified events occurring within the specified directory, and optionally its sub-directories.

Comment: Monitored events are reported via FMS_EVENT.indication primitive.

Verification Method: T

8.3.3.42.2 Semantics

SAVOIR.MMS.FMS.4080

REGISTER_FMS_EVENT.request Semantics

The REGISTER_FMS_EVENT.request primitive shall use the following semantics: REGISTER_FMS_EVENT.request(File Transaction Identifier, Directory Full Path, Monitored Events, Recursive)

Verification Method: T

8.3.3.42.3 When generated

SAVOIR.MMS.FMS.4090

REGISTER_FMS_EVENT.request When generated

The REGISTER_FMS_EVENT.request primitive shall be passed to the FS provider to request the specified events to be reported when occurring within the specified directory, and optionally all its sub-directories.

Comment: Event registration is managed per User Entity.

Verification Method: T

8.3.3.42.4 Effect on Receipt

SAVOIR.MMS.FMS.4100

REGISTER_FMS_EVENT.request Effect on Receipt

Receipt of the REGISTER_FMS_EVENT.request primitive shall cause the FMS provider to associate (register) the specified event(s) with the specified directory, and start monitoring of these events.

Verification Method: T

SAVOIR.MMS.FMS.4110

REGISTER_FMS_EVENT.request Effect on Receipt: Subdirectories

The Recursive parameter extends the monitoring to all the sub-directories of the specified directory.

Comment: The recursive parameter covers all sub-directories, including those which may be further created (i.e. after events registration) anywhere in the tree under the specified directory.

Verification Method: T

8.3.3.42.5 Additional constraints

SAVOIR.MMS.FMS.4120

REGISTER_FMS_EVENT.request Additional constraints Already registered

The registration to an event for a User Entity shall be rejected if the event is already registered with the specified directory.

Rationale: To avoid multiple event generation.

Verification Method: T

SAVOIR.MMS.FMS.4130

REGISTER_FMS_EVENT.request Additional constraints Parent registered

When Recursive parameter has been set, the registration to an event for a User Entity shall be rejected if the event is already registered with one of its parent directory up to the FMS root.

Rationale: To avoid multiple event generation.

Verification Method: T

SAVOIR.MMS.FMS.4140

REGISTER_FMS_EVENT.request Additional constraints Child registered

When Recursive parameter has been set, the registration to an event for a User Entity shall be rejected if the event is already registered with one of its sub- directory.

Rationale: To avoid multiple event generation.

Verification Method: T

SAVOIR.MMS.FMS.4150

REGISTER_FMS_EVENT.request Additional constraints Event combination

It shall be possible to register the events independently or grouped.

Verification Method: T

SAVOIR.MMS.FMS.4160

REGISTER_FMS_EVENT.request Additional constraints Recursivity

When the Recursive parameter is set, it applies to all the provided Monitored Events.

Comment: A User Entity can first register for file creation event within a directory with no recursive monitoring, and later request for file deletion to be monitored. However, registering again for file creation is prevented, and to get a recursive monitoring of this event, it shall be unregistered first, and registered again with the Recursive parameter selected.

Verification Method: T

SAVOIR.MMS.FMS.4170

REGISTER_FMS_EVENT.request Additional constraints: All or none

If the registration of one of the events in a group is not possible, then none of the events event of the group shall be registered.

Comment: Either all events of the group are registered, or none.

Verification Method: T

SAVOIR.MMS.FMS.4180

REGISTER_FMS_EVENT.request Additional constraints: Stop monitoring

Deleting a directory shall cause all the events registered with it to be lost and their monitoring to stop.

Comment: If the directory is re-created, the events shall be registered again.

Verification Method: T

SAVOIR.MMS.FMS.4190

REGISTER_FMS_EVENT.request Additional constraints: Renaming directory

All the registered events on a directory shall be kept and their monitoring continue when the directory is renamed.

Comment: Directory renaming may impact the User Entity which registered for the events, as the file paths of previously reported events are no longer valid.

Verification Method: T

8.3.3.43 REGISTER_FMS_EVENT.indication

8.3.3.43.1 Function

SAVOIR.MMS.FMS.4200

REGISTER_FMS_EVENT.indication Function

The REGISTER_FMS_EVENT.indication primitive shall be used to pass the outcome of registering a single or group of events with a directory to the User Entity.

Verification Method: T

8.3.3.43.2 Semantics

SAVOIR.MMS.FMS.4210

REGISTER_FMS_EVENT.indication Semantics

The REGISTER_FMS_EVENT.indication primitive shall use the following semantics: REGISTER_FMS_EVENT.indication(File Transaction Identifier, File Result Metadata)

Verification Method: T

8.3.3.43.3 When generated

SAVOIR.MMS.FMS.4220

REGISTER_FMS_EVENT.indication When generated

The REGISTER_FMS_EVENT.indication primitive shall be passed by the FS provider to the receiving User Entity in response to a REGISTER_FMS_EVENT.request with File Result Metadata indicating if the request was executed successfully or not.

Comment: The primitive provides File Result Metadata indicating if the request was executed successfully or not, and the reason in case of failure.

Verification Method: T

8.3.3.43.4 Effect on Receipt

The response of the User Entity to a REGISTER_FMS_EVENT.indication is unspecified.

8.3.3.43.5 Additional constraints

SAVOIR.MMS.FMS.4230

REGISTER_FMS_EVENT.indication Additional constraints: Directory does not exist

The REGISTER_FMS_EVENT.indication Result Metadata shall report a failure if the specified Directory Full Path refers to an inexistent directory.

Verification Method: T

SAVOIR.MMS.FMS.4240

REGISTER_FMS_EVENT.indication Additional constraints: Invalid event

The REGISTER_FMS_EVENT.indication Result Metadata shall report a failure if the specified event (or any event of the group) does not correspond to supported ones.

Comment: Only the FILE_CREATION, FILE_AUTOCLOSE, FILE_DELETION and combinations of those events are supported.

Verification Method: T

SAVOIR.MMS.FMS.4250

REGISTER_FMS_EVENT.indication Additional constraints: Registration failed

The REGISTER_FMS_EVENT.indicationMetadata shall report a failure if the single event or group of events cannot be registered with the specified directory.

Verification Method: T

8.3.3.44 UNREGISTER_FMS_EVENT.request

8.3.3.44.1 Function

SAVOIR.MMS.FMS.4260

UNREGISTER_FMS_EVENT.request Function

The UNREGISTER_FMS_EVENT.request primitive shall be passed to the FS provider to request stopping the monitoring and reporting of the specified event(s) within the specified directory, and optionally its sub-directories (based on parameters provided at registration).

Verification Method: T

8.3.3.44.2 Semantics

SAVOIR.MMS.FMS.4270

UNREGISTER_FMS_EVENT.request Semantics

The UNREGISTER_FMS_EVENT.request primitive shall use the following semantics: UNREGISTER_FMS_EVENT.request(File Transaction Identifier, Directory Full Path, Monitored Events)

Comment: In case the directory was renamed since event(s)registration, the new name shall be provided as parameter to the primitive to unregister these events.

Verification Method: T

8.3.3.44.3 When generated

SAVOIR.MMS.FMS.4280

UNREGISTER_FMS_EVENT.request When generated

The UNREGISTER_FMS_EVENT.request primitive shall be passed to the FS provider to request the specified events to no longer be reported when occurring within the specified directory, and optionally all its sub-directories.

Verification Method: T

8.3.3.44.4 Effect on Receipt

SAVOIR.MMS.FMS.4290

UNREGISTER_FMS_EVENT.request Effect on Receipt

Receipt of the UNREGISTER_FMS_EVENT.request primitive shall cause the FS provider to dissociate (unregister) the specified events from the specified directory, and stop monitoring these events (including in subdirectories if recursive monitoring was selected at registration for an event).

Verification Method: T

8.3.3.44.5 Additional constraints

SAVOIR.MMS.FMS.4300

UNREGISTER_FMS_EVENT.request Additional constraints: Event registered

It shall only be possible to unregister an event that is currently registered with the Directory Full Path.

Comment: If FILE_CREATION and FILE_DELETION events were registered with a directory, only those two events can be further unregistered, and only once (until new registration).
Events can be unregistered only at the "level" they were registered (i.e. it is not possible to unregister an event from a sub-directory when recursive monitoring is applied on its parent)

Verification Method: T

SAVOIR.MMS.FMS.4310

UNREGISTER_FMS_EVENT.request Additional constraints: Same directory

It shall only be possible to unregister an event from the directory it was registered with (with its new name in case renaming occurred).

Comment: Events can be unregistered only at the "level" they were registered (i.e. it is not possible to unregister an event from a sub-directory when recursive monitoring is applied on its parent).

Verification Method: T

SAVOIR.MMS.FMS.4320

UNREGISTER_FMS_EVENT.request Additional constraints Individual or grouped unregistration

It shall be possible to unregister the events independently or grouped.

Verification Method: T

SAVOIR.MMS.FMS.4330

UNREGISTER_FMS_EVENT.request Additional constraints Unregistration only by registered User Entity

Only the User Entity which registered an event shall be able to unregister it.

Verification Method: T

SAVOIR.MMS.FMS.4340

UNREGISTER_FMS_EVENT.request Additional constraints All or none

If the unregistration of one of the events in a group is not possible, there shall be no event of this group unregistered.

Comment: Either all events of the group are unregistered, or none.

Verification Method: T

8.3.3.45 UNREGISTER_FMS_EVENT.indication

8.3.3.45.1 Function

SAVOIR.MMS.FMS.4350

UNREGISTER_FMS_EVENT.indication Function

The UNREGISTER_FMS_EVENT.indication primitive shall be used to pass the outcome of unregistering a single or group of events from a directory to the User Entity.

Verification Method: T

8.3.3.45.2 Semantics

SAVOIR.MMS.FMS.4360

UNREGISTER_FMS_EVENT.indication Semantics

The UNREGISTER_FMS_EVENT.indication primitive shall use the following semantics:
UNREGISTER_FMS_EVENT.indication(File Transaction Identifier, File Result Metadata)

Verification Method: T

8.3.3.45.3 When generated

SAVOIR.MMS.FMS.4370

UNREGISTER_FMS_EVENT.indication When generated

The UNREGISTER_FMS_EVENT.indication primitive shall be passed by the FS provider to the receiving User Entity in response to anUNREGISTER_FMS_EVENT.request with File Result Metadata indicating if the operation was successful or not.

Verification Method: T

8.3.3.45.4 Effect on Receipt

The response of the User Entity to an UNREGISTER_FMS_EVENT.indication is unspecified.

8.3.3.45.5 Additional constraints

SAVOIR.MMS.FMS.4380

UNREGISTER_FMS_EVENT.indication Additional constraints Directory does not exist

The UNREGISTER_FMS_EVENT.indication File Result Metadata shall report a failure if the specified Directory Full Path refers to an inexistent directory.

Verification Method: T

SAVOIR.MMS.FMS.4390

UNREGISTER_FMS_EVENT.indication Additional constraints Invalid event

The UNREGISTER_FMS_EVENT.indication Result Metadata shall report a failure if the specified event (or any event of the group) does not correspond to supported ones.

Comment: Only the FILE_CREATION, FILE_AUTOCLOSE, FILE_DELETION and combinations of those events are supported, and thus can be unregistered from a file.

Verification Method: T

SAVOIR.MMS.FMS.4400

UNREGISTER_FMS_EVENT.indication Additional constraints Unregistration failed

The UNREGISTER_FMS_EVENT.indication Result Metadata shall report a failure if the single event or group of events cannot be unregistered from the specified directory.

Verification Method: T

8.3.3.46 FMS_EVENT.indication

8.3.3.46.1 Function

SAVOIR.MMS.FMS.4410

FMS_EVENT.indication Function

The FMS_EVENT.indication primitive shall be used to report the occurrence of a registered event to the User Entity.

Verification Method: T

8.3.3.46.2 Semantics

SAVOIR.MMS.FMS.4420

FMS_EVENT.indication Semantics

The FMS_EVENT.indication primitive shall use the following semantics:
FMS_EVENT.indication(File Transaction Identifier, FMS Event)

Comment: The Transaction Identifier shall correspond to the one provided by the User Entity when requesting the event registration (REGISTER_FMS_EVENT.request primitive).

Verification Method: T

8.3.3.46.3 When generated

SAVOIR.MMS.FMS.4430

FMS_EVENT.indication When generated

The FMS_EVENT.indication primitive shall be passed by the FMS provider to the receiving User Entity at each occurrence of a monitored (i.e. registered) event.

Comment: The primitive provides a parameter that contains the event characteristics.

Verification Method: T

8.3.3.46.4 Effect on Receipt

The response of the User Entity to a FMS_EVENT.indication is unspecified.

8.3.3.46.5 Additional constraints

None.

8.3.3.47 CREATE_DIR.request

8.3.3.47.1 Function

SAVOIR.MMS.FMS.4440

CREATE_DIR.request Function

The CREATE_DIR.request primitive shall be passed to the FS provider to request the creation of the specified directory in a File Store.

Verification Method: T

8.3.3.47.2 Semantics

SAVOIR.MMS.FMS.4450

CREATE_DIR.request Semantics

The CREATE_DIR.request primitive shall use the following semantics:
CREATE_DIR.request(File Transaction Identifier, Directory Full Path)

Verification Method: T

8.3.3.47.3 When generated

SAVOIR.MMS.FMS.4460

CREATE_DIR.request When generated

The CREATE_DIR.request primitive shall be passed to the FS provider to request the creation of a directory.

Verification Method: T

8.3.3.47.4 Effect on Receipt

SAVOIR.MMS.FMS.4470

CREATE_DIR.request Effect on Receipt

Receipt of the CREATE_DIR.request primitive shall cause the FS provider to create the specified directory.

Verification Method: T

8.3.3.47.5 Additional constraints

SAVOIR.MMS.FMS.4480

CREATE_DIR.request Additional constraints Already existing

The creation shall be rejected if the Directory Full Path parameter refers to an existing directory or file.

Rationale: *Name uniqueness within a directory applies to both files and directories (i.e. a file and a directory cannot have the same name despite their different type).*

Verification Method: T

SAVOIR.MMS.FMS.4490

CREATE_DIR.request Additional constraints Invalid path

The creation shall be rejected if the Directory Full Path contains one to several inexistent directories, excluding the one to be created.

Rationale: *All directories that are part of a Directory Full Path have to be created one after the other.*

Verification Method: T

SAVOIR.MMS.FMS.4500

CREATE_DIR.request Additional constraints Maximum depth reached

The creation shall be rejected if it causes the maximum directory depth (i.e. number of sub-directories from the File Store root) to be exceeded.

Rationale: *Depending on the implementation, each File Store may have its own directory depth configuration.*

Comment: *Note that flat File Systems (i.e. authorized directory depth is 0) are covered by this requirement.*

Verification Method: T

8.3.3.48 CREATE_DIR.indication

8.3.3.48.1 Function

SAVOIR.MMS.FMS.4510

CREATE_DIR.indication Function

The CREATE_DIR.indication primitive shall be used to pass the outcome of creating a directory to the User Entity.

Verification Method: T

8.3.3.48.2 Semantics

SAVOIR.MMS.FMS.4520

CREATE_DIR.indication Semantics

The CREATE_DIR.indication primitive shall use the following semantics:
CREATE_DIR.indication(File Transaction Identifier, Result Metadata)

Verification Method: T

8.3.3.48.3 When generated

SAVOIR.MMS.FMS.4530

CREATE_DIR.indication When generated

The CREATE_DIR.indication primitive shall be passed by the FS provider to the receiving User Entity in response to a CREATE_DIR.request.

Comment: The primitive provides Result Metadata indicating if the request was executed successfully or not, and the reason in case of failure.

Verification Method: T

8.3.3.48.4 Effect on Receipt

The response of the User Entity to a CREATE_DIR.indication is unspecified.

8.3.3.48.5 Additional constraints

SAVOIR.MMS.FMS.4540

CREATE_DIR.indication Additional constraints: Already existing

The CREATE_DIR.indication Result Metadata shall report a failure in case the Directory Full Path parameter refers to an existing directory or file.

Verification Method: T

SAVOIR.MMS.FMS.4550

CREATE_DIR.indication Additional constraints: Invalid path

The CREATE_DIR.indication Result Metadata shall report a failure in case the Directory Full Path contains inexistent directories (except the one to be created).

Verification Method: T

SAVOIR.MMS.FMS.4560

CREATE_DIR.indication Additional constraints: Maximum depth reached

The CREATE_DIR.indication Result Metadata shall report a failure in case the maximum directory depth applicable to the File Store would be exceeded by creating the directory.

Verification Method: T

8.3.3.49 LIST_DIR.request

8.3.3.49.1 Function

SAVOIR.MMS.FMS.4570

LIST_DIR.request Function

The LIST_DIR.request primitive shall be passed to the FMS provider to request listing the content of the specified directory.

Verification Method: T

8.3.3.49.2 Semantics

SAVOIR.MMS.FMS.4580

LIST_DIR.request Semantics

The LIST_DIR.request primitive shall use the following semantics:
LIST_DIR.request(File Transaction Identifier, Directory Full Path)

Verification Method: T

8.3.3.49.3 When generated

SAVOIR.MMS.FMS.4590

LIST_DIR.request When generated

The LIST_DIR.request primitive shall be passed to the FMS provider to request the content of a directory to be listed.

Verification Method: T

8.3.3.49.4 Effect on Receipt

SAVOIR.MMS.FMS.4600

LIST_DIR.request Effect on Receipt

Receipt of the LIST_DIR.request primitive shall cause the FMS provider to list all the files and directories contained within the specified directory.

Comment: There is no recursive listing of subdirectories (to get their content, subsequent calls to the LIST_DIR.request primitive with appropriate Directory Full Path are required).

Verification Method: T

8.3.3.49.5 Additional constraints

SAVOIR.MMS.FMS.4610

LIST_DIR.request Additional constraints: Directory is existing

The requested Directory Full Path shall refer to an existing directory.

Verification Method: T

8.3.3.50 LIST_DIR.indication

8.3.3.50.1 Function

SAVOIR.MMS.FMS.4620

LIST_DIR.indication Function

The LIST_DIR.indication primitive shall be used to pass the directory listing to the User Entity.

Verification Method: T

8.3.3.50.2 Semantics

SAVOIR.MMS.FMS.4630

LIST_DIR.indication Semantics

The LIST_DIR.indication primitive shall use the following semantics:
LIST_DIR.indication(File Transaction Identifier, Directory Listing, File Result Metadata)

Verification Method: T

8.3.3.50.3 When generated

SAVOIR.MMS.FMS.4640

LIST_DIR.indication When generated

The LIST_DIR.indication primitive shall be passed by the FMS provider to the receiving User Entity in response to a LIST_DIR.request.

Comment: The primitive provides the list of files and directories contained within the specified directory and File Result Metadata indicating if the request was executed successfully or not, and the reason in case of failure.

Verification Method: T

8.3.3.50.4 Effect on Receipt

The response of the User Entity to a LIST_DIR.indication is unspecified.

8.3.3.50.5 Additional constraints

SAVOIR.MMS.FMS.4650

LIST_DIR.indication Additional constraints: Directory does not exist

The LIST_DIR.indication File Result Metadata shall report a failure in case the requested Directory Full Path refers to an inexistent directory.

Verification Method: T

8.3.3.51 DELETE_DIR.request

8.3.3.51.1 Function

SAVOIR.MMS.FMS.4660

DELETE_DIR.request Function

The DELETE_DIR.request primitive shall be passed to the FMS provider to request the deletion of an existing and empty directory in a File Store.

Verification Method: T

8.3.3.51.2 Semantics

SAVOIR.MMS.FMS.4670

DELETE_DIR.request Semantics

The DELETE_DIR.request primitive shall use the following semantics:
DELETE_DIR.request(File Transaction Identifier, Directory Full Path)

Verification Method: T

8.3.3.51.3 When generated

SAVOIR.MMS.FMS.4680

DELETE_DIR.request When generated

The DELETE_DIR.request primitive shall be passed to the FMS provider to request the deletion of the specified empty directory.

Verification Method: T

8.3.3.51.4 Effect on Receipt

SAVOIR.MMS.FMS.4690

DELETE_DIR.request Effect on Receipt

Receipt of the DELETE_DIR.request primitive shall cause the FMS provider to delete the specified directory.

Verification Method: T

8.3.3.51.5 Additional constraints

SAVOIR.MMS.FMS.4700

DELETE_DIR.request Additional constraints: Directory not empty

The operation shall be rejected if the requested Directory Full Path parameter refers to a non-empty directory.

Comment: A directory is considered empty when it contains no file, and no subdirectory (may it be empty or not).

Verification Method: T

8.3.3.52 DELETE_DIR.indication

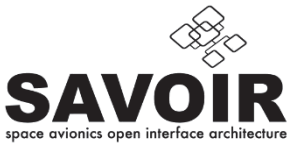
8.3.3.52.1 Function

SAVOIR.MMS.FMS.4710

DELETE_DIR.indication Function

The DELETE_DIR.indication primitive shall be used to pass the outcome of deleting a directory to the User Entity.

Verification Method: T



8.3.3.52.2 Semantics

SAVOIR.MMS.FMS.4720

DELETE_DIR.indication Semantics

The DELETE_DIR.indication primitive shall use the following semantics:
DELETE_DIR.indication(File Transaction Identifier, File Result Metadata)

Verification Method: T

8.3.3.52.3 When generated

SAVOIR.MMS.FMS.4730

DELETE_DIR.indication When generated

The DELETE_DIR.indication primitive shall be passed by the FMS provider to the receiving User Entity in response to a DELETE_DIR.request.

Comment: The primitive provides File Result Metadata indicating if the request was executed successfully or not, and the reason in case of failure.

Verification Method: T

8.3.3.52.4 Effect on Receipt

The response of the User Entity to a DELETE_DIR.indication is unspecified.

8.3.3.52.5 Additional constraints

SAVOIR.MMS.FMS.4740

DELETE_DIR.indication Additional constraints: Directory does not exist

The DELETE_DIR.indication File Result Metadata shall report a failure if the requested Directory Full Path refers to an inexistent directory.

Verification Method: T

SAVOIR.MMS.FMS.4750

DELETE_DIR.indication Additional constraints: Directory is not empty

The DELETE_DIR.indication Result Metadata shall report a failure if the requested Directory Full Path refers to a directory that is not empty.

Verification Method: T

8.3.3.53 RENAME_DIR.request

8.3.3.53.1 Function

SAVOIR.MMS.FMS.4760

RENAME_DIR.request Function

The RENAME_DIR.request primitive shall be passed to the FMS provider to request the renaming of a specified directory in a File Store.

Verification Method: T

8.3.3.53.2 Semantics

SAVOIR.MMS.FMS.4770

RENAME_DIR.request Semantics

The RENAME_DIR.request primitive shall use the following semantics:
RENAME_DIR.request(File Transaction Identifier, Old Directory Full Path, New Directory Name)

Verification Method: T

8.3.3.53.3 When generated

SAVOIR.MMS.FMS.4780

RENAME_DIR.request When generated

The RENAME_DIR.request primitive shall be passed to the FMS provider to request the renaming of the specified directory.

Verification Method: T

8.3.3.53.4 Effect on Receipt

SAVOIR.MMS.FMS.4790

RENAME_DIR.request Effect on Receipt

Receipt of the RENAME_DIR.request primitive shall cause the FMS provider to rename the specified directory if none of the files it contains (recursively including subdirectories) is opened.

Verification Method: T

8.3.3.53.5 Additional constraints

SAVOIR.MMS.FMS.4800

RENAME_DIR.request Additional constraints: Directory exists

The Old Directory Full Path parameter shall refer to an existing directory.

Verification Method: T

SAVOIR.MMS.FMS.4810

RENAME_DIR.request Additional constraints: Rename only

The New Directory Name parameter shall contain the new name of the directory.

Rationale: *The purpose of this primitive is only to support directory renaming, not moving operation, i.e. New Directory Name parameter is not a Directory Full Path.*

Verification Method: T

SAVOIR.MMS.FMS.4820

RENAME_DIR.request Additional constraints: Entry does not exist

The New Directory Name parameter shall not refer to an already existing directory or file.

Rationale: *Each File and Directory name is unique within a same Directory.*

Verification Method: T

SAVOIR.MMS.FMS.4830

RENAME_DIR.request Additional constraints: Keep registered events

Events registered with the directory shall remain associated and monitored after directory renaming.

Rationale: *To not affect on-going monitoring on the Directory to be renamed.*

Comment: *Refer to REGISTER_FMS_EVENT.request and UNREGISTER_FMS_EVENT.request primitives for events (un)registration. The generated events will contain the New Directory Name after renaming.*

Verification Method: T

8.3.3.54 RENAME_DIR.indication

8.3.3.54.1 Function

SAVOIR.MMS.FMS.4840

RENAME_DIR.indication Function

The RENAME_DIR.indication primitive shall be used to pass the outcome of renaming a directory to the User Entity.

Verification Method: T

8.3.3.54.2 Semantics

SAVOIR.MMS.FMS.4850

RENAME_DIR.indication Semantics

The RENAME_DIR.indication primitive shall use the following semantics:
RENAME_DIR.indication(File Transaction Identifier, File Result Metadata)

Verification Method: T

8.3.3.54.3 When generated

SAVOIR.MMS.FMS.4860

RENAME_DIR.indication When generated

The RENAME_DIR.indication primitive shall be passed by the FMS provider to the receiving User Entity in response to a RENAME_DIR.request.

Comment: The primitive provides File Result Metadata indicating if the request was executed successfully or not, and the reason in case of failure.

Verification Method: T

8.3.3.54.4 Effect on Receipt

The response of the User Entity to a RENAME_DIR.indication is unspecified.

8.3.3.54.5 Additional constraints

SAVOIR.MMS.FMS.4870

RENAME_DIR.indication Additional constraints: Directory does not exist

The RENAME_DIR.indication File Result Metadata shall report a failure if the requested Old Directory Full Path refers to an inexistent directory.

Verification Method: T

SAVOIR.MMS.FMS.4880

RENAME_DIR.indication Additional constraints: Directory already exists

The RENAME_DIR.indication Result Metadata shall report a failure if the requested New Directory Name already exists within the parent directory (i.e. in the directory containing the directory to be renamed).

Verification Method: T

SAVOIR.MMS.FMS.4890

RENAME_DIR.indication Additional constraints: File opened

The RENAME_DIR.indication Result Metadata shall report a failure if the Old Directory Full Path contains a file that is current opened by any User Entity.

Verification Method: T

8.3.3.55 LOCK_FILE.request

8.3.3.55.1 Function

SAVOIR.MMS.FMS.4900

LOCK_FILE.request Function

The LOCK_FILE.request primitive shall be passed to the FMS provider to request the locking of an existing.

Verification Method: T

8.3.3.55.2 Semantics

SAVOIR.MMS.FMS.4910

LOCK_FILE.request Semantics

The LOCK_FILE.request primitive shall use the following semantics:
LOCK_FILE.request(File Transaction Identifier, File Full Path, Lock Type)

Verification Method: T

8.3.3.55.3 When generated

SAVOIR.MMS.FMS.4920

LOCK_FILE.request When generated

The LOCK_FILE.request primitive shall be passed to the FMS provider to request the specified file to be locked (i.e. application of restrictions depending on User Entity for file opening and deletion).

Verification Method: T

8.3.3.55.4 Effect on Receipt

SAVOIR.MMS.FMS.4930

LOCK_FILE.request Effect on Receipt

Receipt of the LOCK_FILE.request primitive shall cause the FMS provider to lock the specified file with the specified lock type.

Verification Method: T

8.3.3.55.5 Additional constraints

SAVOIR.MMS.FMS.4940

LOCK_FILE.request Additional constraints: File exists

The File Full Path parameter shall refer to an existing file.

Verification Method: T

SAVOIR.MMS.FMS.4950

LOCK_FILE.request Additional constraints: File not opened by others

The requested locking shall be rejected if the file is currently open by at least one User Entity other than the one calling the lock.

Rationale: It is not possible to apply restrictions on a file being already accessed.

Verification Method: T

SAVOIR.MMS.FMS.4960

LOCK_FILE.request Additional constraints: File not locked

The requested locking shall be rejected if the file is already locked (whatever the lock owner).

Verification Method: T

8.3.3.56 LOCK_FILE.indication

8.3.3.56.1 Function

SAVOIR.MMS.FMS.4970

LOCK_FILE.indication Function

The LOCK_FILE.indication primitive shall be used to pass the outcome of locking a file to the User Entity.

Verification Method: T

8.3.3.56.2 Semantics

SAVOIR.MMS.FMS.4980

LOCK_FILE.indication Semantics

The LOCK_FILE.indication primitive shall use the following semantics:
LOCK_FILE.indication(File Transaction Identifier, Lock Identifier, File Result Metadata)

Verification Method: T

8.3.3.56.3 When generated

SAVOIR.MMS.FMS.4990

LOCK_FILE.indication When generated

The LOCK_FILE.indication primitive shall be passed by the FMS provider to the receiving User Entity in response to a LOCK_FILE.request.

Comment: The primitive provides the identifier of the lock applied on the file, and File Result Metadata indicating if the request was executed successfully or not, and the reason in case of failure.

Verification Method: T

8.3.3.56.4 Effect on Receipt

The response of the User Entity to a LOCK_FILE.indication is unspecified.

8.3.3.56.5 Additional constraints

SAVOIR.MMS.FMS.5000

LOCK_FILE.indication Additional constraints: File exists

The LOCK_FILE.indication File Result Metadata shall report a failure if the requested File Full Path refers to an inexistent file.

Verification Method: T

SAVOIR.MMS.FMS.5010

LOCK_FILE.indication Additional constraints: File opened by other

The LOCK_FILE.indication File Result Metadata shall report a failure if the requested file is currently open by at least one User Entity.

Verification Method: T

SAVOIR.MMS.FMS.5020

LOCK_FILE.indication Additional constraints: File already locked

The LOCK_FILE.indication File Result Metadata shall report a failure if the requested file is already locked.

Verification Method: T

8.3.3.57 UNLOCK_FILE.request

8.3.3.57.1 Function

SAVOIR.MMS.FMS.5030

UNLOCK_FILE.request Function

The UNLOCK_FILE.request primitive shall be passed to the FMS provider to request the unlocking of a locked file in a File Store.

Verification Method: T

8.3.3.57.2 Semantics

SAVOIR.MMS.FMS.5040

UNLOCK_FILE.request Semantics

The UNLOCK_FILE.request primitive shall use the following semantics:
UNLOCK_FILE.request(File Transaction Identifier, Lock Identifier)

Verification Method: T

8.3.3.57.3 When generated

SAVOIR.MMS.FMS.5050

UNLOCK_FILE.request When generated

The UNLOCK_FILE.request primitive shall be passed to the FMS provider to request the specified file to be unlocked.

Comment: Unlocking removes any restriction related to file opening and deletion.

Verification Method: T

8.3.3.57.4 Effect on Receipt

SAVOIR.MMS.FMS.5060

UNLOCK_FILE.request Effect on Receipt

Receipt of the UNLOCK_FILE.request primitive shall cause the FMS provider to remove the lock (i.e. unlock) currently applied on the specified file.

Verification Method: T

8.3.3.57.5 Additional constraints

SAVOIR.MMS.FMS.5070

UNLOCK_FILE.request Additional constraints: Unlock for all

Any User Entity shall be able to unlock a locked file.

Rationale: *To be able to unlock files locked by User entity that failed.*

Comment: *No restrictions are placed on which user entities may unlock a file, e.g. other that the lock-owner.*

Verification Method: T

8.3.3.58 UNLOCK_FILE.indication

8.3.3.58.1 Function

SAVOIR.MMS.FMS.5080

UNLOCK_FILE.indication Function

The UNLOCK_FILE.indication primitive shall be used to pass the outcome of unlocking a file to the User Entity.

Verification Method: T

8.3.3.58.2 Semantics

SAVOIR.MMS.FMS.5090

UNLOCK_FILE.indication Semantics

The UNLOCK_FILE.indication primitive shall use the following semantics:
UNLOCK_FILE.indication(File Transaction Identifier, File Result Metadata)

Verification Method: T

8.3.3.58.3 When generated

SAVOIR.MMS.FMS.5100

UNLOCK_FILE.indication When generated

The UNLOCK_FILE.indication primitive shall be passed by the FMS provider to the receiving User Entity in response to an UNLOCK_FILE.request.

Comment: *The primitive provides File Result Metadata indicating if the request was executed successfully or not, and the reason in case of failure.*

Verification Method: T

8.3.3.58.4 Effect on Receipt

The response of the User Entity to an UNLOCK_FILE.indication is unspecified.

8.3.3.58.5 Additional constraints

SAVOIR.MMS.FMS.5110

UNLOCK_FILE.indication Additional constraints: Invalid lock

The UNLOCK_FILE.indication File Result Metadata shall report a failure if the requested Lock Identifier does not refer to a valid lock (i.e. a lock currently applied to a file).

Verification Method: T

SAVOIR.MMS.FMS.5111

UNLOCK_FILE.indication File locked by another User Entity

The UNLOCK_FILE.indication File Result Metadata shall report a failure if the file is locked by another User Entity in any mode.

Rationale: *Unlocking a File locked by another User Entity is not possible.*

Comment: *If found too restrictive, it is possible to implement a User Entity having all the rights on the FMS or being able to impersonate the User Entity that locked the File.*

Verification Method: T

8.3.3.59 LIST_LOCKED_FILES.request

8.3.3.59.1 Function

SAVOIR.MMS.FMS.5120

LIST_LOCKED_FILES.request Function

The LIST_LOCKED_FILES.request primitive shall be passed to the FMS provider to request the listing of all locked files within all the File Stores managed/accessed by the FMS.

Comment: *On-board remote File Stores included.*

Verification Method: T

8.3.3.59.2 Semantics

SAVOIR.MMS.FMS.5130

LIST_LOCKED_FILES.request Semantics

The LIST_LOCKED_FILES.request primitive shall use the following semantics:
LIST_LOCKED_FILES.request(File Transaction Identifier)

Verification Method: T

8.3.3.59.3 When generated

SAVOIR.MMS.FMS.5140

LIST_LOCKED_FILES.request When generated

The LIST_LOCKED_FILES.request primitive shall be passed to the FMS provider to request all the locked files to be listed.

Verification Method: T

8.3.3.59.4 Effect on Receipt

SAVOIR.MMS.FMS.5150

LIST_LOCKED_FILES.request Effect on Receipt

Receipt of the LIST_LOCKED_FILES.request primitive shall cause the FMS provider to list all the currently locked files (with their lock characteristics) within all the File Stores it manages/accesses.

Verification Method: T

8.3.3.59.5 Additional constraints

None.

8.3.3.60 LIST_LOCKED_FILES.indication

8.3.3.60.1 Function

SAVOIR.MMS.FMS.5160

LIST_LOCKED_FILES.indication Function

The LIST_LOCKED_FILES.indication primitive shall be used to pass the list of all locked files to the User Entity.

Verification Method: T

8.3.3.60.2 Semantics

SAVOIR.MMS.FMS.5170

LIST_LOCKED_FILES.indication Semantics

The LIST_LOCKED_FILES.indication primitive shall use the following semantics:
LIST_LOCKED_FILES.indication(File Transaction Identifier, Locked Files List, File Result Metadata)

Verification Method: T

8.3.3.60.3 When generated

SAVOIR.MMS.FMS.5180

LIST_LOCKED_FILES.indication When generated

The LIST_LOCKED_FILES.indication primitive shall be passed by the FMS provider to the receiving User Entity in response to a LIST_LOCKED_FILES.request.

Comment: The primitive provides the list of locked files, and File Result Metadata indicating if the request was executed successfully or not, and the reason in case of failure.

Verification Method: T

8.3.3.60.4 Effect on Receipt

The response of the User Entity to a LIST_LOCKED_FILES.indication is unspecified.

8.3.3.60.5 Additional constraints

SAVOIR.MMS.FMS.5190

LIST_LOCKED_FILES.indication Additional constraints: No lock files

The LIST_LOCKED_FILES.indication primitive shall return an empty list if no file is currently locked.

Comment: This is not an error case.

Verification Method: T

8.3.3.61 FIND_FILES.request

8.3.3.61.1 Function

SAVOIR.MMS.FMS.5200

FIND_FILES.request Function

The FIND_FILES.request primitive shall be passed to the FS provider to request the finding of all files matching a selection pattern against File Status (which includes the name, creation date, etc.) from a base directory which is located within any of the File Stores managed/accessed by the FMS.

Comment: The capability of the FS provider can be limited to find files only on a subset of the File Status (e.g. only names). If a required search is not supported by the FS, the FMS shall implement it using available FS Services.

Comment: The base directory can be the FMS root.

Verification Method: T

8.3.3.61.2 Semantics

SAVOIR.MMS.FMS.5210

FIND_FILES.request Semantics

The FIND_FILES.request primitive shall use the following semantics:
FIND_FILES.request(File Transaction Identifier, File Selection Pattern, Base Directory Full Path, Recursive option)

Verification Method: T

8.3.3.61.3 When generated

SAVOIR.MMS.FMS.5220

FIND_FILES.request When generated

The FIND_FILES.request primitive shall be passed to the FS provider to request all the files matching the File Selection Pattern to be found from the base directory.

Verification Method: T

8.3.3.61.4 Effect on Receipt

SAVOIR.MMS.FMS.5230

FIND_FILES.request Effect on Receipt

Receipt of the FIND_FILES.request primitive shall cause the FS provider to find all the files matching the selection pattern against file status within the area specified (Base Directory Full Path and recursive option parameter).

Comment: There may zero, one or many files matching the File Selection Pattern.

Verification Method: T

SAVOIR.MMS.FMS.5240

FIND_FILES.request Effect on Receipt: Recursive search

The FS provider shall perform the file search at least in the base directory and, if requested in FIND_FILES.request primitive parameter, search recursively in the all the subdirectories.

Verification Method: T

8.3.3.61.5 Additional constraints

SAVOIR.MMS.FMS.5250

FIND_FILES.request Additional constraints: hierarchical search

Search operation in a hierarchical File Store shall result in a search through all "branches" forming the tree-like hierarchy.

Comment: This corresponds to a recursive search operation.

Verification Method: T

8.3.3.62 FIND_FILES.indication

8.3.3.62.1 Function

SAVOIR.MMS.FMS.5260

FIND_FILES.indication Function

The FIND_FILES.indication primitive shall be used to pass the list of files matching the File Selection Pattern to the User Entity.

Verification Method: T

8.3.3.62.2 Semantics

SAVOIR.MMS.FMS.5270

FIND_FILES.indication Semantics

The FIND_FILES.indication primitive shall use the following semantics:
FIND_FILES.indication(File Transaction Identifier, Found Files List, File Result Metadata)

Verification Method: T

8.3.3.62.3 When generated

SAVOIR.MMS.FMS.5280

FIND_FILES.indication When generated

The FIND_FILES.indication primitive shall be passed by the FS provider to the receiving User Entity in response to a FIND_FILES.request.

Comment: The primitive provides the list of files matching the File Selection Pattern, and File Result Metadata indicating if the request was executed successfully or not, and the reason in case of failure.

Verification Method: T

8.3.3.62.4 Effect on Receipt

The response of the User Entity to a FIND_FILES.indication is unspecified.

8.3.3.62.5 Additional constraints

SAVOIR.MMS.FMS.5290

FIND_FILES.indication Additional constraints: File not found

The FIND_FILES.indication primitive shall return an empty list if no file matches the specified File Selection Pattern within at least the specified Base Directory Full Path and optionally recursively in all subdirectories (depending on the request's parameters).

Comment: Finding no file matching the pattern is not an error case.

Verification Method: T

8.3.3.63 SET_DIR_ATTRIBUTE.request

8.3.3.63.1 Function

SAVOIR.MMS.FMS.5300

SET_DIR_ATTRIBUTE.request Function

The SET_DIR_ATTRIBUTE.request primitive shall be passed to the FS provider to request associating an Attribute to a specified directory.

Comment: An attribute is an information complementary to the directory content.

Verification Method: T

8.3.3.63.2 Semantics

SAVOIR.MMS.FMS.5310

SET_DIR_ATTRIBUTE.request Semantics

The SET_DIR_ATTRIBUTE.request primitive shall use the following semantics: SET_DIR_ATTRIBUTE.request(File Transaction Identifier, Directory Full Path, Attribute)

Verification Method: T

8.3.3.63.3 When generated

SAVOIR.MMS.FMS.5320

SET_DIR_ATTRIBUTE.request When generated

The SET_DIR_ATTRIBUTE.request primitive shall be passed to the FS provider to request an attribute to be associated (created or updated) to the specified file.

Verification Method: T

8.3.3.63.4 Effect on Receipt

SAVOIR.MMS.FMS.5330

SET_DIR_ATTRIBUTE.request Effect on Receipt: Attribute verification

Receipt of the SET_DIR_ATTRIBUTE.request primitive shall cause the FMS provider to first check if the attribute is already associated to the specified directory.

Verification Method: T

SAVOIR.MMS.FMS.5340

SET_DIR_ATTRIBUTE.request Effect on Receipt: Attribute creation

If the Attribute is not already associated to the directory identified by its Directory Full Path, the attribute shall be created with the information provided in the Attribute parameter and associated to the directory.

Verification Method: T

SAVOIR.MMS.FMS.5350

SET_DIR_ATTRIBUTE.request Effect on Receipt: Attribute update

If an attribute with same identifier is already associated to the directory identified by its Directory Full Path, the attribute shall be updated with the information provided in the Attribute parameter.

Comment: Some restriction may apply in some implementations related the restriction of updates to Attribute with the same Attribute Type and Attribute Size.

Verification Method: T

8.3.3.63.5 Additional constraints

SAVOIR.MMS.FMS.5360

SET_DIR_ATTRIBUTE.request Additional constraints: Maximum number of attributes

The maximum number of Attributes that can be associated per directory shall be configured per File system.

Comment: This is mainly related to the used File System limitations and defined in the File System characteristics.

Verification Method: T

8.3.3.64 SET_DIR_ATTRIBUTE.indication

8.3.3.64.1 Function

SAVOIR.MMS.FMS.5370

SET_DIR_ATTRIBUTE.indication Function

The SET_DIR_ATTRIBUTE.indication primitive shall be used to pass the outcome of associating an attribute to a directory to the User Entity.

Verification Method: T

8.3.3.64.2 Semantics

SAVOIR.MMS.FMS.5380

SET_DIR_ATTRIBUTE.indication Semantics

The SET_DIR_ATTRIBUTE.indication primitive shall use the following semantics:
SET_DIR_ATTRIBUTE.indication(File Transaction Identifier, File Result Metadata)

Verification Method: T

8.3.3.64.3 When generated

SAVOIR.MMS.FMS.5390

SET_DIR_ATTRIBUTE.indication When generated

The SET_DIR_ATTRIBUTE.indication primitive shall be passed by the FS provider to the receiving User Entity in response to a SET_DIR_ATTRIBUTE.request with File Result Metadata indicating if the operation was successful or not.

Verification Method: T

SAVOIR.MMS.FMS.5400

SET_DIR_ATTRIBUTE.indication When generated failure

When SET_DIR_ATTRIBUTE.request is unsuccessful, the File Result Metadata shall provide the reason of the failure.

Verification Method: T

8.3.3.64.4 Effect on Receipt

The response of the User Entity to a SET_DIR_ATTRIBUTE.indication is unspecified.

8.3.3.64.5 Additional constraints

SAVOIR.MMS.FMS.5410

SET_DIR_ATTRIBUTE.indication Additional constraints: Directory does not exist

The SET_DIR_ATTRIBUTE.indication File Result Metadata shall report a failure if the requested Directory Full Path refers to an inexistent directory.

Verification Method: T

SAVOIR.MMS.FMS.5420

SET_DIR_ATTRIBUTE.indication Additional constraints: Too many attributes

The SET_DIR_ATTRIBUTE.indication File Result Metadata shall report a failure if the Attribute Name is not already associated to the directory and the maximum number of attributes per directory has been reached (i.e. insertion cannot be achieved).

Comment: Updating the value of an already existing/associated attribute remains possible.

Verification Method: T

SAVOIR.MMS.FMS.5430

SET_DIR_ATTRIBUTE.indication Additional constraints: Bad Type

The SET_DIR_ATTRIBUTE.indication Metadata shall report a failure if the requested Attribute Type is different from the supported types.

Verification Method: T

8.3.3.65 GET_DIR_ATTRIBUTES.request

8.3.3.65.1 Function

SAVOIR.MMS.FMS.5440

GET_DIR_ATTRIBUTES.request Function

The GET_DIR_ATTRIBUTES.request primitive shall be passed to the FMS provider to request reporting the collection of attributes currently associated to the specified directory.

Verification Method: T

8.3.3.65.2 Semantics

SAVOIR.MMS.FMS.5450

GET_DIR_ATTRIBUTES.request Semantics

The GET_DIR_ATTRIBUTES.request primitive shall use the following semantics:
GET_DIR_ATTRIBUTES.request(Transaction Identifier, Directory Full Path)

Verification Method: T

8.3.3.65.3 When generated

SAVOIR.MMS.FMS.5460

GET_DIR_ATTRIBUTES.request When generated

The GET_DIR_ATTRIBUTES.request primitive shall be passed to the FS provider to request listing the attributes (with their respective value and type) currently associated to the directory identified by Directory Full Path.

Verification Method: T

8.3.3.65.4 Effect on Receipt

SAVOIR.MMS.FMS.5470

GET_DIR_ATTRIBUTES.request Effect on Receipt

Receipt of the GET_DIR_ATTRIBUTES.request primitive shall cause the FS provider to get all the attributes currently associated to the directory identified by Directory Full Path.

Verification Method: T

8.3.3.65.5 Additional constraints

None.

8.3.3.66 GET_DIR_ATTRIBUTES.indication

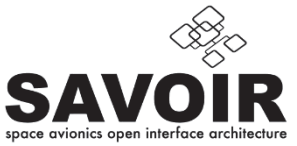
8.3.3.66.1 Function

SAVOIR.MMS.FMS.5480

GET_DIR_ATTRIBUTES.indication Function

The GET_DIR_ATTRIBUTES.indication primitive shall be used to pass the collection of attributes associated to a directory to the User Entity.

Verification Method: T



8.3.3.66.2 Semantics

SAVOIR.MMS.FMS.5490

GET_DIR_ATTRIBUTES.indication Semantics

The GET_DIR_ATTRIBUTES.indication primitive shall use the following semantics:
GET_DIR_ATTRIBUTES.indication(File Transaction Identifier, Attribute List, File Result Metadata)

Verification Method: T

8.3.3.66.3 When generated

SAVOIR.MMS.FMS.5500

GET_DIR_ATTRIBUTES.indication When generated

The GET_DIR_ATTRIBUTES.indication primitive shall be passed by the FS provider to the receiving User Entity in response to a GET_DIR_ATTRIBUTES.request with File Result Metadata indicating if the operation was successful or not.

Verification Method: T

SAVOIR.MMS.FMS.5510

GET_DIR_ATTRIBUTES.indication When generated success

When GET_DIR_ATTRIBUTES.request is successful, the Attribute List parameter shall contain the complete list of attributes associated to the directory.

Verification Method: T

SAVOIR.MMS.FMS.5520

GET_DIR_ATTRIBUTES.indication When generated failure

When GET_DIR_ATTRIBUTES.request is unsuccessful, the File Result Metadata shall provide the reason of the failure.

Verification Method: T

8.3.3.66.4 Effect on Receipt

The response of the User Entity to a GET_DIR_ATTRIBUTES.indication is unspecified.

8.3.3.67 GET_BAU_LIST.request

8.3.3.67.1 Function

SAVOIR.MMS.FMS.5530

GET_BAU_LIST.request Function

The GET_BAU_LIST.request primitive shall be passed to the FMS provider to request reporting the list of BAUs that are used to store the data contained in the file.

Verification Method: T

8.3.3.67.2 Semantics

SAVOIR.MMS.FMS.5540

GET_BAU_LIST.request Semantics

The GET_BAU_LIST.request primitive shall use the following semantics:
GET_BAU_LIST.request(File Transaction Identifier, File Full Path)

Verification Method: T

8.3.3.67.3 When generated

SAVOIR.MMS.FMS.5550

GET_BAU_LIST.request When generated

The GET_BAU_LIST.request primitive shall be passed to the FS provider to request the list of BAUs that are allocated to store the data of the file identified by File Full Path.

Verification Method: T

8.3.3.67.4 Effect on Receipt

SAVOIR.MMS.FMS.5560

GET_BAU_LIST.request Effect on Receipt

Receipt of the GET_BAU_LIST.request primitive shall cause the FS provider to retrieve the list of BAUs associated to the file identified by File Full Path.

Verification Method: T

8.3.3.67.5 Additional constraints

None.

8.3.3.68 GET_BAU_LIST.indication

8.3.3.68.1 Function

SAVOIR.MMS.FMS.5570

GET_BAU_LIST.indication Function

The GET_BAU_LIST.indication primitive shall be used to pass the list of BAUs associated to a file to the User Entity.

Verification Method: T

8.3.3.68.2 Semantics

SAVOIR.MMS.FMS.5580

GET_BAU_LIST.indication Semantics

The GET_BAU_LIST.indication primitive shall use the following semantics:
GET_BAU_LIST.indication(File Transaction Identifier, BAU List, File Result Metadata)

Verification Method: T

8.3.3.68.3 When generated

SAVOIR.MMS.FMS.5590

GET_BAU_LIST.indication When generated

The GET_BAU_LIST.indication primitive shall be passed by the FS provider to the receiving User Entity in response to a GET_BAU_LIST.request with File Result Metadata indicating if the operation was successful or not.

Verification Method: T

SAVOIR.MMS.FMS.5600

GET_BAU_LIST.indication When generated success

When GET_BAU_LIST.request is successful, the BAU List parameter shall contain the complete list of BAUs associated to the file.

Verification Method: T

SAVOIR.MMS.FMS.5610

GET_BAU_LIST.indication When generated failure

When GET_BAU_LIST.request is unsuccessful, the File Result Metadata shall provide the reason of the failure.

Verification Method: T

8.3.3.68.4 Effect on Receipt

The response of the User Entity to a GET_BAU_LIST.indication is unspecified.

8.3.3.68.5 Additional constraints

SAVOIR.MMS.FMS.5620

GET_BAU_LIST.indication Additional constraints: File does not exist

The GET_BAU_LIST.indication File Result Metadata shall report a failure if the requested File Full Path refers to an inexistent file.

Verification Method: T

8.3.3.69 FILE_DEFRAGMENTATION.request

8.3.3.69.1 Function

SAVOIR.MMS.FMS.5630

FILE_DEFRAGMENTATION.request Function

The FILE_DEFRAGMENTATION.request primitive shall be passed to the FMS provider to request the defragmentation of the Files managed by the FMS.

Verification Method: T

8.3.3.69.2 Semantics

SAVOIR.MMS.FMS.5640

FILE_DEFRAGMENTATION.request Semantics

The FILE_DEFRAGMENTATION.request primitive shall use the following semantics:
FILE_DEFRAGMENTATION.request(File Transaction Identifier, Store Identifier)

Verification Method: T

8.3.3.69.3 When generated

SAVOIR.MMS.FMS.5650

FILE_DEFRAGMENTATION.request When generated

The FILE_DEFRAGMENTATION.request primitive shall be passed to the FS provider to request all the files contained in the identified Store to be defragmented.

Verification Method: T

8.3.3.69.4 Effect on Receipt

SAVOIR.MMS.FMS.5660

FILE_DEFRAGMENTATION.request Effect on Receipt

Receipt of the FILE_DEFRAGMENTATION.request primitive shall cause the FS provider to defragment the files contained in the identified Store.

Verification Method: T

8.3.3.69.5 Additional constraints

Some constraints imposed by the File System may be defined such as the need to have no opened files in the identified Store.

8.3.3.70 FILE_DEFRAGMENTATION.indication

8.3.3.70.1 Function

SAVOIR.MMS.FMS.5670

FILE_DEFRAGMENTATION.indication Function

The FILE_DEFRAGMENTATION.indication primitive shall be used to pass the outcome of defragmenting files of a Store.

Verification Method: T

8.3.3.70.2 Semantics

SAVOIR.MMS.FMS.5680

FILE_DEFRAGMENTATION.indication Semantics

The FILE_DEFRAGMENTATION.indication primitive shall use the following semantics:
FILE_DEFRAGMENTATION.indication(File Transaction Identifier, File Result Metadata)

Verification Method: T

8.3.3.70.3 When generated

SAVOIR.MMS.FMS.5690

FILE_DEFRAGMENTATION.indication When generated

The FILE_DEFRAGMENTATION.indication primitive shall be passed by the FS provider to the receiving User Entity in response to a FILE_DEFRAGMENTATION.request with File Result Metadata indicating if the operation was successful or not.

Verification Method: T

SAVOIR.MMS.FMS.5700

FILE_DEFRAGMENTATION.indication When generated failure

When FILE_DEFRAGMENTATION.request is unsuccessful, the File Result Metadata shall provide the reason of the failure.

Verification Method: T

8.3.3.70.4 Effect on Receipt

The response of the User Entity to a FILE_DEFRAGMENTATION.indication is unspecified.

8.3.3.70.5 Additional constraints

SAVOIR.MMS.FMS.5710

FILE_DEFRAGMENTATION.indication Additional constraints: Store does not exist

The FILE_DEFRAGMENTATION.indication File Result Metadata shall report a failure if the identified Store does not exist.

Verification Method: T

SAVOIR.MMS.FMS.5720

FILE_DEFRAGMENTATION.indication Additional constraints: Store not managed

The FILE_DEFRAGMENTATION.indication File Result Metadata shall report a failure if the identified Store is not managed by a File System or does not supporte defragmentation operation.

Verification Method: T

8.3.3.71 MAP_FILE_SPLIT.request

8.3.3.71.1 Function

SAVOIR.MMS.FMS.5730

MAP_FILE_SPLIT.request Function

The MAP_FILE_SPLIT.request primitive shall be passed to the FS provider to request to force the next incoming data to be stored in a new file.

Comment: The service is only valid if Continuous Storage has been selected during the creation of the mapping.

Verification Method: T

8.3.3.71.2 Semantics

SAVOIR.MMS.FMS.5740

MAP_FILE_SPLIT.request Semantics

The MAP_FILE_SPLIT.request primitive shall use the following semantics:
MAP_FILE_SPLIT.request(File Transaction Identifier, File Descriptor)

Verification Method: T

8.3.3.71.3 When generated

SAVOIR.MMS.FMS.5750

MAP_FILE_SPLIT.request When generated

The MAP_FILE_SPLIT.request primitive shall be passed to the FS provider to request the current file to be closed and next file to be opened and used to store data received on MM inputs that match the specified Mapping Criteria.

Verification Method: T

8.3.3.71.4 Effect on Receipt

SAVOIR.MMS.FMS.5760

MAP_FILE_SPLIT.request Effect on Receipt: close and open next file

Receipt of the MAP_FILE_SPLIT.request primitive shall cause the FMS provider to close the file currently used to store the data received from the mapped interface and open the next file that will be used to store next data.

Verification Method: T

8.3.3.71.5 Additional constraints

SAVOIR.MMS.FMS.5770

MAP_FILE_SPLIT.request Additional constraints: Continuous Storage

The switch to the next file shall be performed without any data loss.

Comment: *The way the split is performed is mission dependent. It can be immediate (i.e. at byte level) or at a level depending on the Direct data acquisition type and File Mapping configuration.*

Parent: SAVOIR-DSS-ORG-550

Verification Method: T

SAVOIR.MMS.FMS.5780

MAP_FILE_SPLIT.request Additional constraints: Non Continuous Storage

If the Continuous Storage parameter in the Mapping Criteria is not set, the request shall be rejected.

Rationale: *If the storage is not continuous, it is not possible to segment the data flow.*

Parent: SAVOIR-DSS-ORG-420

Verification Method: T

SAVOIR.MMS.FMS.5790

MAP_FILE_SPLIT.request Additional constraints: Delegation

The FMS operations (closing file and opening file) shall be performed by the FMS provider on behalf of the user's entity at the origin of the initial MAP_FILE_SPLIT.request.

Comment: *The User Entity request the split may be different from the one that initiated the mapping.*

Verification Method: T

8.3.3.72 MAP_FILE_SPLIT.indication

8.3.3.72.1 Function

SAVOIR.MMS.FMS.5800

MAP_FILE_SPLIT.indication Function

The MAP_FILE_SPLIT.indication primitive shall be used to indicate to the User Entity the outcome of the mapping segmentation.

Verification Method: T

8.3.3.72.2 Semantics

SAVOIR.MMS.FMS.5810

MAP_FILE_SPLIT.indication Semantics

The MAP_FILE_SPLIT.indication primitive shall use the following semantics:
MAP_FILE_SPLIT.indication(Transaction Identifier, File Result Metadata)

Verification Method: T

8.3.3.72.3 When generated

SAVOIR.MMS.FMS.5820

MAP_FILE_SPLIT.indication When generated

The MAP_FILE_SPLIT.indication primitive shall be passed by the FS provider to the receiving User Entity in response to an MAP_FILE_SPLIT.request with File Result Metadata indicating if the request was executed successfully or not.

Verification Method: T

SAVOIR.MMS.FMS.5830

MAP_FILE_SPLIT.indication When generated success

When MAP_FILE_SPLIT.request is successful, the File Descriptor is updated to identify then newly opened file.

Verification Method: T

SAVOIR.MMS.FMS.5840

MAP_FILE_SPLIT.indication When generated failure

When MAP_FILE_SPLIT.request is unsuccessful, the File Result Metadata shall provide the reason of the failure.

Verification Method: T

8.3.3.72.4 Effect on Receipt

The response of the User Entity to a MAP_FILE_SPLIT.indication is unspecified.

8.3.3.72.5 Additional constraints

SAVOIR.MMS.FMS.5850

MAP_FILE_SPLIT.indication Additional constraints: File Descriptor invalid

The MAP_FILE_SPLIT.indication File Result Metadata shall report a failure if the requested File Descriptor does not exist.

Verification Method: T

SAVOIR.MMS.FMS.5860

MAP_FILE_SPLIT.indication Additional constraints: Non continuous mapping

The MAP_FILE_SPLIT.indication File Result Metadata shall report a failure if the File Descriptor does not refer to a continuous file mapping:

Verification Method: T

8.3.4 Concurrency management

This section is aiming at describing the FMS behaviour when a single file is concurrently opened by several user entities at a time. A use-case will deeply details the expected behavior depending on file opening criteria (Access Type, Full File Action, Packet Storage).

8.3.4.1 General behavior synthesis (inherited from requirements)

As a basic introduction, here are some concepts to know before analyzing the use-case:

- When a User Entity opens a file, it gets a File Descriptor to access its content. A unique File Descriptor is returned by FMS for each file opening, which means that different File Descriptors are returned when opening twice the same file, whatever the User Entity.
- Each File Descriptor retains a single position for read and write operations in the file (called File Descriptor's current position in the above requirements). Depending on Access Type selected at opening, the position can be moved with the FILE_SEEK.request, otherwise it is internally managed by the FMS.
- When a single file is opened by several User Entities with "Read" or "Read-Write" Access Type, the position in the file is independently managed per File Descriptor (i.e. depending on respective operations performed). This means that overwriting may happen between User Entities.
- When a single file is opened by several User Entities with "Append" Access Type, the position in the file is synchronized between the File Descriptors (i.e. when a User Entity appends data through its File Descriptor, the position of other File Descriptors is automatically advanced to the end-of-file). Append Access Type guarantees no overwriting in case of concurrent opening of the file.
- Selecting the "Packet Storage" option when opening the file prevents a Data Segment to be partially written, or spread over the current and next file (in case the current file gets full). If the Data Segment cannot be integrally appended to the file (insufficient remaining space), it is either discarded or written to the next file, depending on the "Full File Action" option.
- When the File Descriptor's current position reaches the configured Maximum File Size, no more data can be appended to the file. However content overwriting is still possible prior the end-of-file, with "Read+Write" Access type.
- To support creating/opening the next file when "Full File Action" option is set to "Close & Create Next File", the FMS shall define an implementation-dependent naming convention, so that the name of the next file is clearly identified.
- When the "Full File Action" option is set to "Close & Create Next File", the next file is opened if it already exists, otherwise it is created and then opened.
- Once having closed a file (through its File Descriptor), the file content can no longer be accessed through this File Descriptor. Other File Descriptors referring to this file are not affected, and thus the file remains open for their respective User Entities.

8.3.4.2 Explanation by use-case

The initial conditions for the use-case are as follow:

- File "F1" exists in a File Store managed by the FMS. Its Maximum File Size is set to 100 bytes. 85 bytes have already been written to the file, so its EOF is at position 85.

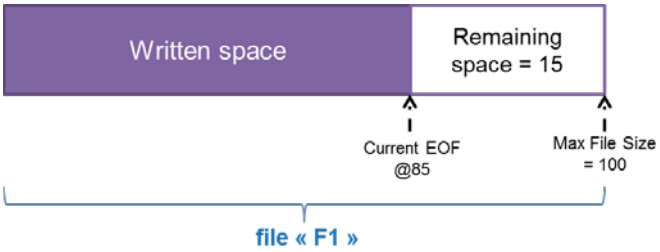


Figure 5: Initial file characteristics

We will consider 3 distinct User Entities (UE1, UE2, UE3), each one opening the "F1" file:

- UE1 opens it with the following "File Opening Criteria":
  - Access Type = "Append",
  - Full File Action = "Close & Create Next File",
  - Packet Storage option selected.
- UE2 opens it with the same criteria as UE1 (cf. above).
- UE3 opens it with the following "File Opening Criteria":
  - Access Type = "Read+Write",
  - Full File Action and Packet Storage are not applicable.

This can be represented as follow, with the 3 File Descriptors and their respective position (R+W pointer):

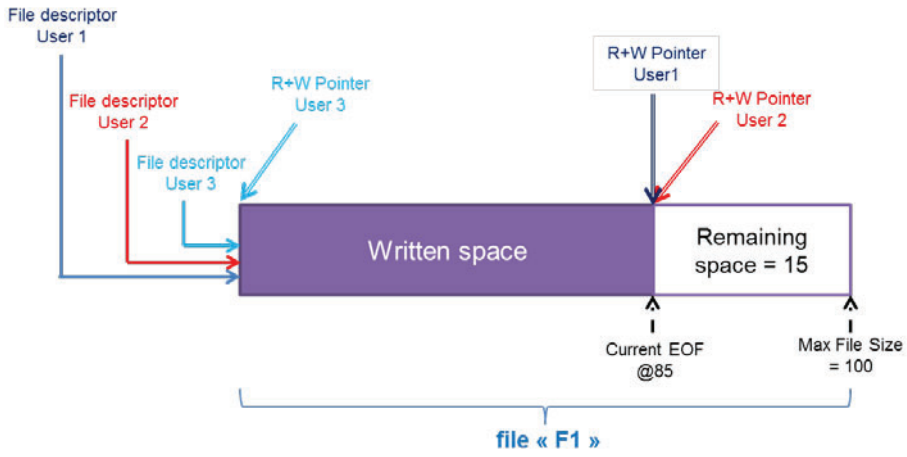


Figure 6: File opening by the 3 User Entities + respective positions

Based on the selected Access Type, the position is initialized to the end-of-file for the UE1 and UE2 File Descriptors. For UE3 the position is initialized at the beginning of the file (i.e. position 0).

Now let's consider that UE1 requests writing a 20 bytes size Data Segment, which is larger than the remaining space in "F1":

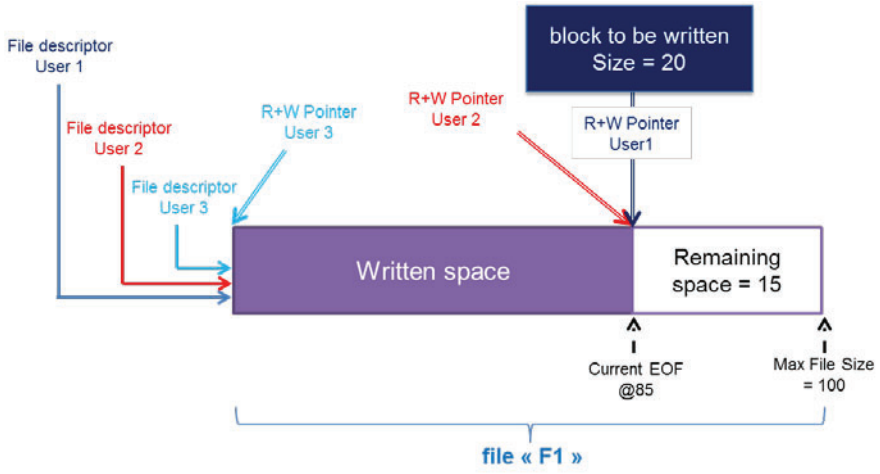


Figure 7: A User Entity requests writing a data segment bigger than the remaining space

As UE1 opened the file with the "Full File Action" set to "Close & Create Next File" and the "Packet Storage" option selected, the Data Segment cannot be split, and thus must be stored into the next file. As file "F2" does not already exist, it is automatically created and opened by the FMS.

Before effectively creating the next file, the Maximum File Size of "F1" is set (by the FMS) to the current end-of-file to prevent more data to be appended. From that point, the remaining space is 0:

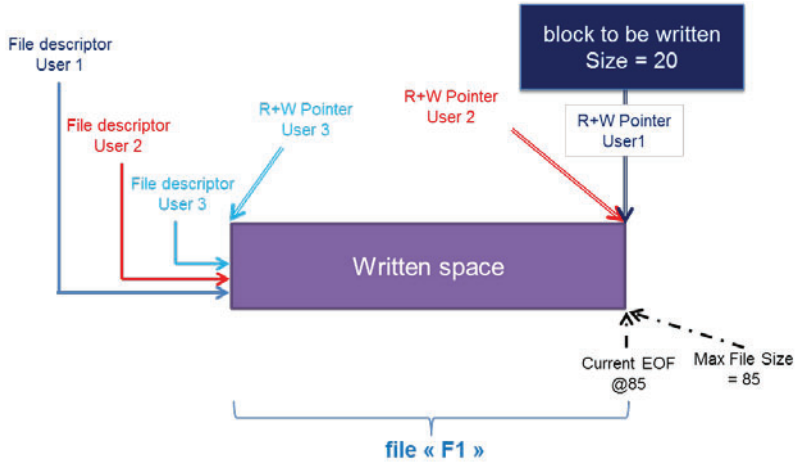


Figure 8: Modification of the Maximum File Size by the FMS to prevent data to be appended

Due to the Full File Action, the FMS closes “F1” for UE1, creates and opens “F2”, stores within it the 20 bytes Data Segment, and returns to UE1 the new File Descriptor for further access to the content of “F2”. The UE2 and UE3 File Descriptors are not affected by the closing, and their positions are not modified:

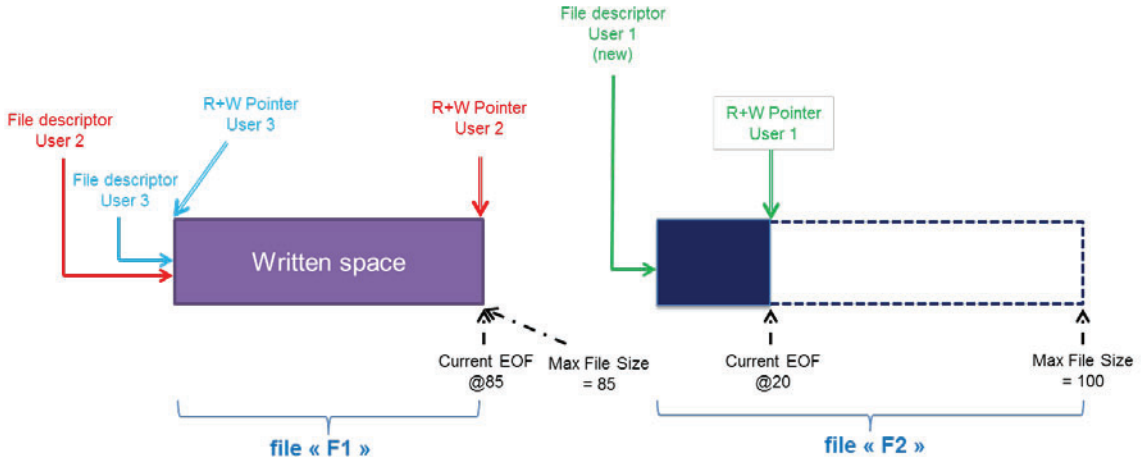


Figure 9: Writing of data segment to the next file (including creation)

From this moment, UE2 can no longer write any data to “F1”, as it opened the file with “Append” Access Type, and its position has reached the Maximum File Size (due to the modification applied by the creation of the next file).

However, UE3 is still able to move its position to overwrite/patch the content of the file prior the end-of-file:

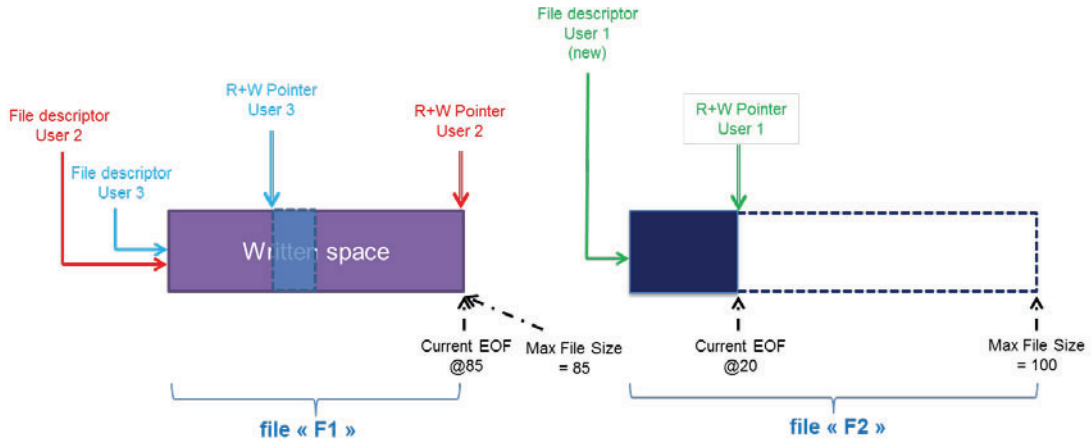


Figure 10: Writing to F1 by User Entity 3, whereas it is impossible for User Entity 2

Now let’s consider that UE2 requests for appending a 10 bytes size Data Segment. This can obviously not be achieved in “F1” as it is full (i.e. its File Descriptor position corresponds to the Maximum File Size).

Note that the position of UE3 File Descriptor was automatically moved by the write operation (it is now located at the end of the overwritten part):

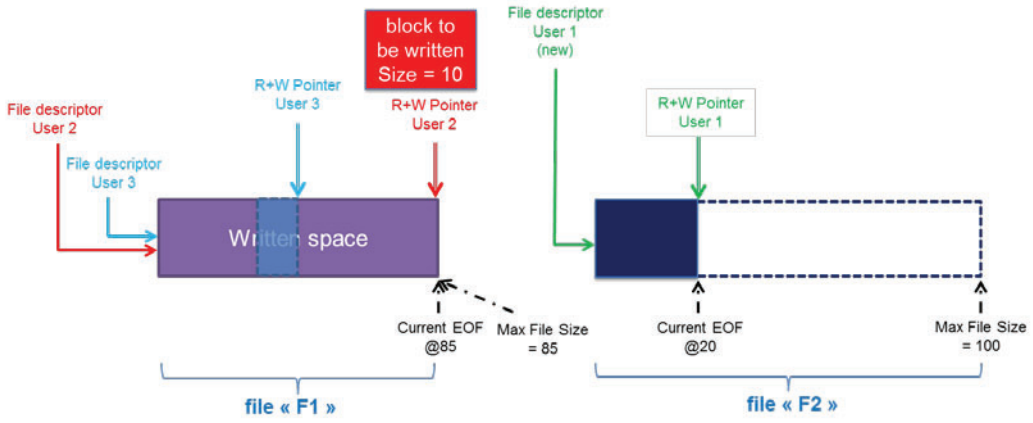


Figure 11: A User Entity attempts to append data to a full file

Based on File Opening Criteria, and as the next file already exists, the FMS simply opens it. Opening of the next file (i.e. "F2") is performed with the same criteria as "F1", and thus with "Append" Access type. For this reason, UE2 position is initialized to the current end-of-file, i.e. at the same location as UE1 (position 20).

As there is no contraindication, the Data Segment is appended to "F2", and positions of UE2 File Descriptors is moved to the new end-of-file (position 30). The position of other File Descriptors for file opened in Append Mode is automatically moved to the end-of-file.

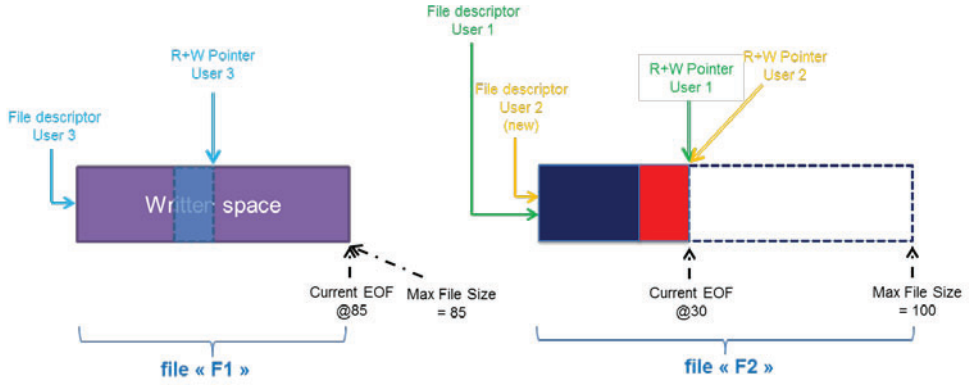


Figure 12: Data is finally appended to the next file, and positions are synchronized

Thanks to these mechanisms, the FMS guarantees that no overwriting occurs between User Entities having opened the file with "Append" Access Type, while still allowing independent modifications of content with "Read+Write" Access Type.

Note that the next file "F2" inherits the initial Maximum File Size of "F1" (i.e. 100 bytes). The Maximum File Size can only be defined at creation by the User Entity, but it can be modified internally by the FMS at any time, with the constraint to never truncate file

content (i.e. delimited by the current end-of-file which corresponds to the highest written position in the file, and is common to all File Descriptors).

Now let's consider that another User Entity (called UE4) opens "F1" with the following File Opening Criteria:

- Access Type = "Append",
- Full File Action = "Close & Create Next File",
- Packet Storage option NOT selected.

Based on these criteria, the position of its File Descriptor is initialized at the end of file (i.e. position 85), and no more data can be directly appended to this file due to the Maximum File Size.

Now, UE4 requests for appending a 120 bytes size Data Segment:

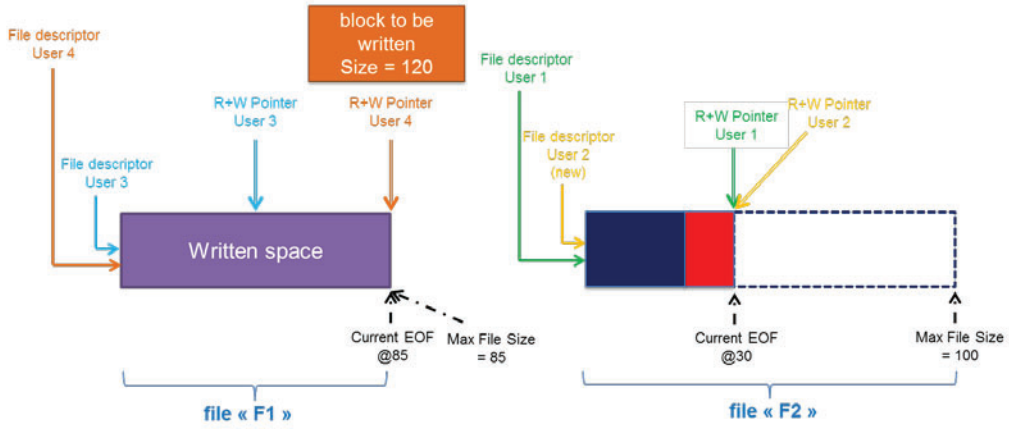


Figure 13: A User Entity opens the first file of the series and requests appending a new data segment

As "F1" is full, no more data can be appended to it. Because of the Full File Action, the next file "F2" is internally opened by the FMS and the position initialized at the current end-of-file (position 30):

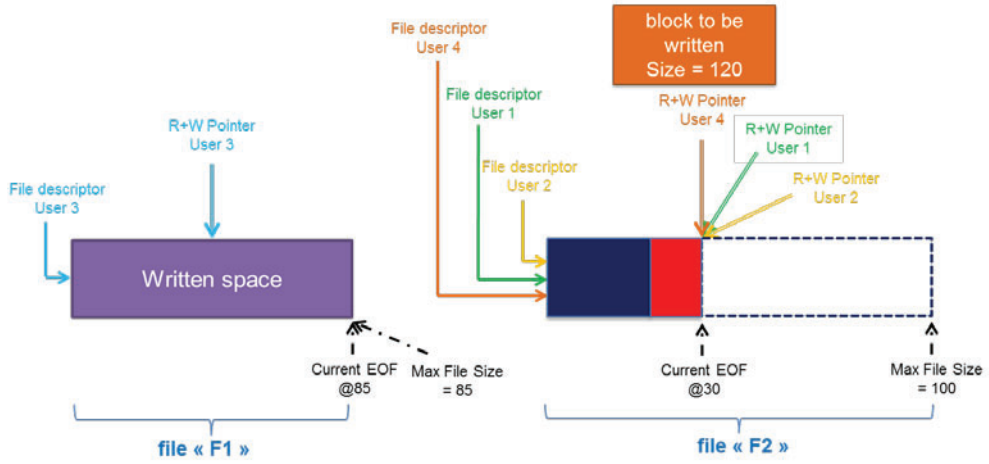


Figure 14: First intermediate step - opening of next file by the FMS

Note that at this time no File Descriptor has been returned to the User Entity yet (it is only handled internally by the FMS).

The first 70 bytes of the Data Segment are appended to “F2” (filling the remaining space), and the positions of UE1 and UE2 File Descriptors are automatically moved to the end-of-file (synchronization related to “Append” Access type):

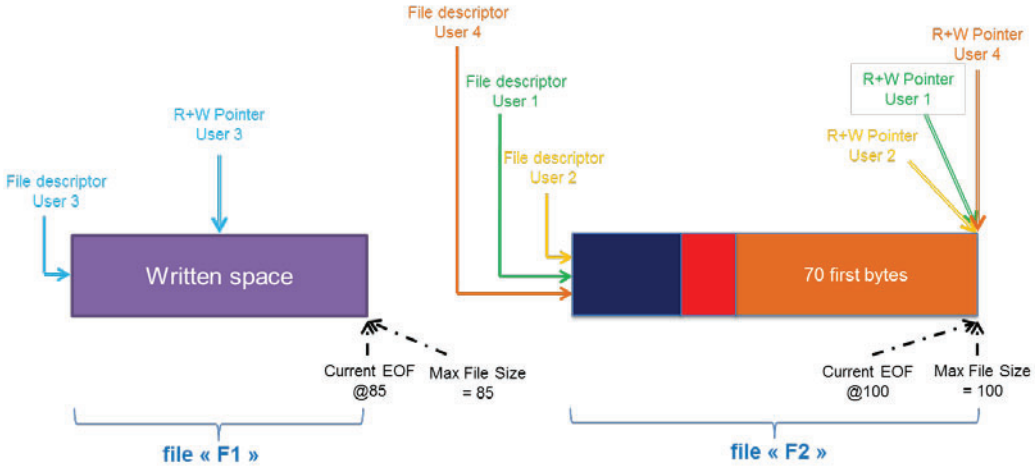


Figure 15: Partial appending of Data Segment (as Packet Storage option was not selected)

But the last 50 bytes of the Data Segment have not been stored yet. As the File Opening Criteria of “F2” are identical to those selected by UE4 at opening of “F1”, the Full File Action requires the FMS to automatically create/open the next file when the current one gets full. As the next file (“F3”) does not already exist, it is created by the FMS to store the last part of Data Segment:

At the end of this set of operations, the FMS changesthe“F3” File Descriptor handled by UE 4 for further access. If the Packet Storage option had been selected, the operation would

have failed as the Data Segment is larger than the Maximum File Size (120 > 100), and such an operation has to be rejected.

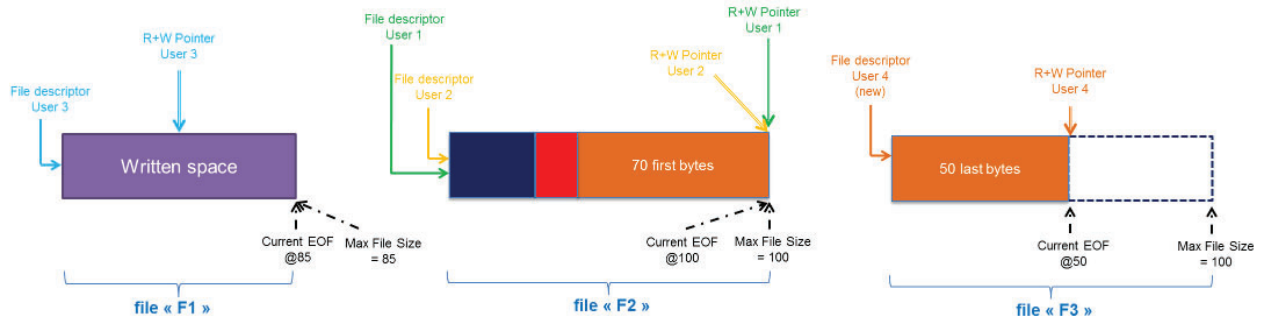


Figure 16: Creation of next File to store last part of Data Segment

Note that if "F1"had been opened by UE4 with "Append" Access Type and Full File Action set to "Close File", the file would have been immediately closed by the FMS at first writing attempt, and no data write at all. Moreover UE4 would not have been able to read anything from the file (File Descriptor position was initialized to end of file at opening).

As a general remark, it is advised not to open a file with Full File Action set to "Close File" when concurrent opening with "Append" Access Type is foreseen for it.

8.4 Performances

This section lists the performance requirements related to the Mass Memory System.

SAVOIR.MMS.PER.0100

MMS uncorrectable bit error rate

The MMS shall ensure an uncorrectable bit error rate (UBER) of less than <MMS uncorrectable bit error rate> per cycle.

Rationale: *UBER has to be computed and maintained below a threshold granting overall reliability.*

Comment: *The cycle is defined at mission definition stage. It can be a type based value (e.g. year, day, hour…), an operation based value (amount of data processed…) or else.*

Comment: *Accepted error rates may be dependent on individual missions. Figures shall be provided for all the functions.*

Comment: *To achieve the targeted rate, the MMS can implement specific mechanisms to increase the reliability of the Data Storage System, e.g. dedicated memory error detection and correction functions or by supporting duplication of data in different Storage areas (similar to RAID 1).*

Verification Method: T

Parent: SAVOIR-DSS-PER-010

SAVOIR.MMS.PER.0110

MMS failure rate

The MMS shall be designed to provide a reliability of no more than <MMS failure rate> FIT over the total operational lifetime including all tests and storage times on ground.

Comment: *The MMS can implement specific mechanisms to increase the reliability of the Data Storage System, e.g. by supporting duplication of data in different Storage areas and/or the use of error correction mechanisms (similar to RAID 5).*

Verification Method: T

Parent: SAVOIR-DSS-PER-020

SAVOIR.MMS.PER.0120

MMS data average input throughput

The MM shall support an average data throughput <MMS average input rate> independently per physical input.

Rationale: *The data throughput includes margin and is specific to each physical input.*

Comment: *The MMS can implement specific mechanisms to increase the performance of the Data Storage System, e.g. by supporting use of independent Storage areas (similar to RAID 0).*

Verification Method: T

Parent: SAVOIR-DSS-PER-030

SAVOIR.MMS.PER.0130

MMS data burst input throughput

The MM shall support a burst rate of <MMS burst input rate> with a maximum duration of <MM burst duration> independently per physical input.

Rationale: *The data throughput is not necessarily constant and is specific to each physical input.*

Comment: *The MMS can implement specific mechanisms to increase the performance of the Data Storage System, e.g. by supporting use of independent Storage areas (similar to RAID 0).*

Verification Method: T

Parent: SAVOIR-DSS-PER-030

SAVOIR.MMS.PER.0140

MMS data overall volume

The MMS shall support writing at least <MMS data volume> data volume from hardware unit manufacturing to launch + <Mission time 2> including all test procedures.

Rationale: *The MMS has to consider the total amount of data to be stored (written in memory) during the all life of the system.*

Comment: *The MMS has to consider limitation of some memory technologies (like Flash devices).*

Verification Method: T

Parent: SAVOIR-DSS-PER-040

SAVOIR.MMS.PER.0150

MMS maximum output throughput

The MMS shall support a maximum data output rate <MMS maximum output rate> independently per physical output.

Rationale: *The MMS has to output the data at the maximum rates requested by the receivers.*

Comment: *This includes the traffic to the OBC (reading operation) and downlink to Ground (potentially via several board/Ground links).*

Verification Method: T

Parent: SAVOIR-DSS-PER-050

SAVOIR.MMS.PER.0160

MMS maximum command rate

The MMS shall support the reception of up to <MMS maximum command rate> commands (from Ground or on-board application) per second.

Rationale: *The MMS receives commands for file management (FMS interface), data transfer (FTS), reconfiguration, etc.*

Comment: *The reception of command is performed at the same time as data (and Housekeeping telemetry in some cases) is being received or sent on same or other physical link.*

Comment: *Commands have to be executed such that a stall of command processing is prevented (provided that the specified maximum MMS command rate is not exceeded).*

Verification Method: T

Parent: SAVOIR-DSS-PER-060

SAVOIR.MMS.PER.0170

MMS housekeeping telemetry

The MMS shall support the generation of its housekeeping telemetry without impacting the other operations (command processing, data storage).

Comment: *Housekeeping emission is performed at the same time as the data is received or sent and commands are received.*

Verification Method: T

Parent: SAVOIR-DSS-PER-070

SAVOIR.MMS.PER.0180

MMS simultaneous accesses

The MMS shall be able to handle up to <MMS simultaneous memory read or write> operations without degradation of performances.

Rationale: *The MMS has to record all the data coming from all sources and extract data for transmission to all destinations concurrently (OBC, FTS) while complying with the performances required by the mission.*

Verification Method: T

Parent: SAVOIR-DSS-PER-080

SAVOIR.MMS.PER.0190

MMS initialisation time

The MMS shall be available for receiving commands and storing data <MMS initialization time> seconds after power-on or reset.

Rationale: *The initialization time of the MMS has an impact on the availability of the complete system.*

Comment: *The requirement includes the time required by the FMS or other DMS to be initialized.*

Verification Method: T

Parent: SAVOIR-DSS-PER-090

SAVOIR.MMS.PER.0200

MMS maximum number of opened files

The MMS shall support a maximum of <MMS simultaneous open files> files to be open at a time.

Comment: *Depending on the implementation, files handled by a copy operation may be counted as open ones.*

Verification Method: T

8.5 FDIR

This section identifies the Fault Detection Isolation and Recovery requirements.

SAVOIR.MMS.FDIR.0100

MMS Fault Detection

The MMS shall implement error detection mechanisms capable of detecting and reporting errors according to the required performance levels.

Rationale: *Radiation may cause bit flip that need to be detected and reported for analysis, isolation and possible recovery.*

Comment: *A scrubbing mechanism may be implemented to avoid error accumulation. The rate may be different depending on the storage areas technology and the mission needs.*

Comment: *Performances are defined in section 8.4.*

Verification Method: T

Parent: SAVOIR-DSS-FDIR-010

SAVOIR.MMS.FDIR.0110

MMS non-destructive self-test

The MM shall provide the capability to perform a non-destructive self-test of all storage media.

Rationale: *To check the correctness of the storage area.*

Comment: *Non-destructive test means that it shall be possible to perform verifications on SAUs without losing stored data (e.g. by temporary saving data before checks and restoring it after). The mechanism is implementation dependent. It can be implemented at higher (MMS) level, e.g. to check SAUs and lower (FS) level, e.g. to check BAUs and file content.*

Verification Method: T

Parent: SAVOIR-DSS-FDIR-020

SAVOIR.MMS.FDIR.0120

MMS list of errors

The MMS shall maintain a list of all the errors affecting the storage media in a non-volatile memory.

Rationale: *The list of errors is used for reporting and isolation.*

Comment: *The list of errors can be maintained at higher (MMS) level, i.e. list of invalid SAUs and lower (FS) level, i.e. list of invalid BAUs.*

Verification Method: RoD, T

Parent: SAVOIR-DSS-FDIR-030

SAVOIR.MMS.FDIR.0130

MMS Software redundancy

The MM shall support maintaining at least two copies of the MMS Software in non-volatile memory.

Rationale: *For redundancy purpose and configuration management.*

Verification Method: RoD

Parent: SAVOIR-DSS-FDIR-040

SAVOIR.MMS.FDIR.0140

MMS Software version used

The version of the MMS Software to be used shall be defined by a flag (HW or SW), and selectable by telecommand.

Rationale: *For configurability purposes.*

Verification Method: T

Parent: SAVOIR-DSS-FDIR-050

SAVOIR.MMS.FDIR.0150

MMS SAU exclusion

The MMS shall support autonomous exclusion of SAU detected as definitively faulty or anticipated to be faulty in the future from the MMS storage media capacity.

Rationale: *To prevent these SAU from being used by upper layers for data storage.*

Comment: *Anticipated to be faulty means that errors have been detected and corrected more times than a given threshold indicating probable failure.*

Comment: *When SAU is faulty, MMS can either replace the SAU from the list of SAUs it appears for a BAU or inform the DMS to declare the complete BAU as faulty.*

Verification Method: T

Parent: SAVOIR-DSS-FDIR-060

SAVOIR.MMS.FDIR.0160

MMS SAU content saving

When excluding an anticipated to be faulty SAU, the MMS shall support moving the data it contains to another location.

Rationale: *To prevent data loss when isolating faulty SAU.*

Comment: *Depending on the implementation, the File System need to be informed when it required to update its internal organization structures to address the BAU containing the new SAU, and thus keep file integrity.*

Verification Method: T

Parent: SAVOIR-DSS-FDIR-070

SAVOIR.MMS.FDIR.0170

MMS SAU Isolation

When isolating a SAU, the MMS shall indicate it to the Fault isolation mechanism which will identify the potential affected area and handle the situation accordingly with support of corresponding DMS.

Rationale: *To maintain the data integrity.*

Comment: *If the SAU is being isolated as a preventive action, it will allow the DMS to duplicate the data it contains ensuring the reliability of the File or Packet content before isolating it.*

Comment: *Multiple mechanisms can be defined, for instance: after duplicating the content of SAU, a checksum can be triggered so the validity of the duplicated SAU can be verified, and the BAU is then reconfigured with this duplicated SAU, replacing the failed one.*

Verification Method: T

SAVOIR.MMS.FDIR.0180

Stoppable DMS

The MMS shall be able to stop the DMS without loss of data already stored.

Rationale: *DMS (e.g. FMS or PMS) needs to be stopped in case of system contingency (power shortage, CPU load balancing) or to disable some control mechanism it might implement (for instance write/read protections to ensure that critical software update are possible).*

Comment: *The different DMS shall be stopped after having unmapped the interfaces. For FMS, this also reauired to close all opened files after having flushed the buffer into the files. The data received after the stop operation are lost.*

Verification Method: T

SAVOIR.MMS.FDIR.0190

DMS Status

The MMS shall maintain the status of the DMS whether it is available or not.

Rationale: *DMS availability is a condition to handle or not handle an access, or the way to access it.*

Comment: *Several mechanisms can be designed. If the DMS is not available, it can be decided to abort a write file operation, or to delay it until DMS is available. Another possibility would be to monitor the primitives received back from the DMS after a write request, however this does not apply to file mapping (which doesn't inform the originating User Entity when data is successfully written).*

Verification Method: T

SAVOIR.MMS.FDIR.0200

DMS Status access

The MMS shall make available the status of the DMS to any User Entity.

Rationale: *For User Entity to know the status of the DMS.*

Verification Method: T

SAVOIR.MMS.FDIR.0210

FMS starting mode

The OBC shall be able to start the DMS and select whether:
- the context shall be recovered or
- the default initial configuration shall be used.

Rationale: *DMS, when started, can be restarted in normal mode, meaning restored from its previous state (same allocation, same Store and block definition, access to data is preserved). It can also be possible to restart from scratch.*

Comment: *If several contexts are stored and data are backed up, it is possible to implement a mechanism like a restore point. In such case the DMS will be started at an intermediate state.*

Verification Method: T

SAVOIR.MMS.FDIR.0220

MMS traces

When performing memory administration operation, the MMS shall store the related trace (parameters, address, size and identifier) in a non-volatile memory.
Memory administration operations are:

- SAU to BAU allocation
- SAU isolation
- BAU creation, modification and deletion,
- STORE creation, modification and deletion
- File allocation table (or equivalent) modification.

Rationale: *To ensure that the structure of the MMS can be recovered (layering, mapping& organization).*

Comment: *The trace is used to restart the FMS in case of its full restart or after satellite power shortage. The traces shall only cover the data that are not yet saved into a non-volatile memory.*

Verification Method: T

SAVOIR.MMS.FDIR.0230

FMS File Status persistance

When a reconfiguration occurs or in any case making the FMS unavailable, the File Status shall be maintained in their state.

Rationale: *File statuses are attached to the file and describe its characteristics.*

Comment: *In case the FMS is made unavailable while a file is opened or locked, the related file descriptor and lock owners and type will therefore be the same when FMS is available again. This mechanism aims at ensuring service continuity.*

Verification Method: T

8.6 User Entity service

The Data Storage services rely on an external service to identify the User Entity associated to a request in order to determine if the User Entity is allowed to perform an operation.

The User Entity service shall be able to provide the identity of the User Entity that is calling a service provided by the MMS. When the MMS is fully local (User Entities and Data Storage implemented on a same equipment), the identification of the User Entity can be implicit and rely on services provided by the underlying operating system, e.g. task or thread identifier. When remote User Entities are accessing the MMS, the identification can be implemented within the Remote File Management Protocol and the Remote Block Access Protocol.

The identification of User Entities can also be implemented at the level of the different services by means of a dedicated parameter.

The MMS provides a service that can be used by a User Entity to retrieve its unique identifier. This can be used when the User Entity identifier needs to be used in arguments when calling MMS services.

8.6.1 Service definition

SAVOIR.MMS.UES.0100

User Entity identification retrieval

The MMS shall be able to provide a User Entity a unique identifier.

Rationale: *The User Entity can use the identifier in all subsequent accesses to the MMS.*

Comment: *Note that a User Entity is any user of the MMS.*

Verification Method: T

8.6.2 Service parameters

SAVOIR.MMS.UES.0110

MMS Transaction Identifier

The MMS Transaction Identifier shall be a value, assigned by the invoking User Entity, which is subsequently used to associate indication primitives with the causal request primitives.

Rationale: *The transaction Identifier ensures the correct management of requests in a multi-User Entity system and the User Entity is able to correlate all indications to a service request.*

Verification Method: T

SAVOIR.MMS.UES.0120

MMS Result Metadata

The MMS Result Metadata parameter shall be used to provide information generated by the MMS provider to the User Entity regarding result of the execution of a request.

Verification Method: T

SAVOIR.MMS.UES.0130

User Entity Identifier

The User Entity Identifier parameter shall uniquely identify a User Entity.

Verification Method: RoD, T

8.6.2.1 GET_USER_ENTITY_IDENTIFIER.request

8.6.2.1.1 Function

SAVOIR.MMS.UES.0200

GET_USER_ENTITY_IDENTIFIER Function

The GET_USER_ENTITY_IDENTIFIER.request primitive shall be passed to the MMS provider to request its User Entity identifier.

Rationale: The User Entity identifier is required to uniquely a User Entity. It is used by all services to check that the resource is not locked by another User Entity.

Verification Method: T

8.6.2.1.2 Semantics

SAVOIR.MMS.UES.0210

GET_USER_ENTITY_IDENTIFIER Semantics

The GET_USER_ENTITY_IDENTIFIER.request primitive shall use the following semantics:

GET_USER_ENTITY_IDENTIFIER.request(MMS Transaction Identifier)

Verification Method: T

8.6.2.1.3 When generated

SAVOIR.MMS.UES.0220

GET_USER_ENTITY_IDENTIFIER When generated

The GET_USER_ENTITY_IDENTIFIER.request primitive shall be passed to the MMS provider to request a unique User Entity identifier to be used.

Verification Method: T

8.6.2.1.4 Effect on Receipt

SAVOIR.MMS.UES.0230

GET_USER_ENTITY_IDENTIFIER Effect on Receipt

Receipt of the GET_USER_ENTITY_IDENTIFIER.request primitive shall cause the MMS provider to generate a unique identifier that can be used by the User Entity to identify itself to the MMS.

Verification Method: T

8.6.2.1.5 Additional constraints

None

8.6.2.2 GET_USER_ENTITY_IDENTIFIER.indication

8.6.2.2.1 Function

SAVOIR.MMS.UES.0240

GET_USER_ENTITY_IDENTIFIER.indication Function

The GET_USER_ENTITY_IDENTIFIER.indication primitive shall be used to return the User Entity Identifier.

Verification Method: T

8.6.2.2.2 Semantics

SAVOIR.MMS.UES.0250

GET_USER_ENTITY_IDENTIFIER.indication Semantics

The GET_USER_ENTITY_IDENTIFIER.indication primitive shall use the following semantics:

GET_USER_ENTITY_IDENTIFIER.indication(MMS Transaction Identifier, User Entity Identifier, MMS Result Metadata)

Verification Method: T

8.6.2.2.3 When generated

SAVOIR.MMS.UES.0260

GET_USER_ENTITY_IDENTIFIER.indication When generated

The GET_USER_ENTITY_IDENTIFIER.indication primitive shall be passed by the MMS provider to the receiving User Entity in response to a GET_USER_ENTITY_IDENTIFIER.request with MMS Result Metadata indicating if the request was executed successfully or not.

Comment: The operation should be always successful except if the number of User Entity is restricted due to mission requirements.

Verification Method: T

SAVOIR.MMS.UES.0270

GET_USER_ENTITY_IDENTIFIER.indication When generated success

When GET_USER_ENTITY_IDENTIFIER.indication primitive is successful, the service returned the unique identifier in the User Entity Identifier parameter.

Verification Method: T

SAVOIR.MMS.UES.0280

GET_USER_ENTITY_IDENTIFIER.indication When generated failure

When GET_USER_ENTITY_IDENTIFIER.indication primitive is unsuccessful, the MMS Result Metadata shall provide the reason of the failure.

Verification Method: T

8.6.2.2.4 Effect on Receipt

The response of the User Entity to a GET_USER_ENTITY_IDENTIFIER.indication is unspecified.

8.6.2.2.5 Additional constraints

None