

Scalable Intelligent Video Server System

<i>Title</i>	High Speed Data Mover Software Prototype Publishable version
<i>Revision</i>	D
<i>Deliverable #</i>	D8.1
<i>Author</i>	Fabrice Lecointe, Nicolas Sapin
<i>Company</i>	Hi-Stor Technologies SA
<i>Date</i>	19/04/2007
<i>Filename</i>	SIVSS_D8.1_V4.doc
<i>Dissemination²</i>	PU

REVISION	DATE	DESCRIPTION
A	12/02/2006	Created by FL
B	28/02/2006	updated
C	31/07/2006	Finished
D	19/04/2007	Public version

² **CO** = Confidential (only for members of the consortium + EC); **RE** = Restricted to a stated circulation list (+ EC)
[replace this footnote with the list]; **PP** = Restricted to other FP6 participants (+ EC); **PU** = Public

TABLE OF CONTENTS

1.	INTRODUCTION	4
2.	ARCHITECTURE AND DETAILED SPECIFICATIONS	4
3.	DATA MOVER API.....	9
3.1.	ARCHIVE AND RESTORE	9
3.1.1.	Archive.....	9
3.1.2.	Restore.....	9
3.1.3.	Archive and Restore use case diagram.....	10
3.2.	COPY	11
3.2.1.	Copy use case diagram.....	11
3.3.	DELFROMDISK	13
3.3.1.	DelFromDisk use case diagram	13
3.4.	DELFROMARCHIVE	14
3.4.1.	DelFromArchive use case diagram	14
3.5.	QUERYMD	15
3.5.1.	QueryMD use case diagram.....	16
4.	DATA MOVER UML DESCRIPTION	17
4.1.	DMC UML DESCRIPTION.....	17
4.2.	DME UML DESCRIPTION.....	17
4.3.	DMSL UML DESCRIPTION.....	17
4.4.	DMSF UML DESCRIPTION.....	17
5.	DATA MOVER GUI GENERAL DESCRIPTION.....	18
5.1.	MODULE PERSPECTIVE	18
5.2.	MODULE INTERACTION.....	18
5.3.	GENERAL CONSTRAINTS	18
6.	DATA MOVER GUI SPECIFICATIONS.....	19
6.1.	DATAMOVER GENERAL STATE FUNCTIONS PANEL	19
6.2.	PERFORMANCE PANEL.....	20
6.3.	HARDWARE MONITORING PANEL.....	21
7.	DATAMOVER PERFORMANCE TESTS.....	22
7.1.	TEST STRATEGIES DESCRIPTION	22
7.1.1.	Tested Data Mover modules.....	23
7.1.2.	Description of the Environment.....	23
7.1.3.	Strategy.....	23
7.1.3.1.	General strategy.....	23
7.1.3.2.	Corrupt components	23
7.1.4.	Extra Features	24
7.1.4.1.	Commands from external agent to DMC.....	24
7.1.4.2.	Commands from MAM to DMSL	24
7.1.5.	USED DATA SETS	24
7.1.6.	HARDWARE configuration for performances testing	25
7.2.	TESTS RESULTS	25
8.	CONCLUSIONS	33



8.1.	FUNCTIONAL TESTS CONCLUSIONS	33
8.2.	PERFORMANCE ANALYSIS AND FURTHER WORK.....	33

1. INTRODUCTION

This deliverable is linked with the Deliverable 7.1 that is the feasibility study of the Data Mover. This D7.1 deliverable presents:

- The Data Mover market: the products from the competition, the functionalities and the differences between of these product and SIVSS needs
- The architecture of the Data Mover built for SIVSS project and the reasons why this architecture achieves performance, scalability, cost-effectiveness and high-availability
- The functionalities that are offered by the Data Mover

The present deliverable comprises the following parts:

1. This first chapter introduces the document structure
2. The second chapter of the document reminds the architecture of the Data mover and presents the different software modules that compose the Data mover
3. The third chapter describes the API between the Data Mover and the application that want to use the Data Mover that is to say the MAM from Thomson
4. The fourth chapter describes the low level objects that composed the different modules of the Data Mover software prototype
5. The fifth and sixth chapters present the Data Mover GUI
6. The seventh chapter presents the performance achieved and the test results

To end this introduction, the information provided in this deliverable applies to the Data Mover prototype, it will be detailed in the next deliverable D8.2 and can evolve till the delivery of the D8.2 document.

2. ARCHITECTURE AND DETAILED SPECIFICATIONS

The Data Mover architecture comprises 2 functional modules:

-One single **DMC** (Data Mover Controller) module that receives commands from the **Media Asset Manager** designed by Thomson, translates these commands into one or several jobs and spreads these jobs to the **DMEs** (Data Mover Engine). This single DMC communicates with MAM and DME via TCP/IP protocol but is not linked to the shared file system or SCSI tape drive and so, can be installed anywhere on SIVSS architecture.

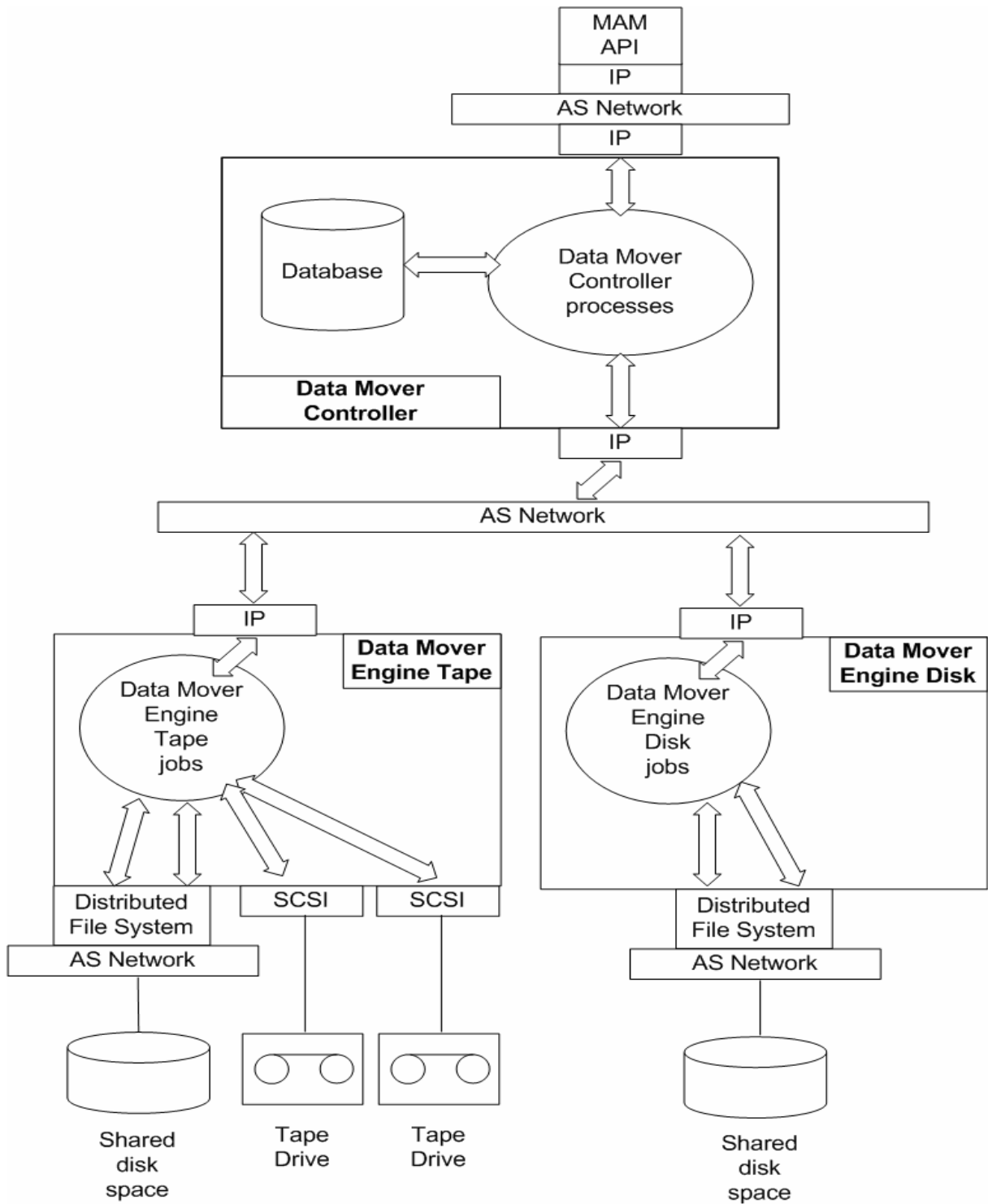
Remark: The same name “Data Mover Controller” has been given to this logical module and a software module that will be presented later in this document.

-Two types of **DME** modules that listen to the **DMC** and execute jobs using storage peripherals:

1. The Data Mover Engine Tape: This Data Mover module is the most important DME. It moves video files between the shared file system and the archive systems (tapes). This function allows the lowest cost of storage for video facilities as long as tapes are less expensive than disk systems. The Data Mover Engines Tape have to manage data transfers between shared file system and SCSI tape drives and as a consequence, one DME has to be installed on each server where a tape drive is connected via a SCSI link.
2. The Data Mover Engine Disk: This Data Mover module is responsible for archiving/restoring video data between the disk space accessible to video applications

(ingest, player ...) and the disk space reserved for archive. The disk space configured to host archived data can be very low cost, low performance and directly attached to DM nodes but should protect the archived data against a disk failure with a RAID configuration.

The following diagram presents these 2 functional modules and the video environment, that is to say the shared file system, tape drives, disk systems and the Media Asset Manager:



These high-level logical modules diagram can be broken down into a more detailed software modules diagram.

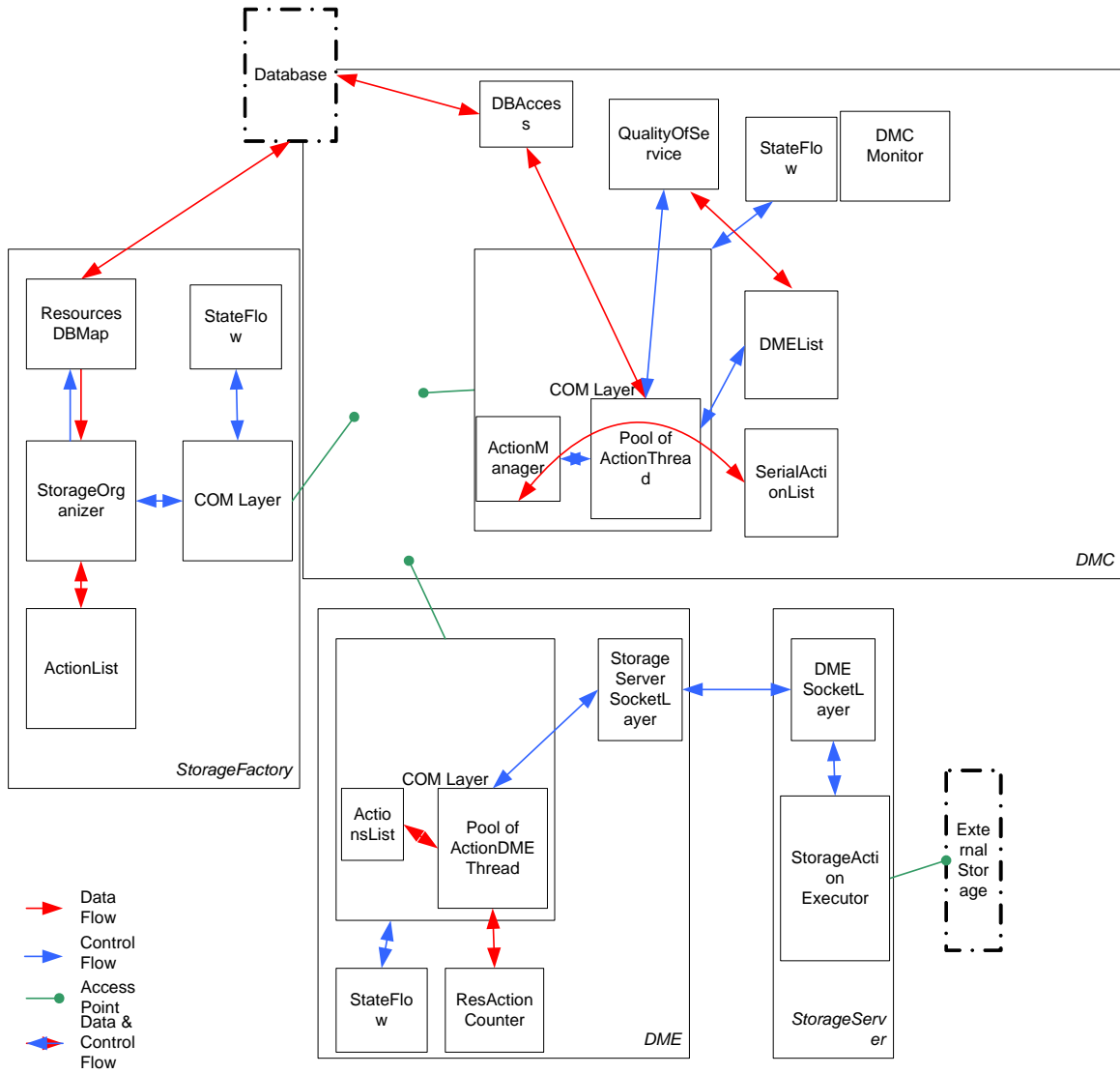
The DMC **logical** module can be broken down into 4 **software** modules:

1. one single **DMSL** module (Data Mover Service Layer) listens to messages coming from the MAM and translates them into several simplified commands to the **DMC** (Data Mover Controller)
2. one single **DMC** (Data Mover Controller) software module receives commands from the **DMSL**, communicates with the **DMSF** (Data Mover Storage Factory) to allocate storage resources for the jobs that must be performed, and spreads the jobs to the **DMEs** (Data Mover Engine)
3. One single **DMSF** (Data Mover Storage Factory) that deals with the discovery, the allocation and the management of storage peripherals owned by the **DMEs** (Data Mover Engines)
4. One single **database** that is used to store metadata about Archive Units, locations, transfers done, running or to be done and so on.

The DME logical module can be broken down into 2 software modules:

1. The **DME** (Data Mover Engine) software module listens to the **DMC** software module and executes the jobs with its **DMSS** (Data Mover Storage Server) modules. One **DME** software module is mandatory and has to be installed on each server where the Data Mover has to manage one (or more) storage peripheral(s).
2. The **DMSS** (Data Mover Storage Server) modules listen to DME commands and execute low level reading and writing tasks.

All these modules are represented on this architecture diagram:



3. DATA MOVER API

Hi-Stor software development team used UML object-oriented method to specify the Data Mover system. The first step of the specification included a high-level system approach with the capture of use cases and message sequences.

This paragraph describes the Data Mover API that can be invoked by SIVSS facility applications and the use case diagrams associated with them. The set is not complete but is the one that was necessary to build the Data Mover prototype for IBC show. The complete set of API will be detailed in D8.2 deliverable.

3.1. Archive and Restore

3.1.1. Archive

Action_ID; ACK/ABORT; MAM_AU_ID = **Archive**(Path, [Array of path/files to archive], [Array of path/files to exclude of the archive] , priority , QOS)

This command archives folders and files from the shared File System to the Archive (tapes). Once the folders or files are archived, the original information will be deleted from the shared File System.

Path is an absolute path.

[Array of path/files to archive] is a list of either relative directories or files separated by a semicolon. For each file or directory in the array, the complete path will be created by preceding the path/file with the Path parameter. All elements present in the Array will be archived.

[Array of path/files to exclude of the archive] is a list of elements separated by a semicolon that will not be included in the archive.

The moment when the migration will be effectively executed depends on the **Priority** parameter and the other requests that are in the Data mover queue.

The **QOS** will define the maximum number of Tape Drives that will be used in parallel to perform the data transfer.

The command returns an **Action_ID** number that identifies the request (used to monitor the progress of the command execution) followed by an acknowledgement of the command reception and the **MAM_AU_ID** that will be affected to the archive unit to be created.

3.1.2. Restore

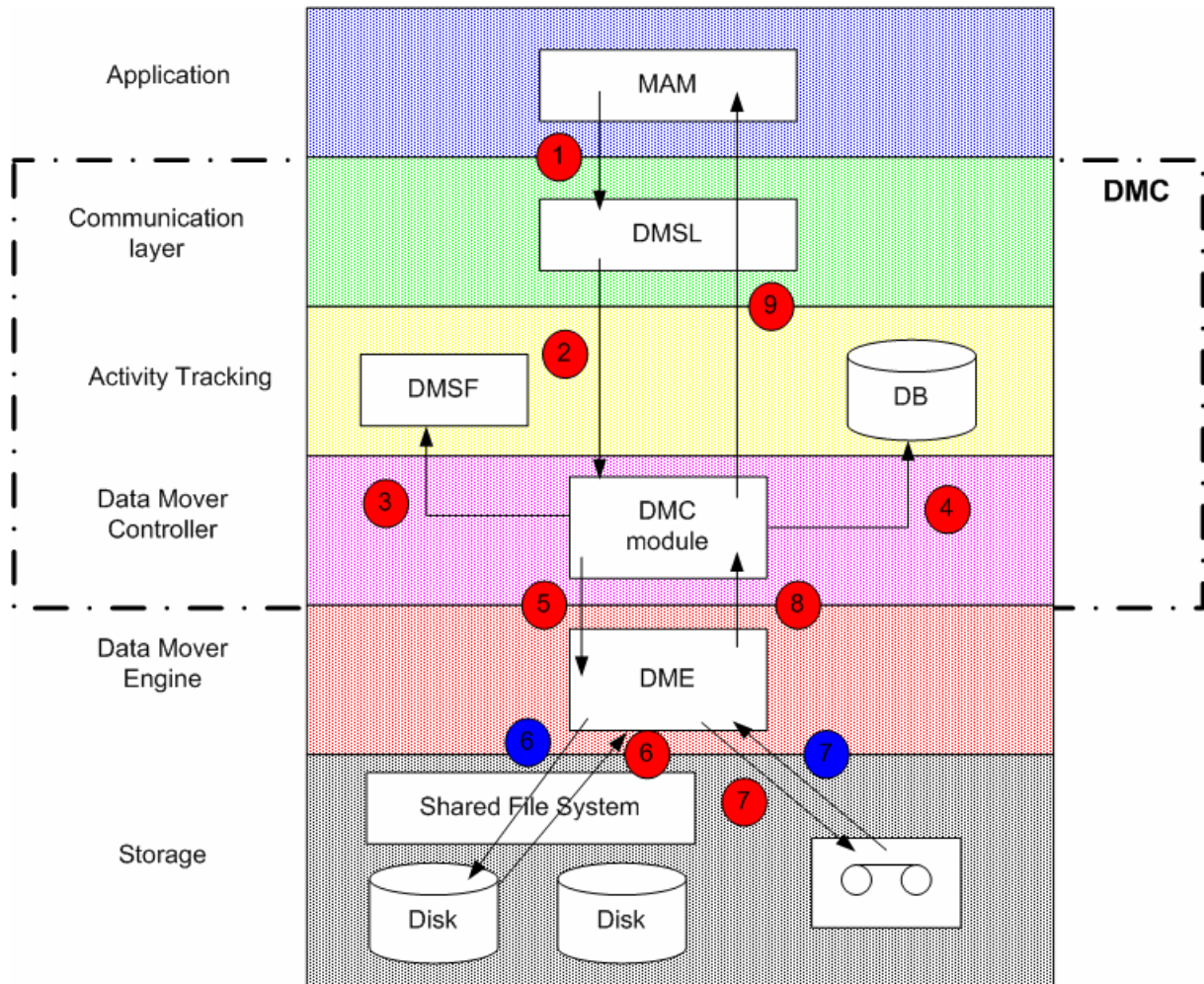
Action_ID; ACK/ABORT = **Restore** (AU_ID , destination_path , 0)

This command restores in the absolute path **destination_path** the Archive Unit identified by the parameter **AU_ID**. The third parameter allows to indicate the criticality of the request. For the Data mover prototype this parameter is set to 0 because priorities are not implemented yet but it will be settable to a valid value in the final version.

The command returns an **Action_ID** number that identifies the request (used for monitoring progress of execution of the command) followed by an acknowledgement of the command reception.

3.1.3. Archive and Restore use case diagram

This diagram presents the different steps that are comprised in an Archive or Restore action:



Step	Description
1	The Media Asset Manager sends an archive or restore request to the Data Mover Service Layer. The Service Layer translates this request into several simpler requests. <u>Example:</u> When the MAM asks the Data Mover to archive 2 files with a QOS of 2, the DMSL will transmit 2 requests to the DMC to archive 1 file.
2	The Data Mover Controller software module receives the requests from the Data Mover Service Layer
3	The Data Mover Controller software module communicates with the Data Mover Storage Factory to allocate Data Mover Engines Tape and storage peripherals for the needs of these requests
4	The Data Mover Controller software module inserts or looks for the location of the data it has to archive/restore and other useful metadata to provide/store
5	The Data Mover Controller software module sends several requests to the Data Mover Engines that have to perform the archive or restore actions
6 and 7 red (Archive)	The Data Mover Engines Tape read data from the shared file system space

only)	and write them to tape. Step 6 and 7 are executed in parallel: reading data from disk and writing them on tape is performed by a classical TAR command. Several Data Mover Engines Tape work in parallel to provide performance
6 and 7 blue (Restore only)	The Data Mover Engines Tape read data from the tapes and write them to the shared file system. Step 6 and 7 are executed in parallel: reading data from tape and writing them on disk is performed by a classical TAR command. Several Data Mover Engines Tape work in parallel to provide performance
Step 8	Each Data Mover Engine that has performed its task sends an acknowledgement to the Data Mover Controller software module
Step 9	As soon as the Data Mover Controller software modules has received all acknowledgments from the Data Mover Engines that are involved in the archive/restore process, it sends an acknowledgement to the Media Asset Manager via the Data Mover Service Layer

3.2. Copy

Action_ID; ACK/ABORT; MAM_AU_ID = **Copy** (path , [Array of path/files to archive], [Array of path/files to exclude of the archive], destination_path)

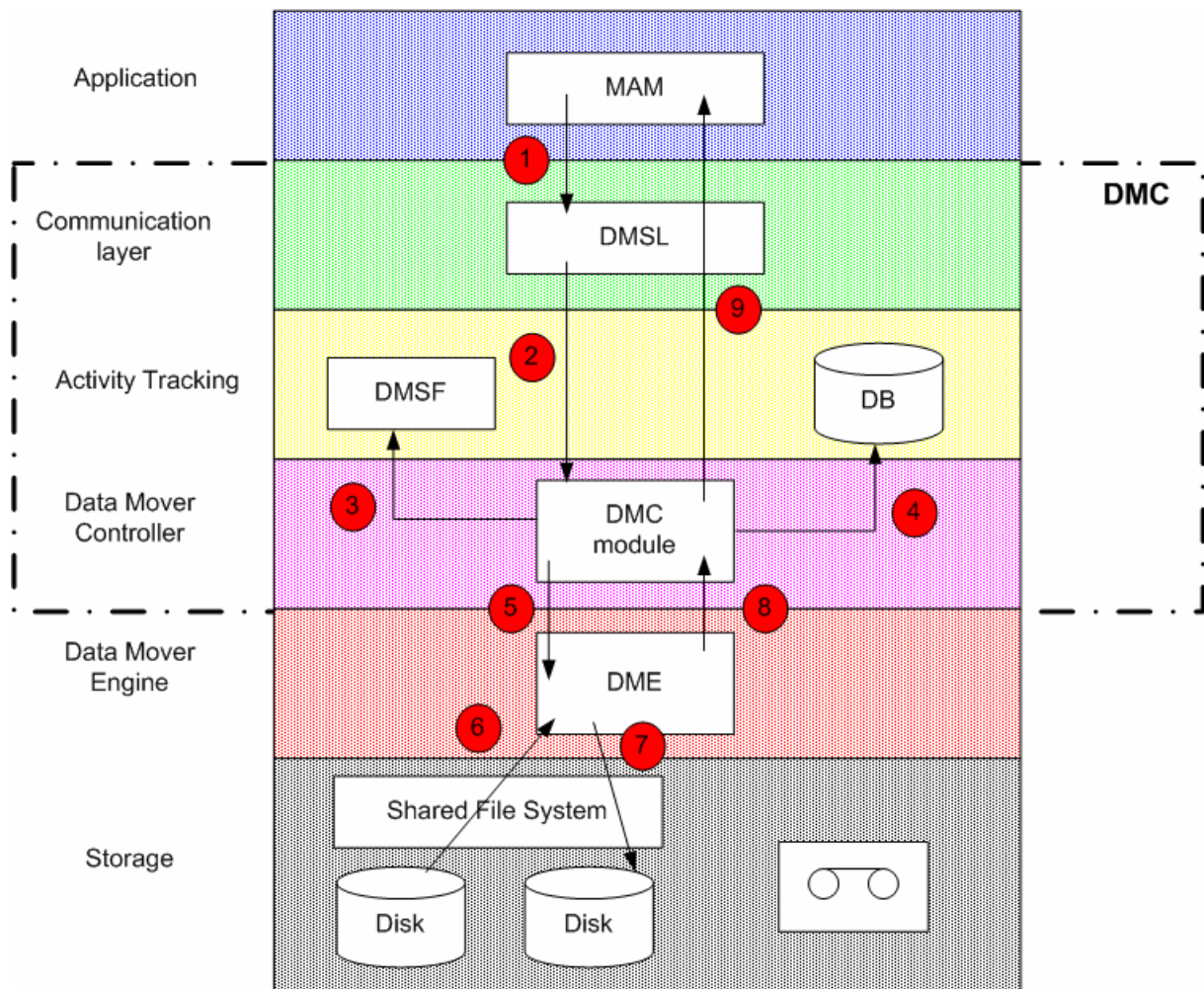
This command allows to archive on disk the directories/files list specified by the [**Array of path/files to archive**] parameter.

The **destination_path** gives the absolute path that will be used to generate the destination path for each directories/files to be copied.

The command returns an **Action_ID** number that identifies the request (used for monitoring progress of execution of the command) followed by an acknowledgement of the command reception and the **MAM_AU_ID** that will be affected to the archive unit to be created.

3.2.1. Copy use case diagram

This diagram presents the different steps that are comprised in a Copy action:



Step	Description
1	The Media Asset Manager sends a copy request to the Data Mover Service Layer. The Service Layer translates this request into several simpler requests.
2	The Data Mover Controller software module receives the requests from the DMSL
3	The Data Mover Controller software module communicates with the Data Mover Storage Factory to allocate Data Mover Engines Disk and storage peripherals for the needs of these requests
4	The Data Mover Controller software module inserts the location of the data it has to archive and other metadata to store
5	The Data Mover Controller software module sends several requests to the Data Mover Engines Disk that have to perform the copy actions
6 and 7	The Data Mover Engines Disk read data from the shared file system and write them to the new location on the disk space. Step 6 and 7 are executed in parallel. Several Data Mover Engines Disk work in parallel to provide performance
Step 8	Each Data Mover Engine Disk that has performed its task sends an acknowledgement to the Data Mover Controller software module
Step 9	As soon as the Data Mover Controller software modules has received all acknowledgments from the Data Mover Engines Disk that are involved in the copy action, it sends an acknowledgement to the Media Asset Manager via

	the Data Mover Service Layer
--	------------------------------

3.3. DelFromDisk

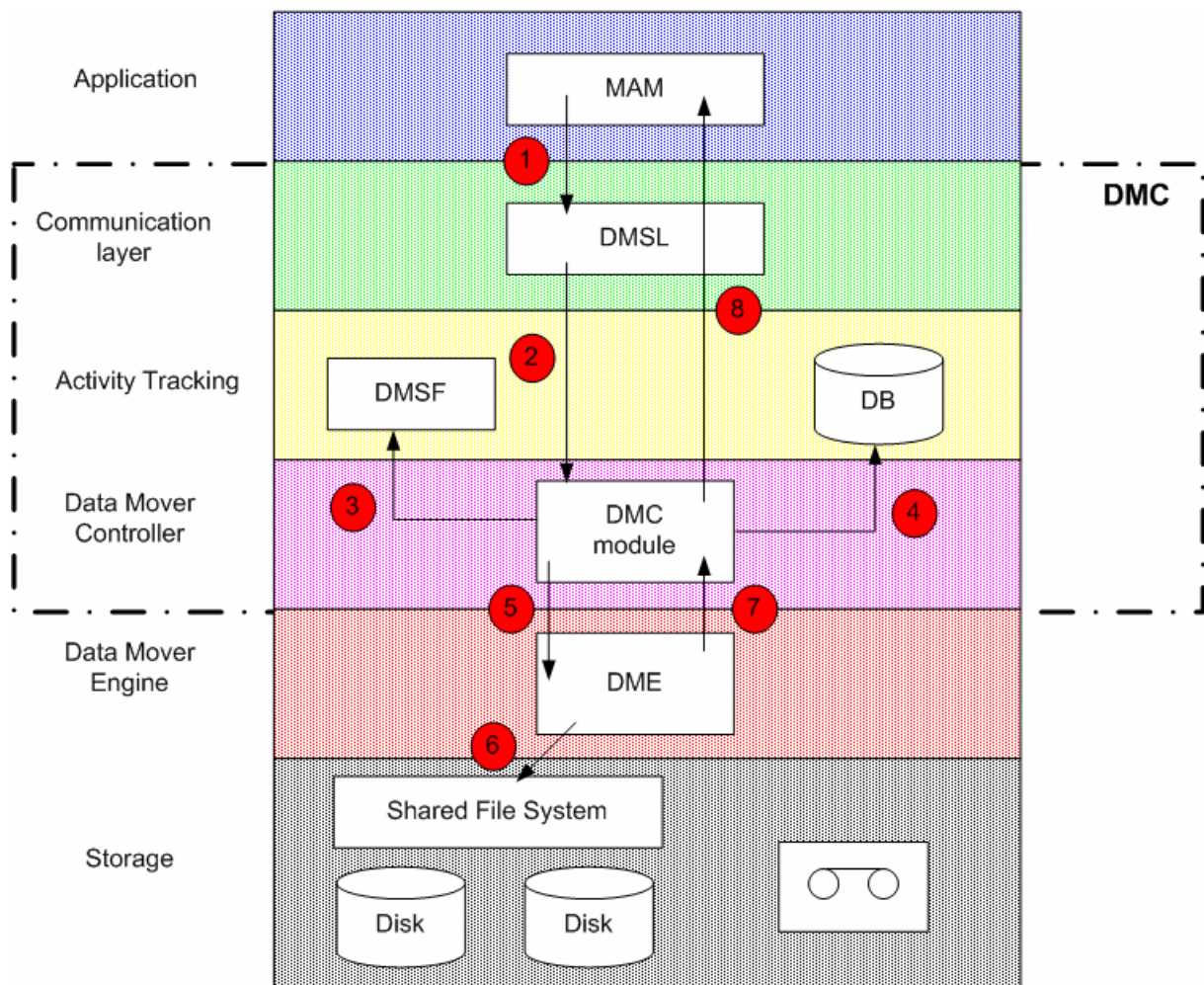
Action_ID; ACK/ABORT = **DelFromDisk** (AU_ID)

This command deletes an Archive Unit from the disk archive space identified by **AU_ID**.

The command returns an **Action_ID** number that identifies the request (used for monitoring progress of execution of the command) followed by an acknowledgement of the command reception.

3.3.1. DelFromDisk use case diagram

This diagram presents the different steps that are comprised in a DelFromDisk action:



Step	Description
1	The Media Asset Manager sends DelFromDisk request to the Data Mover Service Layer. The Service Layer translates this request into several simpler requests
2	The Data Mover Controller software module receives the requests from the DMSL
3	The Data Mover Controller software module communicates with the Data Mover Storage Factory to allocate storage Data Mover Engines Disk for the needs of these requests
4	The Data Mover Controller software module deletes the metadata concerning the Archive Unit that will be deleted from the system
5	The Data Mover Controller software module sends several requests to the Data Mover Engines Disk that have to perform the DelFromDisk actions
6	The Data Mover Engines Disk delete data from the archive disk space
7	Each Data Mover Engine Disk that has performed its task sends an acknowledgement to the Data Mover Controller software module
8	As soon as the Data Mover Controller software modules has received all acknowledgments from the Data Mover Engines Disk that are involved in the copy action, it sends an acknowledgement to the Media Asset Manager via the Data Mover Service Layer

3.4. DelFromArchive

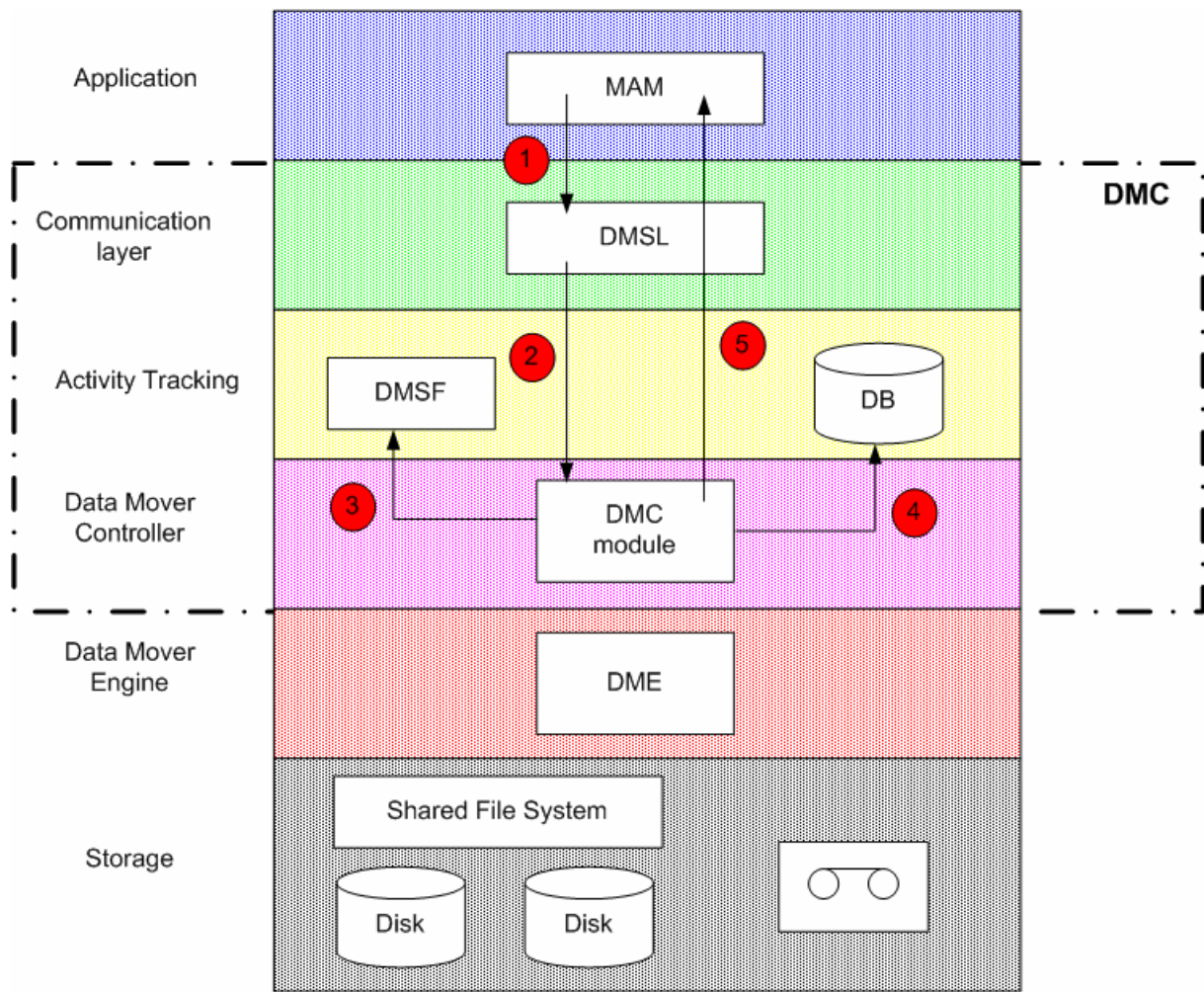
Action_ID; ACK/ABORT = **DelFromArchive** (AU_ID)

This command deletes from Tape the Archive Unit identified by **AU_ID**.

The command returns an **Action_ID** number that identifies the request (used for monitoring progress of execution of the command) followed by an acknowledgement of the command reception.

3.4.1. DelFromArchive use case diagram

This diagram presents the different steps that are comprised in a DelFromArchive action:



Step	Description
1	The Media Asset Manager sends a DelFromArchive request to the Data Mover Service Layer. The Service Layer translates this request into several simpler requests
2	The Data Mover Controller software module receives the requests from the DMSL
3	The Data Mover Controller software module communicates with the Data Mover Storage Factory to allocate Data Mover Engines Tape and storage peripherals for the needs of these requests
4	The Data Mover Controller software module removes the metadata concerning the Archive Unit that will be deleted from the system
5	The Data Mover Controller software module sends an acknowledgement to the Media Asset Manager via the Data Mover Service Layer

3.5. QueryMD

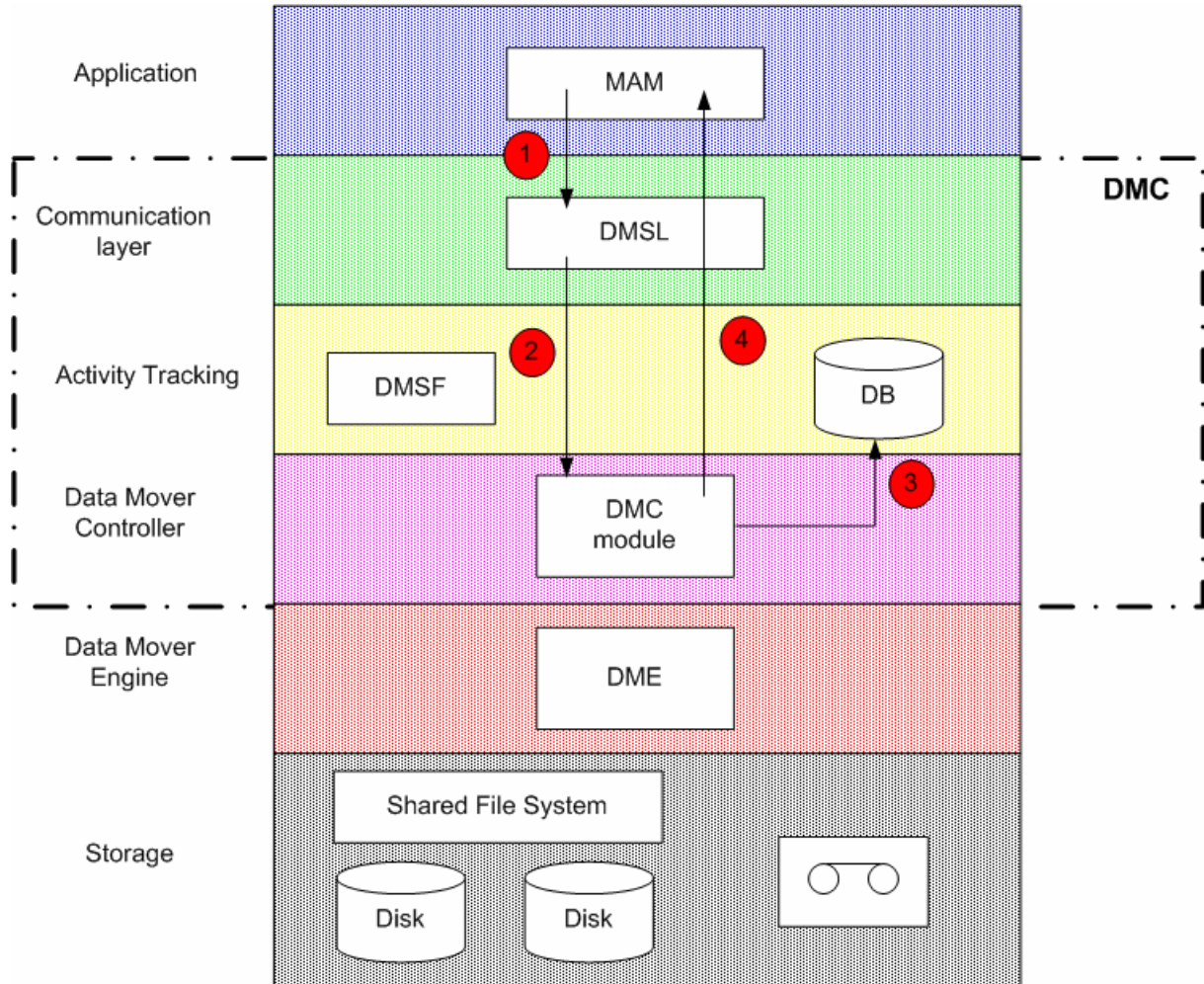
StrMetaData = **QueryMD** (AU_ID , MD_name)

This command allows the MAM to get a particular metadata named **MD_name** for the Archive Unit identified by **AU_ID**.

The returned value is a string that contains the metadata value.

3.5.1. QueryMD use case diagram

This diagram presents the different steps that are comprised in a QueryMD action:



Step	Description
1	The Media Asset Manager sends a QueryMD request to the Data Mover Service Layer that transmits the request to the Data Mover Controller software module
2	The Data Mover Controller software module receives the requests from the DMSL
3	The Data Mover Controller software module looks for the needed metadata concerning the Archive Unit
4	The Data Mover Controller software module sends the answer to the Media Asset Manager via the Data Mover Service Layer

4. DATA MOVER UML DESCRIPTION

In the next chapters, each software module (DMSL, DMC, DME, DMSF) are split into objects and their attributes are displayed following the UML description method.

UML diagrams are also available as files on CD for an easiest consultation.

The graphics that are presented hereafter show the detailed class diagrams of each module. The DMSL, DMC, DME and DMSF are implemented as several objects in the Data Mover computers memory. The relationship between the different objects that compose the Data Mover, the different methods that can be invoked to interact with the objects are presented in the following class diagrams:

4.1. *DMC UML description*

This section has been removed from this document to produce a public version.

4.2. *DME UML description*

This section has been removed from this document to produce a public version.

4.3. *DMSL UML description*

This section has been removed from this document to produce a public version.

4.4. *DMSF UML description*

This section has been removed from this document to produce a public version.

5. DATA MOVER GUI GENERAL DESCRIPTION

5.1. *Module perspective*

The GUI is tightly integrated with the Data Mover. Many users can use the GUI at the same time for monitoring or administration purposes for example.

5.2. *Module interaction*

The Data Mover is capable of showing information about each Data Mover module. As a result there is no specific module that has the ability to interact with the Data Mover GUI. Therefore the monitoring java methods are distributed on each Data Mover modules.

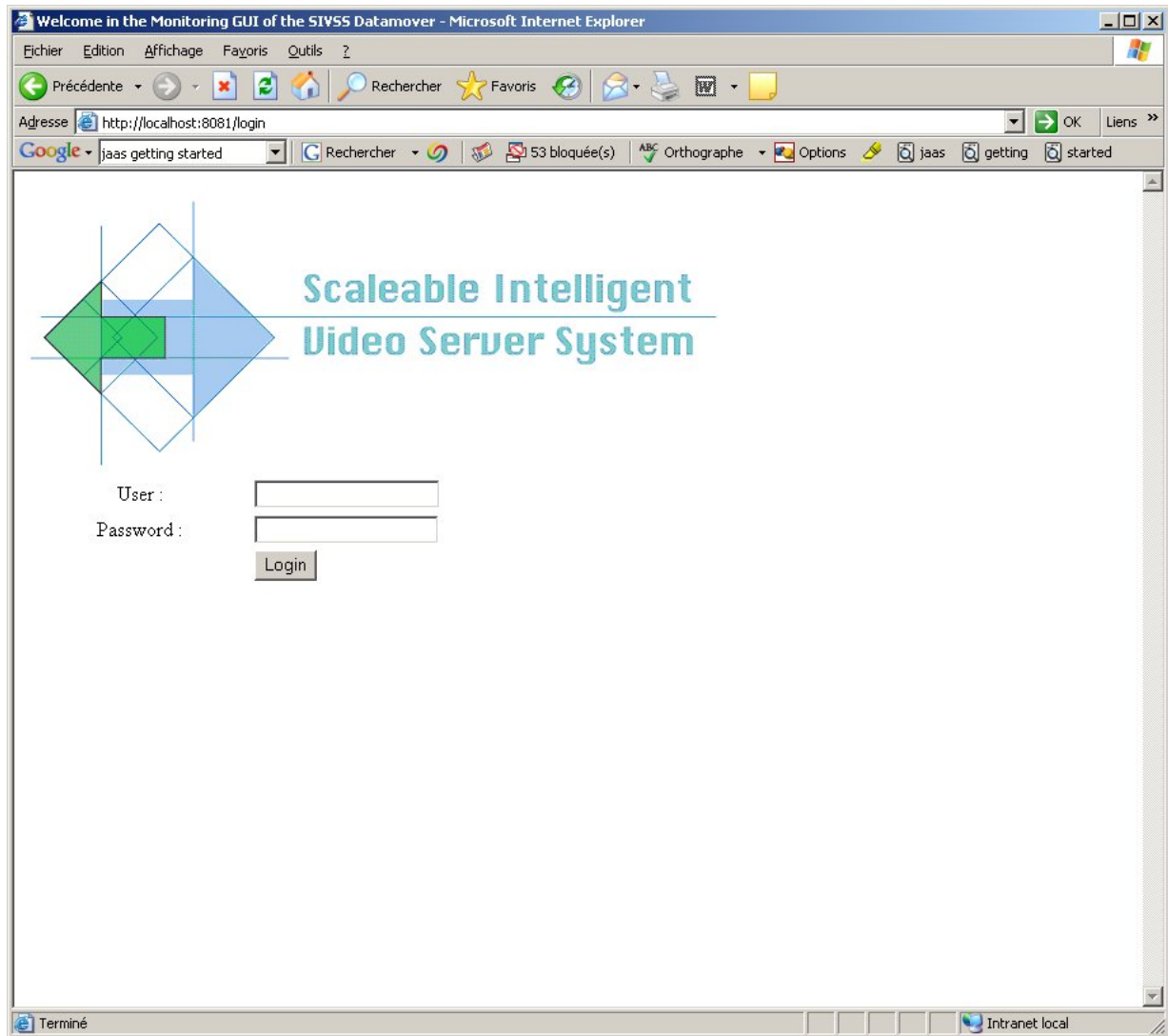
5.3. *General Constraints*

Several GUI can be launched in parallel and send requests simultaneously to the Data mover without any performance loss.

The GUI has to provide real time information about its modules. The Data Mover views and status must be updated frequently.

6. DATA MOVER GUI SPECIFICATIONS

The Data Mover administration can be done through an html browser. A login and a password are necessary to connect to the Data Mover GUI:



6.1. Datamover general state functions panel

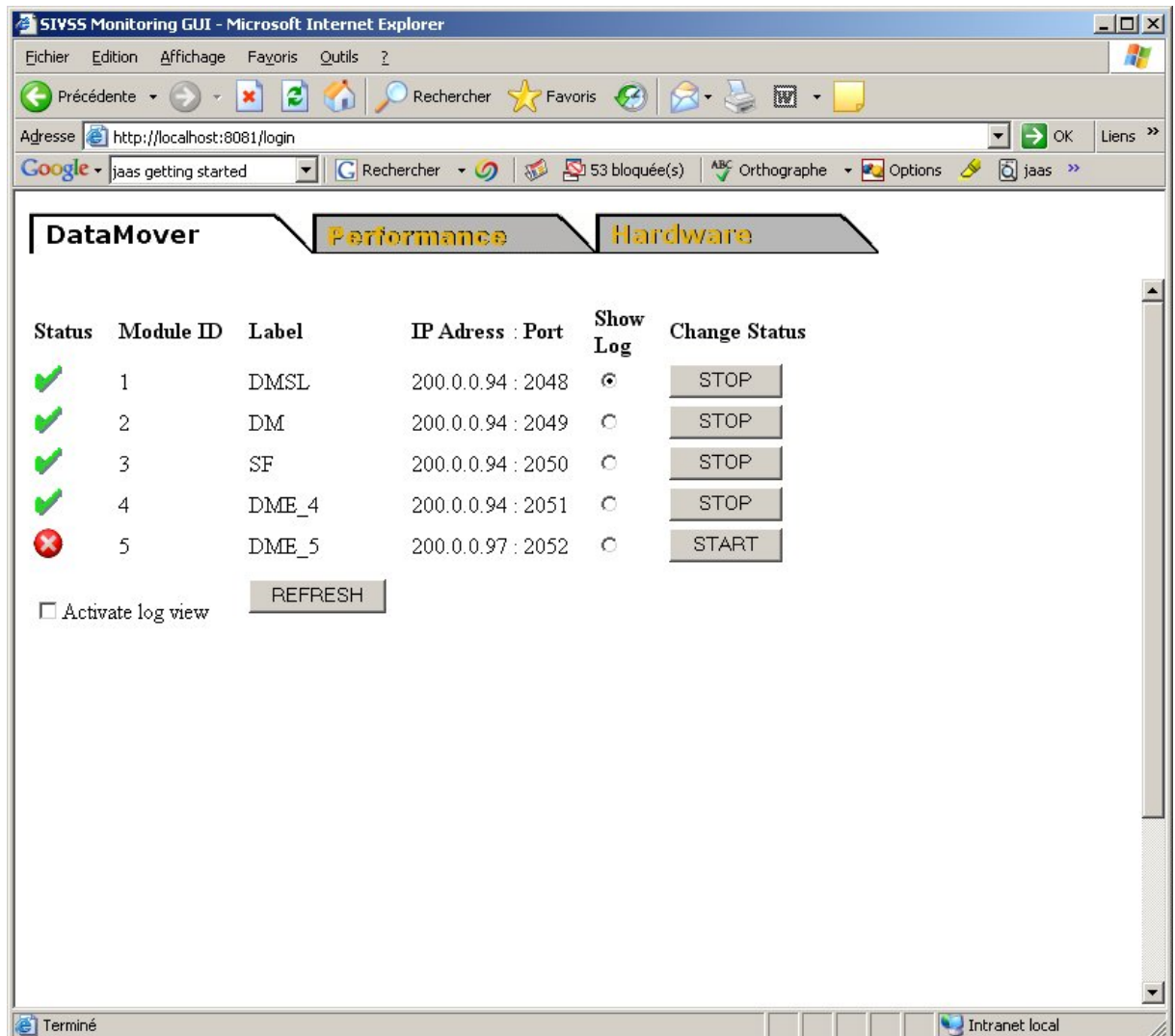
The GUI shows information about the different modules of the Data Mover:

- The Data Mover Controller
- The Data Mover Storage Factory
- The Data Mover Service Layer
- All the Data Mover Engines listed in the Data Mover configuration

Each of them is associated with a status (stopped, running, out of order).

The GUI offers the possibility to start or stop any of these modules and to access its log file.

This screenshot presents the Data Mover's modules monitoring panel:

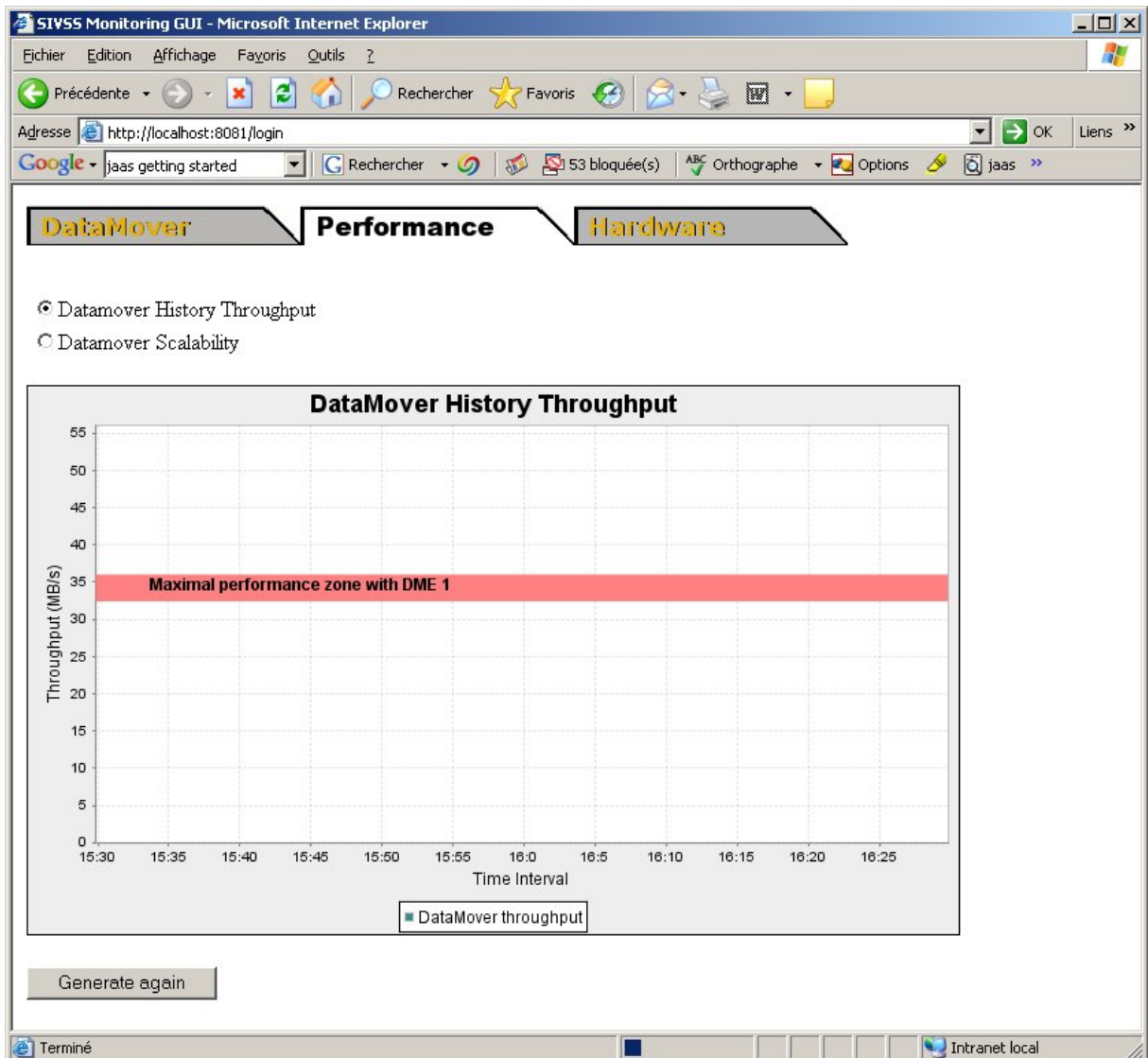


6.2. Performance panel

The GUI presents some graphics of the Data Mover activity. Two kind of diagrams are available:

1. The first one shows the global activity observed. The throughput delivered by the Data Mover is displayed on the first graphic as well as the performance that can be reached with the Data Mover configuration.
2. The second one shows a more detailed view that displays the activity of each DME

This screenshot presents the Data Mover performance panel:



6.3. Hardware monitoring panel

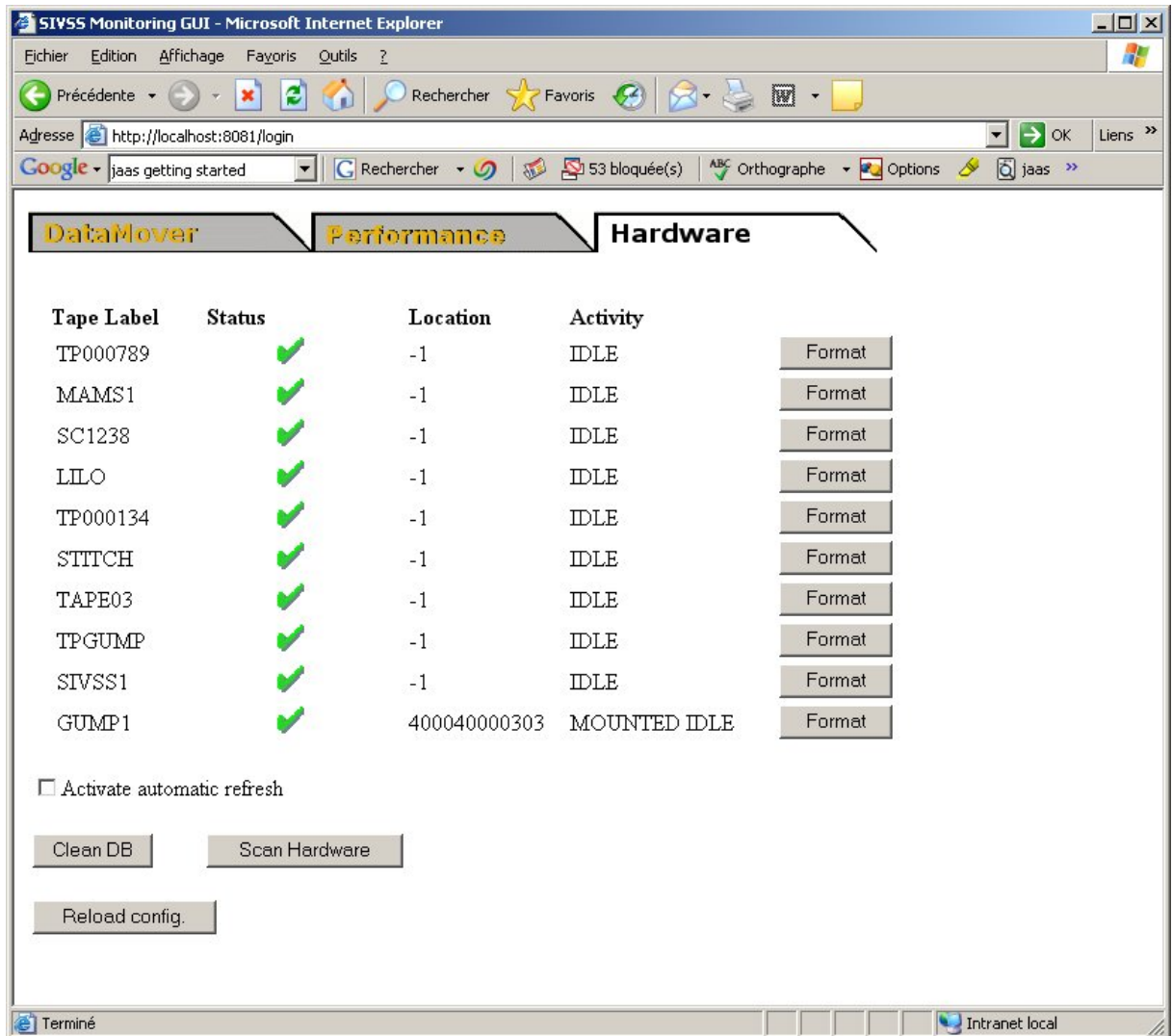
The GUI lists tape drives and disk file systems that can be accessed by the Data Mover. In addition a list of tapes known by the Data Mover system is displayed.

For each of these equipments the following information is displayed:

- Status
- Current activity
- Free space (if pertinent)
- For the tapes and file systems a list of the Archive Units hosted is displayed

The GUI allows to remove a tape or a drive from the list of available storage equipments (black-listing functionality). When a tape is blacklisted all its hosted Archive Units are marked as no longer available.

This screenshot presents the Hardware monitoring panel:



7. DATAMOVER PERFORMANCE TESTS

In this chapter, the Data Mover prototype test results are displayed. The results of these tests are the ones that have been performed on the Data Mover prototype used for IBC demonstrations. These tests allow to detect and correct some bugs and to complete the Data Mover validation process. Critical bugs were corrected before the show, other bugs has been or will be corrected in the Data Mover final version that will be presented in D8.2 deliverable.

7.1. Test strategies description

In the next paragraphs, the environment used to test the Data move is described:

7.1.1. Tested Data Mover modules

The components that have to be tested include all the components of the Data Mover that is to say:

- 1 The Storage Factory
- 2 The Data Mover Service Layer
- 3 The Data Mover Engine
- 4 The Data Mover Controller
- 5 The Storage Server

7.1.2. Description of the Environment

- Linux 2.6 – Red Hat/Fedora
- MySQL Server 4.0.24
- Java JRE 1.5

Different configurations of storage devices have been tested. Here is a set of environments that will appear in a lot of Test Case (Test Cases):

- Tests labelled TE 010 : Data Mover composed of 1 Data Mover Engines and 1 Tape Drive
- Tests labelled TE 020 : Data Mover composed of 1 Data Mover Engines and 2 Tape Drives
- Tests labelled TE 030 : Data Mover composed of 2 Data Mover Engines and 2 Tape Drives

7.1.3. Strategy

7.1.3.1. General strategy

- The Data Mover features have been tested by emulating a MAM and sending an almost exhaustive set of commands to the DME, to check components behavior:
 1. First, atomic independent commands have been sent to the Data mover. For each test case, a large range of input parameters have been tested to validate Data Mover basic behavior
 2. Then, complex request sequences have been sent to the Data Mover to create possibly ambiguous states
- Some components might be independently tested using java Netty2 command format

7.1.3.2. Corrupt components

Some test cases, especially those related to “*TS04x-Generic error cases while executing a command*» needed to use somehow corrupted components.

Such corruption might be for example, using a Storage Server that never send an answer message, in order to simulate a failure.

7.1.4. Extra Features

7.1.4.1. Commands from external agent to DMC

- Blacklist a Tape and a Tape Drive :

This command allows to remove a tape or a tape drive from the current Data Mover configuration. This command can be sent by the Quality Monitoring Software and has been simulated for testing purposes.

7.1.4.2. Commands from MAM to DMSL

- ArchiveWResidual:

Action_ID; ACK/ABORT; MAM_AU_ID = **ArchiveWResidual**(Path, [Array of path/files to archive], [Array of path/files to exclude of the archive] ,[Array of path/files to leave on disk], priority , QOS)

This special Archive command allows to archive video data leaving a percentage of the video files online on the disk space. This command targets the video on-demand market where video files have to be streamed rapidly without any scheduling. The command allows to use the video files left on disk space while the rest of the data is retrieved from tapes.

7.1.5. USED DATA SETS

Several sets of data have been used to perform the tests. Indeed, the Data Mover behavior depends on the type of files that it has to archive/restore. This data set aims to represent the types of files the Data mover will have to face in a broadcast environment, in a Video on Demand environment and so on.

All these files and directories are found in the *dataSets* directory (39 GB) that is attached with this validation plan.

The test cases are designed assuming that *dataSets* are situated in the root directory: /
Some test cases, especially when complex directories hierarchy have to be archived, might need the user to move some data inside *dataSets* before test processing.

- **Files:**

small1.avi	239.1 KB	small2.avi	2.3 MB
small3.avi	4.6 MB	small4.avi	6.3 MB
small5 .avi	7.3 MB	small6.avi	7.4 MB
small7.avi	8.4 MB	small8.avi	16 MB
big1 .avi	369.7 MB	big2.avi	350.5 MB
big3.avi	350.4 MB	big4.avi	350.5 MB
big5.avi	349.7 MB	big6.avi	350.1 MB
big7.avi	350.3 MB	big8.avi	349.8 MB
- **Directories:**
 - **dir-set1:** contains small files

small1.avi	239.1 KB
small2.avi	2.3 MB
small3.avi	4.6 MB
small4.avi	6.3 MB

- | | |
|-------------|--------|
| small5 .avi | 7.3 MB |
| small6.avi | 7.4 MB |
| small7.avi | 8.4 MB |
| small8.avi | 16 MB |
- **dir-set2:** contains big files

big1.avi	369.7 MB
big2.avi	350.5 MB
big3.avi	350.4 MB
big4.avi	350.5 MB
big5.avi	349.7 MB
big6.avi	350.1 MB
big7.avi	350.3 MB
big8.avi	349.8 MB
 - **dir-set3:** with mixed content

small1.avi	239.1 KB
small2.avi	2.3 MB
small3.avi	8.4 MB
small4.avi	16 MB
big1.avi	369.7 MB
big2.avi	350.5 MB
big3.avi	350.4 MB
big4.avi	350.5 MB
 - **dir-set4:** contains 113 files with insignificant size
Used to test the behavior on a big amount of files as well as the buffers limits in different components of the DM.

7.1.6. HARDWARE configuration for performances testing

1 computer i386:

CPU	3GHz
RAM	512 M
1 Hard drive IDE with 2 partitions	
Tape drive 1:	TANDBERG TS400
Tape drive 2(optional):	HP Ultrium 960

Number of DMEs and tape drives used to perform the tests are specified for each test case

7.2. Tests results

The test results are showed in the next graphics

/Scenario/Environment/Case		Durations (seconds)		RESTORE		RESTORE		Bandwidth (kb/s)		RESTORE		Config reminder				
		Tape access	Full process	Tape access	Full process	Tape access	Full process	Tape access	Full process	Tape access	Full process					
TS060	TE10	0010	1									1	1	yes		
		0020	2										1	1	no	
	TE20	0010	1	149	304	175	179	19,387	9,502	16,507	8,753		1	1	no	
		0020	2	144	295	179	179	20,060	9,792	16,131	8,701		1	2	yes	
	TE30	TE40	0010	1	316(1433504) and 316(1455168)	699	159(1433504) and 184(1455168)	4,536+4,604 =9,140	4,132	9,015+7,908 =16,923	8,372		1	2	no	
			0020	2	274(1455168) and 296(1433504)	594	111(1455168) and 108(1433504)	5,310+4,842 =10,152	4,863	13,109+13,273=26,382	10,659		2	2	yes	
TE50		0030	1										2	2	yes	
		0040	2										2	2	no	
TE60		TE70	0010	1										1	3	yes
			0020	2										1	3	no
	TE80	0030	1										2	3	yes	
		0040	2										2	3	no	
	TE90	0010	1										1	4	yes	
		0020	2										1	4	no	
TE00	TE10	0030	1									2	4	yes		
		0040	2									2	4	no		
	TE20	0010	1										2	4	yes	
		0020	2										2	4	no	

/Scenario/Environment/Case Description			Written	Current Status
TS010 Program launch				
TE10 DM full				
0010	Verifying links and connections enabled	yes	Ok	
TS02x MAM unitary commands				
TS020 ARCHIVE				
TE10 DM full (1 DME 1 Tape drive)				
0010	1 file type /dir/file	yes	Ok	
0020	1 file type /basedir/dir/file	yes	Ok	
0030	multiples files	yes	Ok	
0040	1 dir type /basedir/dir	yes	Ok	
0050	multiples dirs	yes		
0060	dirs excluding file(s)	yes	Ok	
0070	archiving file with space in filename/path	yes	Ko	
0080	archiving empty directory	yes	Ko	
0090	No (not enough) space available on tape (ActionWillFail?)	yes	Ok	
0100	big amount of files to archive in a row (testing socket buffer and message management)	yes	Ok	
0110	No space available in ingest zone (copy to ingest activated)	yes		
0120	verifying that it works with different QOS although it doesn't make any difference here	yes	MALFORME D	
0130	Files/dirs don't exist	yes	Ok	
TE20 DM full (2 DME 2 Tape drive)				
0010	1 file	yes	Ok	
0020	2+ files checking files splitting validity, DME choice	yes	Ok	
0030	1 dir	yes	Ko	
0040	2+ dirs	yes		
0050	1 dir QOS = 3	yes	Ko	
0060	No space available on 1 tape	yes	Ok	
0070	No space available on both tapes	yes	Ok	
0080	Testing files splitting system	yes	Ok	
TE30 DM full (1 DME 2 Tape drive)				
0010	1 file	yes	Ok	
0020	2+ files checking files splitting validity	yes		
0030	1 dir	yes	Ok	
0040	2+dir	yes		
0050	1 dir QOS = 3	yes	shouldn't work	
0060	No space available on 1 tape	yes	Ok	

0070	No space available on both tapes	yes	Ok
0080	Testing files splitting system	yes	Ok

TS021 RESTORE (quite symetrical with TS020 ARCHIVE)
TE10 DM full (1 DME)

0010	1 file type /dir/file	yes	Ok
0020	1 file type /basedir/dir/file	yes	Ok
0030	multiples files	yes	Ok
0040	1 dir type /basedir/dir	yes	Ok
0050	multiples dirs	yes	
0060	restoring file with space in filename/path	yes	
0070	File(s) to restore already exists in filesystem	yes	Ok
0080	Part of the archive to restore is already in the filesystem	yes	Ok

TE20 DM full (2 DME 2 Tape drive)

0010	2+ files	yes	Ok
0020	1 dir	yes	Ok
0030	2+ dirs	yes	
0040	File(s) to restore already exists in filesystem	yes	
0050	Part of the archive to restore is already in the filesystem	yes	
0060	some part of the archive is not present (missing 1 tape)	yes	

TE30 DM full (1 DME 2 Tape drive)

0010	2+ files	yes	Ok
0020	1 dir	yes	Ok
0030	2+dir	yes	Ok
0040	File(s) to restore already exists in filesystem	yes	
0050	Part of the archive to restore is already in the filesystem	yes	
0060	some part of the archive is not present (missing 1 tape)	yes	Ok

TS022 COPY
TE10 DM full (1 DME)

0010	1 file type /dir/file	yes	Ok
0020	1 file type /basedir/dir/file	yes	Ok
0030	multiples files	yes	
0040	1 dir type /basedir/dir	yes	Ok
0050	multiples dirs	yes	Ok
0060	dirs excluding file(s)	yes	
0070	copying file with space in filename/path	yes	Ko
0080	files/dir don't exists	yes	Ok

TS023 DEL FROM DISK

TE10 DM full			
0010	AU contains a single file	yes	Ok
0020	AU contains multiple files	yes	Ko
0030	AU doesn't exists	yes	Ok
0040	The refered files don't exist on disk	yes	Ok
TS024 DEL FROM ARCHIVE			
TE10 DM full			
0010	AU contains a single file	yes	
0020	AU contains multiple files	yes	Ok
0030	AU doesn't exists	yes	Ok
0040	Tape containing AU is not reachable	yes	MALFORME D
TS025 Others commands			
TE10 DM full			
0010	Getting metadata		Ko
0020	Defrag		
0030	Scan bus		
0040	Cancel	yes	
0050	Progress		Ok
0060	Archive with residual	yes	Ko
TE10 DM without DMSL (command sent to DMC)			
0010	Black list a tape, tape exists		
0020	Black list a tape, tape does not exist		
TS030 MAM commands sequence			
TE10 DM full			
0010	testing saturating DM with max number of command		
0020	testing simultanous commands accessing to the same data	yes	Ko
0030	testing priority system		
0040	testing tape drive choice when at least 1 drive is already in use	yes	
TS04x Generic error cases while executing a command			
TS040 Verifying error detection and notification			
TE10 DM full			
0010	SS failed executing a command,reported to DME	yes	Ok
0020	Communication with SS lost during command	yes	Ok
0030	DME failed executed a commmand, reported		

	to DMC		
0040	DMC reach timeout on waiting for DME answer		
0050	DMC never receives SF answer	yes	
0060	DMC receive malformed answer from SF		
0070	DMC receive malformed answer from DME		Ko
0080	DB became unreachabele	yes	Ko

TS041 Verifying program state after an error has occurred
TE10 DM full

0010	SS failed executing a command,reported to DME	yes	
0020	Communication with SS lost during command		
0030	DME failed executed a commmand, reported to DMC		
0040	DMC reach timeout on waiting for DME answer		Ok
0050	DMC never receives SF answer		
0060	DMC receive malformed answer from SF		
0070	DMC receive malformed answer from DME		
0080	DB became unreachabele		
0090	DME has just restarted		Ok

TS050 Verifying SF strategies independantly (using SF_test module)
TE10 Strategy 1: Archive disk to tape

0010	1 tape is available and known by a DME, tape space exists	yes	Ok
0020	No tape is known	yes	Ok
0030	1 tape is know but no dme is known that can manage it	yes	Ok
0040	Tape doesn't have enough space	yes	Ok
0050	1 tape is available but busy	yes	Ok
0060	2 tape are available but only one have space	yes	Ok
0070	2 tape are available but only one have space and no DME manage it	yes	Ok
0080	the concerned directory is not referenced in the DB	yes	Ok
0090	the concerned directory is not managed by any DME	yes	Ok
0100	SF has to choose between a MOUNTED IDLE, a IDLE and a MOUNTED READING tapes	yes	Ok
0110	SF has to choose between a MOUNTED READING and a MOUNTED WRITNG tapes	yes	Ok
0120	Only a IDLE tape can be accessed	yes	Ok

TE20 Strategy 2: Archive disk to disk

0010	directory is managed by a dme, disk space exists	yes	Ok
0020	directory is not present in DB	yes	Ok

0030	directory is mounted but no DME can manage it	yes	Ko
0040	no disk can satisfy the query	yes	Ko
0050	not enough space on satisfying disk	yes	Ok
0060	SF has to choose between a IDLE, a READING and a WRITING disks	yes	Ok
0070	SF has to choose between a READING and a WRITING disks	yes	Ok

TE30 Strategy 3: Restore disk to disk

0010	both directories are managed, disk space exists	yes	Ok
0020	both directories are managed, destination disk doesn't have enough space	yes	Ok
0030	the concerned destination directory is not managed by any DME	yes	Ko
0040	the concerned source directory is not managed by any DME	yes	Ko

TE40 Strategy 4: Restore tape to disk

0010	Tape and directory are managed by a DME, disk space exists	yes	Ok
0020	tape is not managed by a DME	yes	Ko
0030	tape is unknown by SF	yes	Ok
0040	Disk is unknown	yes	Ok
0050	not enough space on disk	yes	Ok
0060	Disk is not managed by a DME	yes	Ko

TE50 Strategy 5: Del from disk

0010	au exists on disk	yes	
0020	au exists only on tape	yes	
0030	au doesn't exists in DB	yes	Ok

TE60 Strategy 6: Format tape

0010	tape is managed, new name is not used	yes	Ok
0020	tape is unknown by SF	yes	Ok
0030	new tape already exists for another tape	yes	Ok

Test case file is the generic TS060-TExx-TCxx for all the following cases

TS060 Performances : estimating Archiving and Restoring bandwidth

TE10 1 DME 1 Tape drive

0010	Copy to ingest activated	yes	Ok
0020	Copy to ingest deactivated	same	

TE20 2 Tape Drive

0010	Copy to ingest activated (1 DME)	same	
0020	Copy to ingest deactivated (1 DME)	same	

0030	Copy to ingest activated (2 DME)	same
0040	Copy to ingest deactivated (2 DME)	same

TE30 3 Tape drive

0010	Copy to ingest activated (1 DME)	same
0020	Copy to ingest deactivated (1 DME)	same
0030	Copy to ingest activated (2 DME)	same
0040	Copy to ingest deactivated (2 DME)	same

TE40 4 Tape drive

0010	Copy to ingest activated (1 DME)	same
0020	Copy to ingest deactivated (1 DME)	same
0030	Copy to ingest activated (2 DME)	same
0040	Copy to ingest deactivated (2 DME)	same

8. CONCLUSIONS

8.1. *functional tests conclusions*

The other tests performed are some validation tests that are normally done within a classical software development cycle.

No major bug have been found and so the specification and design of the DM have not been modified.

8.2. *Performance analysis and further work*

A first remark can be done when Data Mover is configured with a single tape drive connected to a single Data Mover Engine. In this configuration, the maximum tape drive throughput achieved is about 70% of the tape drive maximum throughput specification.

An additional integration work between the OS that is used for the project (Linux) has to be done. Indeed, the maximum performance given by O'Mass can be obtained on windows without tuning the OS parameters but specific parameters have to be positioned for this OS: Data mover and Tape Drives have to be more integrated together within SIVSS architecture.

A second remark can be added when the Data mover uses several tape drives connected to a single server. In this configuration, the maximum throughput achieved by each tape drive does not exceed 60% of the maximum tape drive maximum throughput specification. This means that some other tuning parameters have to be done to achieve the desired level of performance. The SCSI chain between the tape drives and the OS also has to be tuned so that each tape drives on the chain can provide the maximum throughput.

Furthermore, a single Data Mover node cannot manage an infinite number of tape drives. Each server has a limitation. Indeed the Data Mover Engine node moves data from the distributed file system (SAN file system) to the tape drives subsystem. The Server motherboard has to manage the data flows from the HBA connected to the SAN to the SCSI card that connects the tape drives. The motherboard bandwidth is limited and several servers have to be tested to optimise the price/performance ratio. The shared file system can also limit the throughput and further test results will be displayed in D8.2 deliverable to characterize the impact of the shared file system on performance.