



PUBLIC  
DELIVERABLE  
AC-018  
SMASH

Deliverable # 15  
A0018/td/r&d/ds/p/015/b1  
August 1998

# **Report on Ongoing Standardization Efforts**



<b>Project number</b>	:	AC018
<b>Project Title</b>	:	SMASH
<b>Deliverable Type</b>	:	Public

<b>CEC Deliverable Number</b>	:	A0018/td/r&d/ds/p/015/b1
<b>Internal Project Number</b>	:	SMS-TD-834-2
<b>Contractual Delivery date:</b>		31 <sup>st</sup> August 98
<b>Actual Delivery date</b>	:	30 <sup>th</sup> August 98
<b>Title of Deliverable</b>	:	Report on Ongoing Standardization Efforts
<b>Contributing Workpackages</b>	:	WP500
<b>Nature of Deliverable</b>	:	Report
<b>Author(s)</b>	:	Ivar Miljeteig (Tandberg Data) Ronald M. Tol (Philips)

## Abstract

This report describes some of the most important aspects of the different standards relevant for Tape Devices.

The most important Tape Device technologies are described.

The relevant standards are listed together.

Also a report on the standardization work in DAVIC is included.

## Keyword List

SMASH, Multimedia, Linear Tape Drive, Storage, SCSI, ANSI, QIC, NCITS, DAT, LTO, DDS, AIT, Travan, DAVIC



---

## Contents

<b>1.</b>	<b>ABOUT THIS REPORT</b>	<b>7</b>
<b>2.</b>	<b>TAPE TECHNOLOGY AND STANDARDIZATION</b>	<b>8</b>
<b>3.</b>	<b>QIC AND MLR</b>	<b>9</b>
3.1	QIC 5010 (MLR1)	9
3.2	QIC 5210 (MLR3)	9
3.3	Other QIC standards	10
<b>4.</b>	<b>TRAVAN</b>	<b>11</b>
<b>5.</b>	<b>DLT</b>	<b>12</b>
<b>6.</b>	<b>AIT</b>	<b>14</b>
<b>7.</b>	<b>MAMMOTH</b>	<b>15</b>
<b>8.</b>	<b>LTO</b>	<b>16</b>
8.1	Ultrium: High Capacity	16
8.2	Accelis: Short access time	16
<b>9.</b>	<b>DDS</b>	<b>18</b>
<b>10.</b>	<b>TRENDS AND NEW FUNCTIONALITY</b>	<b>19</b>
10.1	Self Diagnostics - Tape Alert	19
10.2	Host Interface	19
10.3	Access Time	19
<b>11.</b>	<b>DAVIC STANDARDIZATION</b>	<b>21</b>
11.1	Example #1 - "Pause" of Broadcast Content	21
11.2	Example #2 - Dynamic Advert Insertion	21
11.3	Example #3 - Caching Data for EPGs	21
11.4	Example #4 - Caching Data for Other Applications	22
11.5	Example #5 - Caching Linear TV Applications	22
11.6	Example #6 - Caching Interactive Applications	22
11.7	Security Requirements	22

---

11.8	Management of Local Storage Devices	22
11.9	Caching of Data	22
11.10	Recording and Access to Non Real-Time Data	23
11.11	Recording and Playback of Real-Time Data	23
11.12	Peliminary conclusions	24
<b>12.</b>	<b>BACKGROUND INFORMATION ON DSM-CC</b>	<b>25</b>
<b>13.</b>	<b>NAMING OBJECTS STORED LOCALLY</b>	<b>26</b>
13.1	Naming DSM-CC UU objects	26
13.2	Naming other content	26
13.3	Metadata	26
<b>14.</b>	<b>STORAGE CAPACITY MANAGEMENT</b>	<b>27</b>
14.1	Quotas	27
14.2	Ownership/Security	27
<b>15.</b>	<b>MEDIA CAPTURE API</b>	<b>28</b>
<b>16.</b>	<b>FILTERS</b>	<b>29</b>
<b>17.</b>	<b>EXAMPLE INTEROPERABLE APPLICATION</b>	<b>30</b>
17.1	Application scenario	30
17.2	Object properties	31
17.3	Subscribing	31
17.4	Filtering	32
17.5	Provisional API proposal	33
17.6	Service provider controlled cache priorities	35

## 1. About This Report

This report gives a brief status of the QIC standardization work. QIC (Quarter Inch Committee) is an international standardization committee that make standards for a wide range of tape streamers using Quarter Inch media.

Brief descriptions of some of the competing tape standards in the market today, such as LTO, DLT, AIT and DDS are included.

The report also gives an overview over the ongoing DAVIC standardization that is relevant to SMASH. The Digital Audio-Visual Council (DAVIC) is a non-profit Association based in Geneva, Switzerland, aimed at promoting the success of

Digital audio-visual applications and services based on specifications that maximize interoperability across countries and applications/services.

Within the context of SMASH we have responded to the Call for Proposals Number 11 regarding "Local Storage in the Home". We have defined user and market requirements and discussed example applications for local storage in a DAVIC Set-Top Unit (STU). The discussion on the technical issues is still ongoing. However, we have made a proposal for a provisional API.

In this report we summarize the following documents:

- Philips response to DAVIC's Call for Proposals Number 11 "Local Storage in the Home" also know as CFP11\_044 (Twentieth DAVIC Meeting, Milan, Italy, March 9-13, 1998)
- Local storage: some technical issues (Ad-hoc group meeting on Local Storage Based Systems, Bethesda, Maryland, USA, May 13-15, 1998)
- Local storage: follow-up to CFP11\_044 (Twenty-first DAVIC Meeting, Kuala Lumpur, June 15-19, 1998)

## 2. Tape Technology and Standardization

The tendency the last years has been that most Tape Device vendors are moving away from standardization.

We typically see that companies are forming alliances around concepts (as IBM, HP and Seagate do with LTO) and that companies are trading products to be able to offer a complete range of systems from Entry Level through Mid Range to High End.

The QIC committee, once a dominant standardization organization, is now in practice divided into the MLR technology (basically Tandberg Data products based on the traditional DC9xxx or Magnus cartridge) and Travan technology (a smaller footprint cartridge with 8mm tape). Imation (a former part of 3M) is still the major media vendor for these technologies.

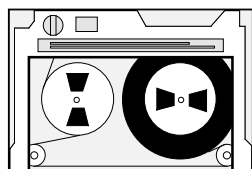
There has been very low activity in the QIC committee the last year and nothing indicates that this trend will change.

Since the market is now more influenced by alliances and solitude technologies this report will concentrate on describing the most important existing tape technologies also discuss the LTO (Linear Tape Open) concept backed by IBM, Seagate and HP.

### 3. QIC and MLR

QIC (Quarter Inch Committee) is an international standardization committee that make standards for a wide range of tape streamers using Quarter Inch media.

The MLR (Multichannel Linear Recording) concept is based on the QIC standards described below.



The recording medium has two physical attributes called *Beginning-Of-Tape* (BOT) and *End-Of-Tape* (EOT). BOT is at the end of the medium that is attached to the take-up reel. EOT is at the end of the medium that is attached to the supply reel.

#### 3.1 QIC 5010 (MLR1)

This Standard provides a format and recording standard for a streaming 0.25 inch (6.3 mm) wide, 144 data tracks, magnetic tape in a cartridge to be used for information interchange between information processing systems, communication systems, and associated equipment utilizing a standard code for information interchange, as agreed upon by the interchange parties. The Standard provides a capacity of 16 GB on a 1500ft tape (13GB on a 1200ft tape) of formatted data on MLR1.

MLR1 incorporates IBM's Advanced Lossless Data Compression (ALDC) technology to provide state-of-the-art performance and capacity features. The ALDC compression chip, which was previously only available in mainframe type tape drives, can offer an average data compression ratio of 2.6:1 across multiple data types, compared to 2.0:1 with older IDRC or DLZ compression algorithms.

#### 3.2 QIC 5210 (MLR3)

This standard is similar to QIC 5210 except it describes the format used by MLR3.

The MLR3 will be give 25GB on a 1500ft cartridge.

##### 3.2.1 Differences between MLR1 and MLR3

All aspects previously discussed are similar for MLR1 and MLR3 unless otherwise stated. The most important differences are:

	MLR1/QIC 5010	MLR3/QIC 5210
<b>Capacity</b>	13GB/16GB	25GB
<b>Transfer rate</b>	1.5MB/s	2MB/s
<b>Flux transition Density</b>	50.8kftpi	76.2kftpi

Table 2.1 Differences between MLR1 and MLR3

### 3.3 Other QIC standards

<b>QIC CRF1</b>	Common Recording Format Standard which specifies the formatting of data and the Reed-Solomon Error Correcting Code for data retrieval.
<b>QIC 139</b>	Unrecorded Magnetic Tape Cartridge for Information Interchange, 0.315 inch (8.0 mm), 75,000 fpi (2,953 ftpmm). The following sections are described in detail: General requirements, definition, tape and cartridge, physical and magnetic requirements, speed requirements, and write enable feature.
<b>QIC 134</b>	Recording Head Standard. Specifies the recording head required.
<b>QIC 121</b>	This is the host interface specification (SCSI). See also ANSI SCSI-2 and SCSI-3 standards.
<b>QIC 154</b>	Description of the data compression algorithm.



## 4. Travan

Supported by former QIC member like Seagate, Exabyte, Tecmar, Imation, Tandberg Data and others. Travan 1 through 4 are based on former QIC-standards.

Low cost alternative to MLR. Based on 8mm, small footprint cartridges.

Tape Format	Description
<b>Travan 1 GB</b>	The Travan 1 GB cartridge offers a native capacity of 400 mega-bytes and conforms to the QIC-80 format standard.
<b>Travan 2 GB</b>	The Travan 2 GB cartridge offers a native capacity of 800 megabytes of formatted usable data (1,600 megabytes compressed). <ul style="list-style-type: none"><li>• 750 ft. of high capacity 900 Oe Gamma Ferric Oxide magnetic media.</li><li>• Up to 50 Data Tracks.</li><li>• Preformatted per QIC-3010-MC.</li></ul>
<b>Travan 3 GB</b>	The Travan™ 3 GB cartridge offers a native capacity of <ul style="list-style-type: none"><li>• 1.6 gigabytes of uncompressed formatted usable data (3.2 gigabytes compressed).</li><li>• 750 ft. of high capacity 900 Oe Gamma Ferric Oxide magnetic media.</li><li>• Tested for 50,800 flux transitions per inch (FTPI).</li><li>• Identified by a unique hole pattern and red lockout switch.</li><li>• Up to 50 Data Tracks.</li><li>• Preformatted per QIC-3020-MC.</li></ul>
<b>Travan 5</b>	Travan 5GB offers 5GB of compressed capacity (2.5GB native). Data transfer rates as fast as 60MB per minute.
<b>Travan 8 GB</b>	The Travan™ 8 GB cartridge offers a native capacity of 4 gigabytes and conforms to the QIC-3095-MC format standards. Drive products capable of using the 8 GB cartridge are being planned from a number of manufacturers. The Travan™ 8 GB cartridge features include: <ul style="list-style-type: none"><li>• 8 gigabytes of compressed formatted usable data.</li><li>• 740 ft. of high capacity 900 Oe Gamma Ferric Oxide magnetic media.</li><li>• Up to 72 Data Tracks</li></ul>



## 5. DLT

DLT (Digital Linear Tape) was originally developed by Digital Equipment Corporation's systems and later adopted by Quantum, a major disk drive manufacturer.

DLT is aimed at mid-range systems, network servers, and high-end workstations.

Quantum's DLT products are 0.5-inch cartridge streaming tape storage solutions, with a 5.25-inch form factor.

The newest DLT product, the DLT 7000, features a data transfer rate of 5.0 MB per second and a native capacity of 35 GB.

Figure 1 illustrates the simple tape path design of DLT tape drives. Tape moves from the cartridge, across the read/write head, then onto the take-up reel. When loading, the tape drive adjusts the tensioning of the tape across the head, to ensure the lightest possible contact, thus reducing wear both on the head and the media.

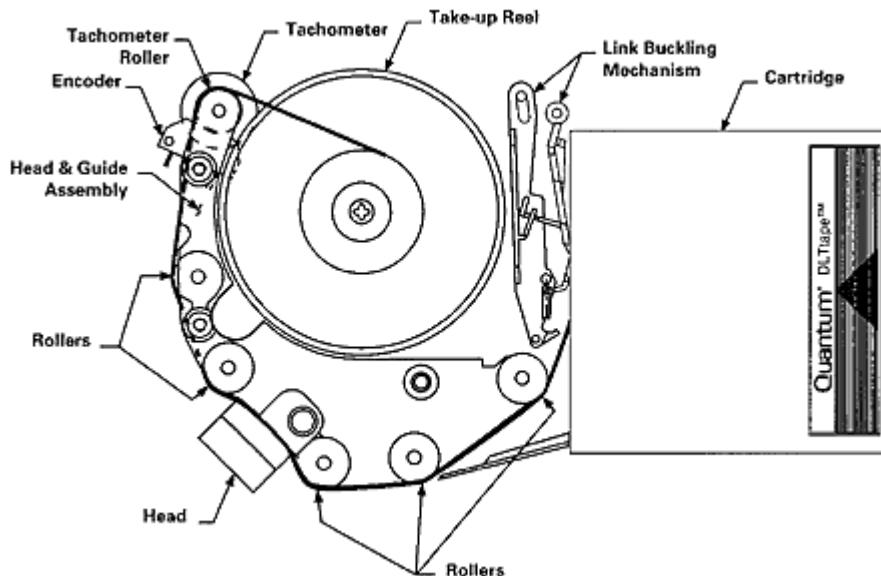


Figure 1: DLT Tape Path

Data is recorded linearly, parallel to the edges of the tape media in longitudinal tracks, allowing the drive to read multiple data channels simultaneously via the patented stationary read/write head. Combined with the tape path, and its minimal tape tension, wear on both the head and the tape media is minimized.

Since the read/write heads are non-rotating, there is no wear on the heads or tape media when the tape stops.

The DLT7000 tape drive implements a **four**-channel read/write system: four channels of data can be read or written simultaneously, resulting in a data transfer rate of 5.0 MB per second.

DLT technology uses Lempel-Ziv compression methodology.

DLT technology incorporates high-grade metal particle (MP) tape in a half-inch format.

DLT cartridges measure approximately four inches by four inches are one inch thick. The cartridge itself contains only a supply reel of tape; the take-up reel is within the tape drive.

## 6. AIT

AIT (Advanced Intelligent Tape) was introduced by Sony.

### **25/50GB Capacity**

AIT was developed by Sony. The SDX drives and media, aided by the Advanced Metal Evaporated (AME) tape formulation, achieve a capacity of 25 GB (native), 50 GB (compressed)\* on a single AIT data cartridge.

### **6.0 MB/s Data Transfer Rate**

AIT incorporates Fast/Wide SCSI technology with a 68-pin single-ended interface to maximize data transfer rates up to 3.0 MB/s (native), 6.0 MB/s (compressed)\* and burst rates of 20 MB/s to and from the host system.

### **ALDC Data Compression**

AIT incorporates IBM's Advanced Lossless Data Compression (ALDC) technology (as used in Tandberg Data's MLR).

### **Data Management with MIC**

When designing AIT, Sony recognized the need to improve the overall drive performance and incorporated a revolutionary 16 Kbit EEPROM flash memory chip within the data cartridge. This is the first time that such a feature, which Sony named Memory-in-Cassette (MIC), has been utilized in a computer tape peripheral. MIC is used to store the tape's data log, history and other user-definable information. The benefits of MIC include the following:

1. Faster access to data
2. Predictive diagnosis of media degradation
3. Faster and more reliable access to volume serial information
4. Greater data integrity through a fault-tolerant system log
5. Enhanced media security with MIC-stored decryption codes

## **7. MAMMOTH**

### **Exabyte Mammoth (Exabyte 8900)**

With 40-gigabyte capacity and 360-megabyte-per-minute data transfer rates (with data compression), Exabyte's Mammoth records at 21.6 gigabytes per hour and can search more than 20 gigabytes in 72 seconds. Mammoth features an industry-standard 5.25-inch half-high form factor.

### **Specifications**

#### **Capacity**

40 GB, compressed  
20 GB, uncompressed

#### **Performance**

Transfer rate, sustained:  
6 MB per second, compressed  
3 MB per second, uncompressed  
Transfer rate, burst:  
7 MB per second, asynchronous  
20 MB per second, synchronous

#### **Recording Method**

8mm helical scan

## 8. LTO

HP, IBM and Seagate are working together with the LTO technology. So far LTO is not a product but rather a concept based on two entirely new cartridges, Accelis and Ultrium. The reason for offering the technology on two cartridges with very different characteristics is to be attractive in a wide range of applications.

### 8.1 Ultrium: High Capacity

The Ultrium tape format is the implementation of LTO technology optimized for high capacity and performance with outstanding reliability, in either a stand-alone or an automated environment. The Ultrium tape format uses a single reel cartridge to maximize capacity. It is ideally suited for backup, restore, and archive applications.

The first generation of this technology allows for storage of up to 100GB of data (>200GB compressed) on a single cartridge at data rates as high as 20MB/sec. (>40MB/sec. compressed). The performance is made possible through the use of linear recording techniques that employ either a four or eight channel head and closed loop servo technologies. The first generation cartridge will contain 600 meters of half-inch tape to achieve its 100GB native capacity. The new cartridge will also have a cartridge memory chip that is part of the LTO interchange specification.

Ultrium partitions the half-inch tape into 384 data tracks, evenly divided into four data bands. The number of tracks in a data band is the same whether the data is written by a four- or eight-element head. If an eight-channel head is used, the 96-track data band results from six head positions dictated by the servo. The four-channel implementation requires extra servo readers on the head to generate 12 head positions from the six predefined positions.

Once a data band is filled, the head is aligned on the next prescribed pair of servo bands and begins to write data in the same manner. Special offsets in the servo bands ensure that the head is on the correct data band for reading or writing.

The Ultrium single-reel cartridge design uses a take-up reel that is located inside the drive.

### 8.2 Accelis: Short access time

The Accelis tape format is the implementation of LTO technology optimized for fast access to data.

It uses a two-reel cartridge that loads at the middle of the tape to minimize access time. The Accelis tape format is targeted at automated environments and can enable a wide range of "on-line" data inquiry and retrieval applications.

The capacity reduction is the result of making the tape narrower (hence fewer tracks) and by putting two reels in the cartridge instead of one. This results in

216 meters of 8mm wide tape, which contains 25GB of data (>50GB compressed). The dual reel cartridge implementation provides a number of advantages.

Accelis records the LTO format in two data bands instead of four for Ultrium and each data band contains 128 tracks (256 tracks total). Like Ultrium, the data bands can be recorded with eight-channel or four-channel heads and eight servo positions are defined in the servo band to support the recording.

---

## 9. DDS

DDS Manufacturers Group are working with the DDS (Digital Data Storage) DAT technology. HP and Exabyte have been the main players in this technology.

DDS and Travan have lower cost and performance compared to the other technologies discussed.

DDS-1	Capacity of 4 GB and transfer rate of 366KB/s (compression 2:1)
DDS-2	Capacity of 8 GB and transfer rate of 1 MB/s (compression 2:1)
DDS-3	Capacity of 24 GB and transfer rate of 2 MB/s (compression 2:1)
DDS-4	Planned 1999 40GB capacity on a 150-meter DDS-4 tape (compression 2:1) Transfer rate will be in the range of 1 to 3MB/s (native).

## 10. Trends and new functionality

### 10.1 Self Diagnostics - Tape Alert

Tape Alert is an emerging standard for communicating the tape Drive status to the host system. This functionality may introduce more intelligence when it comes to predict what kind of problems will occur in the future from indicators like soft errors, servo problems, time since last cleaning etc. TapeAlert was initially developed by Hewlett-Packard as a proposed new standard for tape drive management. Hewlett-Packard have worked with the industry to form the TapeAlert Working Group, where representatives from all major tape drive and backup software companies (Tandberg Data included) meet regularly to discuss implementation across the industry and also agree on enhancements to the TapeAlert specification to support additional features.

### 10.2 Host Interface

#### 10.2.1 Parallel Interface

NCITS is the standardization committee responsible for SCSI standards. LVDS (Fast 40 or Ultra 2) will be a market standard the next couple of years. Ultra 3 is the successor but is so far not very well defined.

#### 10.2.2 Serial interfaces

For HighEnd systems Fiber Channel is growing in popularity also among Tape Device manufacturers. For LowEnd and Multimedia the IEEE1394 will be more common. In applications like the SMASH Combo Box an IEEE1394 interface would be optimal.

### 10.3 Access Time

Time to Data, Load Time and other measures for waiting time get more and more focus, and in the future we will probably see development in this area. Experience from the SMASH project is that this is a very important aspect.

#### 10.3.1 MIC - Memory In Cartridge

AIT's solution is to include a memory chip on the cartridge. The memory chip may contain the tape position, file directory etc. The concept of using a memory chip on the cartridge is later adopted by other technologies.

#### 10.3.2 Dedicated Cartridge

LTO offers a particular cartridge that trades capacity for access time. Shorter length means shorter positioning time. Two reels in the cartridge means shorter time to load a new cartridge.

#### 10.3.3 Center Load

Using the tape center position as the parking place and to store directory information are ways to reduce Load Time. LTO and other technologies are using this principle.

#### 10.3.4 Partitioned tape

In the SMASH project we have discussed the possibility of dividing the tape into several data segments to reduce positioning time. This is something we may look further into in the future.

## **11. DAVIC Standardization**

Below we describe example applications for local storage in a DAVIC STU. A DAVIC specification for in-home storage should be able to support these example applications and hence these applications should be considered as user and market requirements (i.e., UM3 from CFP11).

We provide a short technical analysis of what may need to be standardised by DAVIC in order to support the example applications. We also propose at a very simple level some basic approaches for technical solutions to enable DAVIC systems to implement these example applications (i.e. INR9 from CFP11).

We have further worked out an interoperable application, the electronic newspaper example, in considerable detail. Using this example we have come to a description of a provisional API. Please note that the work presented needs completion, before it is suitable for DAVIC specification.

### **11.1 Example #1 - “Pause” of Broadcast Content**

A local storage device could be used to provide some form of “pause” function for broadcast content where the receiver cannot request the transmitter to pause. When pausing of content is required, the broadcast content could start being written to the local storage device. When pausing of content is no longer required, the broadcast content previously written to the local storage device could be played back and presented to the end user. Obviously the capacity of the storage device sets an upper limit on the length of time for which the end user can apparently pause the broadcast content. Some examples of when this feature could be used include handling incoming phone calls, visits to mens/ladies rooms and to provide a personal action replay feature.

### **11.2 Example #2 - Dynamic Advert Insertion**

A local storage device could be used to hold adverts for dynamic insertion into TV content broadcast later. This would allow different adverts to be inserted for different users, for instance depending on the user preferences, user location or user maturity rating. It could also allow adverts to use content, which would require higher bandwidth than is available at a particular time. A number of interesting business models should be possible here including ones based on sale or rental of the space on the local storage device.

### **11.3 Example #3 - Caching Data for EPGs**

A local storage device could be used to cache data for an inter-operable EPG. The obvious example of such data to be cached would be SI data in order to improve the response time of the EPG when being started. Other data for use in an EPG could also be cached such as still images or even short previews of content such as movies. Caching of data is particularly important for EPGs since in order for end users to use an EPG to navigate between services, the start up time for such an EPG must be minimised.

#### **11.4 Example #4 - Caching Data for Other Applications**

A local storage device could be used to cache data for other applications apart from EPGs. One example of this could be multimedia data for those sections of an electronic newspaper, which match end user preferences held in the STU. The recording and playback of this data could well be done at very different times in different applications.

#### **11.5 Example #5 - Caching Linear TV Applications**

A local storage device could be used to cache linear TV content. Some suitable example content could include short TV news bulletins or weather forecasts. As with the previous example, the recording and playback of this data could well be done at very different times in different applications. Users should also be able to watch from the beginning TV programs which have not yet finished, i.e. simultaneous recording and playback from different points in a program should be possible as in the first example above.

#### **11.6 Example #6 - Caching Interactive Applications**

A local storage device could be used to cache interactive applications themselves. Some example applications which it might be relevant to cache on such a device could include an inter-operable EPG itself, home banking applications, home shopping applications or an electronic newspaper possibly using previously cached data.

#### **11.7 Security Requirements**

The most important security requirement for this local storage concerns how the space on a local storage device is shared between applications, application providers, service providers and network operators that may be mutually hostile. Protection of data written by a competitor is critical to enable many useful business models. It should also be possible to integrate control of access to stored data into conditional access systems.

#### **11.8 Management of Local Storage Devices**

Clearly management of usage of local storage is a major issue. This should be as automatic as possible. When local storage devices become full, any need for end users to personally delete content should be minimised. The DAVIC specifications should include support for intelligent management of local storage devices while retaining options for manufacturer added value. Where end users are required to personally make decisions, there should be provision for applications to provide supporting information to make the decision making process as user friendly as possible.

#### **11.9 Caching of Data**

The simplest method for using local storage in the home is for that use to happen automatically. Automatic caching of data requires no extra specifications by DAVIC and should improve the responsiveness of existing DAVIC applications without any need to modify those applications. Automatic

---

caching of data is most applicable to those types of data whose size is small relative to the size of the local storage device and hence where the entire data set can be stored without any need for priorities to be specified. The most obvious type of data which could be automatically cached would be service information.

Where the size of a data set is too large for the entire data set to be cached on a local storage device, there needs to be some means to specify priorities to the caching mechanism. There are two ways in which such priorities could be supplied to the STU; one option is for these priorities to be supplied as part of the data set itself. A second option would be for the priorities to be contained in an interactive application. The second option is more flexible since it allows different STUs to use different priorities depending on local factors such as location or user preferences. We believe both options should be supported. The most obvious type of data, which this model applies to, would be data broadcast in a DSM-CC object carousel. One obvious way to allow for applications to set priorities would be to extend the current `org.davic.net.dsmcc.uu` package.

For all caching approaches, the existing DAVIC defined mechanisms for accessing the data should apply transparently regardless of whether the data is cached or not. No special data access facilities should be defined.

### **11.10 Recording and Access to Non Real-Time Data**

Several of the example applications require a facility for an interactive application in the STU to initiate storage of some non real-time data. This facility is required for example #2 (dynamic advert insertion) and example #4 (other data) since in these examples, only an interactive application in the STU possesses the information needed to decide which non-real time data should be recorded.

For access to such non real-time data, we propose that the existing mechanisms defined in DAVIC should be used as transparently as possible. This should be done by extending the name space specified in DAVIC part 9, section 9.3.3.1. This mechanism would allow access both from MHEG-5 and from Java since both use this name space mapping.

Recording non real-time data should be done at a high level. One example of this could be defining a set of primitives to copy objects from the DSM-CC object carousel to the local storage device and to manage and structure those objects after copying. Obviously these primitives should report any violations of the security requirements.

### **11.11 Recording and Playback of Real-Time Data**

Several of the example applications require a facility to record and playback real-time media. Example #1, "pause" of broadcast content requires to be able to both record and playback at the same time. There are several slightly different options within this example which need to be addressed, "pause" only starts recording when the user selects the feature whereas the personal action replay feature must start recording some time before the feature can be used.

Examples #2, #3 and #5 require to be able to record real time content and then play back that content later. There are several types of real time content which need to be addressed here, these range from fragments of real time media (e.g. short clips from films in example #3, adverts in example #2), through linear DAVIC services (e.g. weather reports in example #5) to complete DAVIC services including an interactive component. Obviously integration with a conditional access system will be particularly required for this type of data, possibly for both recording and playback.

As with the playback of non real-time data, playback of real-time data should be done using the existing mechanisms defined by DAVIC as transparently as possible.

### **11.12 Preliminary conclusions**

We believe the example applications listed above form a good set of target applications, which a DAVIC in-home storage specification should support. In order to support these applications, we suggest the following technical issues need to be addressed by a DAVIC solution:

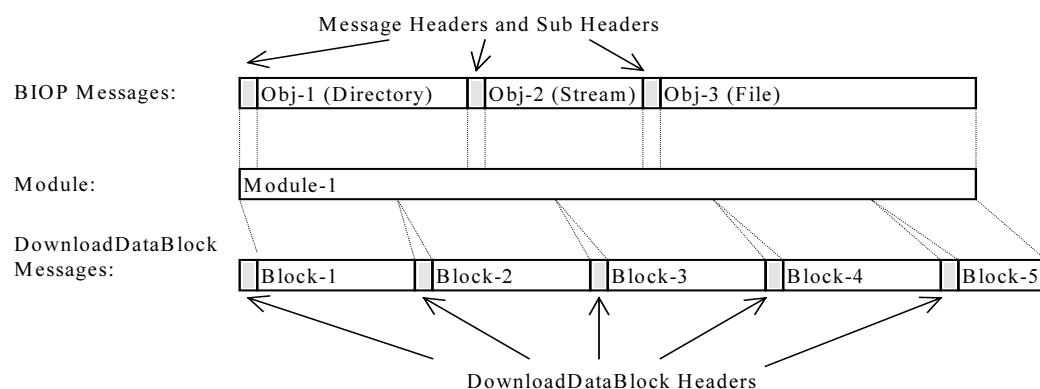
- Signaling standards to support service provider controlled cache priorities.
- DSM-CC UU API extensions to enable application provided cache priorities both for content data and for the application program itself.
- A media capture API to allow both streaming media and complete services to be recorded on a local storage device.
- Extensions to the name space for addressing content to support addressing content on a local storage device.
- Extra semantics for current media playback APIs (MHEG-5, JMF) to specify how to address content on local storage.
- Security or reference decoder model extensions to address use of local storage capacity by conflicting and/or hostile applications.

In the next few sections we elaborate on some of those issues. We first provide some background information on DSM-CC in Section 12. Then we describe in more depth some of the technical issues that we think should be addressed. We end the discussion with the example of an interoperable application, which will lead us to the definition of a provisional API.

## 12. Background information on DSM-CC

DSM-CC offers, among other things, a mechanism to broadcast objects (like files and directories) in a carousel-like fashion. Each instance of an object carousel represents a service domain. Each service domain has one special object, the ServiceGateway. This ServiceGateway object is the root of the object tree contained in the service domain. DSM-CC also allows that a service domain is represented as a set of objects that must be accessed through the back channel.

For the broadcast case (the object carousel), DSM-CC specifies three layers. At the top layer (called the object carousel layer), the DSM-CC User-to-User objects (like files and directories) are visible. These objects are transported in so-called modules. These modules represent the middle layer and this layer is called the data carousel layer. At this layer, the modules are just a container for data. The User-to-User objects are not interpreted here; this is done at the top layer. The modules are broadcast as a sequence of DownloadDataBlocks, which are MPEG-2 private sections with added semantics. These DownloadDataBlocks form the lowest layer.



**Figure 1:** DSM-CC object carousel: layering

DSM-CC allows that a directory object in one service domain has a child object in another service domain. To the application, however, it then looks like that that child object is part of the same object tree (same namespace).

DAVIC has specified how to refer to DSM-CC UU objects using `java.io`: it is the same as how MHEG refers to those objects. This means that the addressing is always relative to the current ServiceGateway. The DAVIC notation is as follows:

1	<code>DSM://apps/otherAppl</code>	Path is relative to current ServiceGateway
2	<code>DSM:/scenes/myScene</code>	Path is relative to the directory where the MHEG Application object was found
3	<code>/apps/otherAppl</code>	Shorthand notation for case 1.
4	<code>~/scenes/myScene</code>	Shorthand notation for case 2.

## 13. Naming objects stored locally

The set-top box will be able to cache application code/data and linear audio-visual content on local storage. This section describes some possible naming schemes that an interoperable application can use to refer to or access this cached information.

### 13.1 Naming DSM-CC UU objects

It is possible to use the same names for UU objects that are stored/cached locally as for non-cached UU objects. The STU simply first checks the local storage and only if it can't find the object there, it tries to retrieve it from the carousel. So, for UU objects, the problem seems solved: no extra naming scheme is needed.

### 13.2 Naming other content

For non-UU objects, like stored video streams, a new naming mechanism is needed. This could for instance be something like:

```
file://[<pathName>/]<fileName>
```

(or more general, the URL definition for the `file://` protocol). The media capture API can use such names, `java.io` can use it, and JMF can use it (JMF accepts URLs).

Note that according to the Java API specification, the Java class `java.io.File` does not normally take a general URL to construct a File object, but is also does not forbid it. `java.io.File` generally uses a path name plus a file name to refer to a file (e.g., `/dir1/dir2/file1`).

### 13.3 Metadata

For the Java Media Framework, some meta-data is needed next to a recorded stream to indicate the content type of the stream. It might be possible to implement this meta-data storing/querying 'under the surface', i.e. not visible through extensions to neither JMF or the media capture API. Or it can (automatically) be stored in a file name with a special name, like for instance the same file name as the stream but with an extra `.meta` suffix.

## 14. Storage capacity management

The capacity of local storage is (always) limited and therefore management of the usage of local storage is important. This management should be as automatic as possible. This section describes some of the issues of local storage capacity management.

### 14.1 Quotas

Quotas can be useful for the management of the storage capacity. The issues here are at least:

*To which kind of entity should quotas be allocated?*

It is possible to allocate quotas based on service provider, service ('channel'), application and possibly others.

*Who is able to set the quotas?*

Here options include the STU manufacturer (e.g., in a business model where the hard disk in the STU is sponsored by a specific service provider), the end-user (e.g., the STU can automatically allocate quotas based on the end-user's selection of a set of application/services), and the service provider (e.g. by sending packages to the STU that configure the quotas). Should the quotas be static (i.e., set only once) or dynamic? Dynamic quotas are more interesting, but also more difficult.

### 14.2 Ownership/Security

An interoperable application should not be allowed to access data written by a competitor. Therefore, an interoperable application should authenticate itself. Using digital signatures for this authentication is a logical solution. When the STU knows the origin of an application (e.g., the service provider of the application), it can enforce access to only that part of the local storage that has been allocated to the service provider of the application.

It should thus be impossible for applications from different service providers to access each other's cached content. This can be done by giving each service provider its own 'home directory' and making all content references to local storage relative to that home directory. The inability to reference content of other service providers then gives the desired access control. When using the `file://` naming scheme to refer to local storage, the semantics of a relative path name could be such that the path is relative to the root directory of the service provider.

## 15. Media capture API

The Java Media Framework and the MHEG Stream class look the most obvious ways to play back recorded material. An interoperable application can use the naming conventions described earlier for this. For the recording of continuous media, another API is required. We call this API the media capture API.

Roughly, the API would require the following methods:

<i>SetSource(URL)</i>	To indicate what should be recorded. The URL could for instance refer to a service or service event.
<i>SetDestination(URL)</i>	To indicate where the data should be stored. The URL could for instance represent a filename on local storage.
<i>Start()</i>	To activate the settings.
<i>Stop()</i>	To stop recording.
<i>Pause()</i>	To pause recording.
<i>Resume()</i>	And to resume it again.

## **16. Filters**

The media capture API described above allows an interoperable application to record a single item. It is not directly suitable for the caching of linear TV programs, like caching the latest news bulletins or the latest weather forecasts. For this kind of caching, a filtering mechanism seems appropriate. The interoperable application can define a filter that specifies what should be recorded when. This specification can be indirect (e.g., on channel x, cache the latest service event whose content type is 'news bulletin'). Currently, we are investigating possible filter specifications and how to handle conflicting filters (e.g., two filters want to record things on two different transport streams at the same time while there is only a single network interface).

## 17. Example interoperable application

We have worked out an interoperable application, the electronic newspaper example, in considerable detail. We do not give a single solution for this application, but we present options for such solutions. These solutions may also be applicable to other kinds of application, but we have not looked at that here.

Please note that the work presented needs completion, before it is suitable for DAVIC specification.

In Section 0 a sample application scenario of the electronic newspaper is given. In the Sections 0, 0, and 0 some issues and possible solutions are presented. Section 0 describes a provisional API.

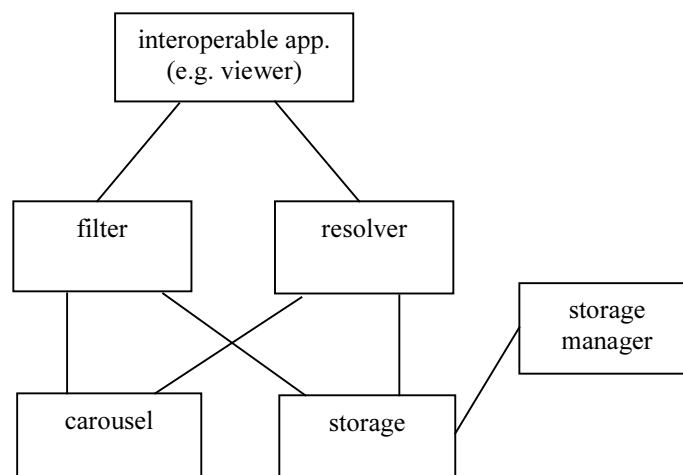
At the end of this document, in Section 0, we also describe a mechanism through which a service provider can indicate what it would like to have cached.

### 17.1 Application scenario

An electronic newspaper can consist of many objects in the carousel. These objects are to be presented to the user in some manner; e.g., a *viewer* application presents a front page representing the table of contents with links to other elements of the newspaper. Typical elements are headlines, current affairs in politics, sports, the weather, etc. These elements are typically only relevant for a limited time and they can be replaced by more recent ones.

A customer can subscribe to an electronic newspaper server of some service provider. This will result in the installation of a so-called *filter*, which will be responsible to cache objects related to the service on local storage. Also a certain amount of local storage is reserved for the service. A *storage manager* will be responsible for the handling and the administration of storage.

Since we do not want the viewer to be aware of local storage, i.e., we pursue transparent access, we also introduce a *resolver* to take care of resolving the correct name of a requested object.



---

## 17.2 Object properties

To facilitate the electronic newspaper service, objects in the carousel need to have certain properties, like

- type, describing the kind of object (e.g., sports, financial information, headlines ...)
- expiration date, until when the information in this object is valid
- date last changed, when the information was renewed
- originator, where this object comes from
- priority, how important the object is

Properties can either be assigned to individual objects or to sets of objects. When properties are assigned to individual objects, we have at least the following options:

1. Listing the object properties in the directory object listing the object. DSM-CC specifies the objectInfo field (in the Binding structure) which suits this purpose.
2. Listing the object properties in the header that precedes the actual object. DSM-CC specifies the objectInfo field (in the MessageSubHeader structure) which suits this purpose.
3. Centralising the object property information. This means providing a list of (object, object properties). This list could be broadcast as an object in the carousel.
4. Listing the properties in the filter.

However, assigning priorities to individual objects has serious disadvantages. First, it requires non-negligible bandwidth. Second, the set-top box must do considerable processing to examine all the properties of the individual objects. It is therefore much better to assign properties to sets of objects. Here we have at least two options:

1. Properties are assigned based on the hierarchical structure of the object carousel. If an object has a property, it implies that all its children have the same property.
2. Properties are assigned to the modules that carry the objects in the carousel. That is, properties are assigned at the data carousel layer instead of at the object carousel layer. If a property is assigned to a module, it implies that all the objects in the module have that property.

The second option enables easy pre-fetching, as will be explained in Section 0 on filtering.

## 17.3 Subscribing

This section describes in more detail how an end-user could subscribe to an electronic newspaper.

The electronic newspaper application can be an MHEG-5 (or MHEG-6) application, associated with a particular service. When the end-user selects that service, the set-top box retrieves the application from the broadcast stream and starts up the electronic newspaper application. The application asks the end-user whether he wants to subscribe and to which parts of the

newspaper. Based on the end-users' answers, the application constructs a filter and 'installs' it. This completes the subscription phase.

Note that the same MHEG application can also contain the functionality to view the newspaper.

## 17.4 Filtering

An electronic newspaper application may wish to cache (pre-fetch) two types of data. The first type of data is a set of objects contained in an object carousel. The second type of data is (MPEG-2) video streams. This section focuses on the first type of data. Currently, we are still investigating the second type.

Two possible ways for a filter to specify which objects in a carousel should be cached are:

1. Specify one or more sub-trees of the carousel. All objects in those sub-trees should be cached. For example, a filter could specify '/apps/news/sports'.
2. Specify one or more tags that should match the tags assigned to the modules of the data carousel layer. All objects in those modules should then be cached. For example, a filter could specify 'content-type=sports OR content-type=headlines'.

The first option has a number of disadvantages. First, it requires that the set-top box must parse the module that contains an object in which the set-top box is interested. It must also parse the modules that contain an intermediate directory object in the path to the object. Also, all the directory objects in the specified sub-tree must be parsed, so the set-top box knows which other objects are part of the sub-tree. In short, this requires a lot of processing. Second, the set-top box must decompress modules if they are compressed. However, the set-top box ideally wants to store the objects in local storage in a compressed form. This means that first a module is decompressed and that subsequently the individual objects are compressed. This is waste of processing power.

The one advantage of this first option, however, is that it can be used for objects that must be accessed through the back channel, whereas the second option cannot be used for those objects. The reason for this is that the module concept is only used for object carousels, and not on the back channel, where DSM-CC uses an RPC-like mechanism.

The second option does not have the disadvantages of the first option: the set-top box no longer needs to parse any module when pre-fetching and it can store modules in a compressed form if they were broadcast in compressed form, thus saving storage space.

Another advantage is that it is easy to find out which modules should be pre-fetched. The service provider periodically broadcasts a message listing the modules that are part of an object carousel, the 'DownloadInfoIndication'

message. This message contains for each module a 'ModuleInfo' structure. This structure has a field 'userInfo'. The DSM-CC standard does not specify the use of this field; it is meant as an extension mechanism. This 'userInfo' field can easily contain the module tags. Thus, the set-top box retrieves the 'DownloadInfoIndication' message(s), and compares for each module the listed tags (in the 'userInfo' field) with the tags in the filter. If a tag matches, it pre-fetches the module.

Finally, it is easy to see whether a pre-fetched module needs updating. DSM-CC assigns to each module a version number (listed in the 'DownloadInfoIndication' message). Thus, to check whether a pre-fetched module is still up-to-date, the set-top box retrieves again the 'DownloadInfoIndication' message and compares the version numbers.

## 17.5 Provisional API proposal

Note: This API needs completion before it is suitable for DAVIC specification.

### 17.5.1 Requirements

1. An interoperable application shall be able to indicate which DSM-CC UU objects should be cached:
  - by specifying a set of module tags. The objects contained in a module that matches (at least one of) the specified tags should be cached.
  - by specifying a sub-tree of objects.
2. An interoperable application shall be able to specify whether a cache request applies to the lifetime of the current execution of the application or whether it also applies afterwards.
3. An interoperable application shall be able to indicate that it is no longer interested in the caching of certain DSM-CC UU objects.
4. An interoperable application shall be able to query which cache requests it has set.
5. An interoperable application shall be able to query whether a specific DSM-CC UU object is in the cache or not.
6. An interoperable application shall not be able to inspect, or modify cache requests of other applications.

An open issue is how long the filters (cache requests) stay active. For example, what if the end-user runs the electronic news program once, subscribes, and then 'never' runs it again? It could be possible that the storage system tracks which data has been accessed and which not and when the data was stored. Then data that has been stored for a long time and that was never accessed, could be a candidate for being removed, and the filter that captured it, could be a candidate for being removed also.

Another open issue is whether there are limits on how many filters are allowed to be active at one time.

### 17.5.2 API specification

The logical place to add the functionality that fulfils the requirements above is the DSM-CC UU API. Next to this API, it may be desirable to add a small set

---

of MHEG resident programs for similar functionality, especially for MHEG-5 programs.

As each DSM-CC service domain has its own Service Gateway, representing the root of the object tree, the ServiceGateway class of the DSM-CC UU API may be a good place to add caching related functionality for that service domain. Methods similar to the following could be added to this class:

```
/** This method starts the pre-fetching of modules whose tags
 * match at least one of the specified tags (subject to
 * storage space limitations). The modules are part of
 * the service domain of this ServiceGateway.
 * If 'temporarily' is true, the caching request applies
 * only to the lifetime of the current execution of
 * the application, otherwise it also applies afterwards. */
public void SetModuleTagsForCaching(String[] tags,
    boolean temporarily);
```

```
/** Stops the pre-fetching of modules based on the specified
 * tags (for this service domain). */
public void RemoveModuleTagsForCaching(String[] tags);
```

```
/** Returns all the module tags that are used to pre-fetch
 * modules of this service gateway.
 * If 'temporarily' is true, then the returned values
 * represent cache requests with the lifetime of the
 * current application, otherwise the returned values
 * represent cache requests that are not limited by
 * the lifetime of the application.
 **/
public String[] GetModuleTagsForCaching(boolean temporarily);
```

```
/** This method starts pre-fetching the sub-tree with the
 * root specified by the pathName (subject to storage space
 * limitations). In principle, it will cross possible
 * service boundaries,
 * if needed. However, if a 'newly needed' service domain
 * needs to be accessed through the back channel, then
 * that part of the sub-tree that is part of that
 * service domain will not be pre-fetched.
 * If 'temporarily' is true, the caching request applies
 * only to the lifetime of the current execution of
 * the application, otherwise it also applies afterwards. */
public void SetSubtreeForCaching(NameComponent[] pathName,
    boolean temporarily);
```

```
/** Stops the pre-fetching of the sub-tree with the root
 * specified by the pathName. */
public void RemoveSubtreeForCaching(NameComponent[] pathName);
```

```
/** Returns all the names of the sub-tree roots that are
 * used to pre-fetch objects.
 * If 'temporarily' is true, then the returned values
 * represent cache requests with the lifetime of the
 * current application, otherwise the returned values
 * represent cache requests that are not limited by
 * the lifetime of the application.
 **/
```

---

```
public NameComponent[][] getSubtreesForCaching(  
    boolean temporarily);
```

A method similar to the following could be added to the Directory class:

```
/** Returns whether the object specified by the pathName  
 * that is relative to this directory object is currently  
 * in the cache. */  
public boolean isCached(NameComponent[] pathname);
```

Note that the appropriate exceptions, e.g. for indicating that an application is not allowed to request caching beyond the lifetime of its current execution, are to be added.

## 17.6 Service provider controlled cache priorities

The above described how an interoperable application can take the initiative to cache data. However, it may also be desirable that the service provider indicates what it would like to have cached. Suppose a service provider has been allocated a certain amount of local storage. Since storage capacity is limited, the service provider may indicate the relative importance of its data by giving priorities to its data elements.

As an example, a service provider may broadcast multiple applications, like an EPG, play-along game and a banking application. It considers the EPG as its most important application in the sense that the EPG should give the best response times. Thus, it wants that this application to be in the cache whenever possible. The service provider can signal this by giving the EPG application the highest priority.

An obvious way to provide the cache priorities is to include them in the Service Information (SI) tables/descriptors that describe cacheable data. For example, an Event Information Table could indicate the cache priority of each individual event, or the descriptor announcing an interactive application could indicate with which priority the application should be cached.

There is, however, a problem with providing cache priorities this way. If a STU wants to be sensitive to the cache priorities determined by the service provider and thus cache only those data and linear TV content that have a high enough priority, then the STU must have a complete view of all cacheable data and linear TV content and their associated cache priorities. Without such a complete view, the STU may miss cacheable data that it would have cached if the STU had known about it. The problem is that the STU must scan all tables/descriptors that can contain a cache priority, in all TSs, in order to get this complete view. This is a tedious way to obtain the cache priorities, because it requires sequential tuning to all TSs and the parsing of all the potentially interesting tables/descriptors.

A way to circumvent the problem described above is to provide a SI table or descriptor that lists all cacheable data to which the service provider has assigned a cache priority. The scope of such a table/descriptor can be TS or a

network<sup>1</sup>. An item in the list is a < pointer to cacheable data, cache priority > tuple. Optionally, a list item can contain a field giving information on the required storage.

The pointer to the cacheable data can for instance be the SI identification of a service event (“TV program”) or an identification of where to find an interactive application or other kind of data. The information on the required storage gives an indication on how much storage the STU needs, to cache all the cacheable data that is pointed to by the first element in the tuple.

By using the TS-wide or network-wide table/descriptor, the STU can quickly get a complete view of all the cacheable data. For each TS/network, it simply parses a single SI table/descriptor, and it combines the results.

The required bandwidth for broadcasting the table/descriptor can be low; i.e. the table only needs to be broadcast occasionally. The STU does not need instant access to the table/descriptor, because an access latency does not affect the performance of a direct interaction with the end-user.

---

<sup>1</sup> For instance, a new descriptor could be defined that is to be included in the TSDT (Transport Stream Description Table).