



PUBLIC  
DELIVERABLE  
AC-312  
STORit

Deliverable #5  
AC312/bbc/r&d/ds/p/005/b1  
February 1999

## Content Description Interface for Home Storage Applications.



<b>Project Number</b>	:	AC 312
<b>Project Title</b>	:	STORit
<b>Deliverable Type</b>	:	Public

<b>CEC Deliverable Number</b>	:	AC312/bbc/r&d/ds/p/005/b1
<b>Internal Project Number</b>	:	STR-P-908-2
<b>Contractual Deliverable Date</b>	:	28 <sup>th</sup> February 1999
<b>Actual Date of Deliverable</b>	:	24 <sup>th</sup> February 1999
<b>Title of Deliverable</b>	:	Content Description Interface for Home Storage Applications
<b>Contributing Workpackages</b>	:	WP 310/320/330
<b>Nature of Deliverable</b>	:	Report
<b>Author(s)</b>	:	G.Robertson (BBC R&D) Ir. P.G. Meuleman (Philips Research) m.sc. M.P. Ceccarelli (Philips Research) Dr. ir. E.A. Montie (philips Research) Dr.Ing. D. Melgignano (Philips Monza) Ir. G.C. van den Eijkel (TUD)

## Abstract

The objective for the deliverable is to 'set requirements and propose a suitable database structure for standardisation of content descriptors in the area of home storage applications'

The report will;

1. Analyse the objectives of the content 'indexing' system
2. Analyse the user's requirements for meta-information (attractors & descriptors)
3. Define an interface for the manipulation of meta-information
4. Describe framework for management of meta-information throughout the delivery chain
5. Consider user and system requirements for content capture based on 'user interest profiles'
6. Report on use of existing standards and ways in which STORit can provide input to current standardisation initiatives

## Keyword list

STORit, TV Anytime, video, storage, meta data, DVB, Internet, interoperability, user interface, filter, user profile, navigation



## CONTENTS

<b>1</b>	<b>INTRODUCTION</b> .....	<b>7</b>
1.1	Purpose & Scope of Deliverable #5.....	7
1.2	Structure of Deliverable #5 .....	7
<b>2</b>	<b>DESCRIPTION OF A STORIT SYSTEM</b> .....	<b>8</b>
2.1	The STORit applications .....	8
2.1.1	DVB / Internet application .....	8
2.1.2	Personalised Remote Learning.....	8
2.2	System outline .....	8
<b>3</b>	<b>REQUIREMENTS</b> .....	<b>10</b>
3.1	Requirements for automatic recording component .....	10
3.1.1	Basic technology outline .....	10
3.1.2	User requirements for profile based recording.....	10
3.1.3	Automatic filtering .....	11
3.1.4	Manual filtering.....	11
3.1.5	System requirements for profile based recording .....	11
3.2	Fulfilment.....	12
3.3	Creation of meta-data.....	12
3.4	Delivery of meta-data in a broadcast channel.....	13
3.5	Consumption of meta-data in the receiver .....	14
<b>4</b>	<b>EXISTING STANDARDS &amp; TECHNOLOGY</b> .....	<b>16</b>
4.1	Content descriptions .....	16
4.2	User profile syntax .....	16
4.3	Descriptors Representation.....	16
4.3.1	RDF .....	16
4.3.2	XHTML.....	17
4.3.3	SMIL .....	17
4.3.4	CDF .....	17
4.4	Descriptors Transport.....	17
<b>5</b>	<b>PROPOSALS</b> .....	<b>18</b>
5.1	Proposal for automatic recording of TV-content .....	18
5.1.1	Position of the profile based recording component.....	18
5.1.2	Agent framework for profile based recording .....	18
5.2	Visual Descriptors and Indexing .....	19
5.2.1	Visual descriptors.....	19
5.2.2	Indexing .....	21
5.3	STORit XML .....	24
5.3.1	Location Resolution Table .....	24
5.3.2	Group Resolution Table .....	25
5.3.3	Meta Data Table .....	25
5.3.4	Segmentation Table.....	26
5.4	Channel Definition Format.....	26
5.4.1	Meta-Data API .....	29



# 1 Introduction

## 1.1 Purpose & Scope of Deliverable #5

Deliverable 5 covers the following workpackages;

- **310 Meta-Information and Advanced Indexing**  
WP Leader: Technical University Delft
- **320 Advanced Information Retrieval**  
WP Leader: Philips Italy
- **330 Personalised Service Access (user interest profiles)**  
WP Leader: Philips NL

The objective for the deliverable is to 'set requirements and propose a suitable database structure for standardisation of content descriptors in the area of home storage applications.'

The report will;

1. Analyse the objectives of the content 'indexing' system
2. Analyse the user's requirements for meta-information (attractors & descriptors)
3. Define an interface for the manipulation of meta-information
4. Describe framework for management of meta-information throughout the delivery chain
5. Consider user and system requirements for content capture based on 'user interest profiles'
6. Report on use of existing standards and ways in which STORit can provide input to current standardisation initiatives

## 1.2 Structure of Deliverable #5

The document will be split into the following five main sections;

1. **Introduction.** This section will explain the purpose of the whole document and it's relation to the project. This section will also outline the aims of the document (section 1).
2. **Description of a STORit system.** Description of application arenas and system outline.
3. **Requirements.** This will outline the individual components, technologies and algorithms that are required to produce such a system. This will also look at the functional requirements of these components;
  - Fulfilment (capture or delivery of required content)
  - Retrieval (from local storage)
  - Navigation within content
  - Navigation to other content
  - User interface issues - how do we present these functions to user?
4. **Existing standards & technology.** How we are using or implementing existing standards and technologies. How we are providing input to standardisation efforts.
5. **Proposals.** What we will do.

## 2 Description of a STORit system

### 2.1 The STORit applications

The STORit system supports two applications 'DVB/Internet' and 'Personalised Remote Learning.' These are described extensively in deliverable #2, but a brief summary is given below. Both, essentially, concern the interoperability of DVB, the Internet, and local storage. Combined use of these three media will result in additional user benefits.

#### 2.1.1 DVB / Internet application

In the DVB/Internet application, we aim to combine the possibilities of DVB, Internet and local storage-giving the viewer an entirely new way of selecting and watching television programs (now referred to in DAVIC as TV Anytime). This is based on several features, such as the ability to refer to a TV series from a web site. Here, simply 'clicking' on the link to a series ensures that the entire series will be captured on local storage. Similarly, clicking a button while watching a program trailer will guarantee that the program advertised will be recorded by the STORit system. Another important ingredient is the inclusion of additional data in the broadcast stream to support the TV programs. This ranges from traditional meta-data describing the programs (textual descriptions, categories, genres, images etc.) to relevant sections of the broadcasters web site. This meta-data will also be used by agents in order to select interesting content on behalf of the user, thus maintaining a sufficient supply of interesting material on the local storage media. For example, always the latest news and weather forecast, the two latest episodes of 'Eastenders' and a few good movies. A parallel can be drawn to a refrigerator, where one can always find some fresh food. The agent is the housekeeper monitoring the supplies. Removable media lying on a shelf are the equivalent of a freezer: a huge supply, but not instantly available.

#### 2.1.2 Personalised Remote Learning

In the 'Personalised Remote Learning' application (also referred to as 'Personalised Learning Assistant'), computer based training and television education (Open University) are combined, utilising the options offered by TV Anytime. Links between the computer based training components and the educational television programs will strengthen the coherence between the two. Thus, viewers watching an educational program will be led to the web based courseware, while students following the course via web or local storage will at appropriate stages be referred to the relevant parts of the accompanying TV programmes, which have been automatically captured on the STORit box. This application will illustrate how the TV Anytime framework, primarily designed with television entertainment in mind, is in fact perfectly suited for educational purposes as well.

## 2.2 System outline

From the application descriptions above, it is clear that our system will need to;

- Support DVB and internet access,
- Have the ability to insert (at the broadcaster's end) and extract (at the user's end) additional data in DVB,
- Have huge local storage capacity, both for video and other data,
- Allow regular PC use, esp. Internet browsing and commercially available computer-based training.

This suggests a good approach would be to separate the functionality in two 'boxes' in the user environment. A regular PC to carry out computing tasks and a 'STORit box' that gives access to DVB, the Internet and local storage. In addition to that, the broadcaster will need some tools to generate and insert the appropriate meta-data into the broadcast stream. This is reflected in Figure 2 below, where the STORit box is the gateway connecting the in-home digital network (IHDN) to DVB, internet and local storage, to be used by all PCs and (digital) televisions in the network.

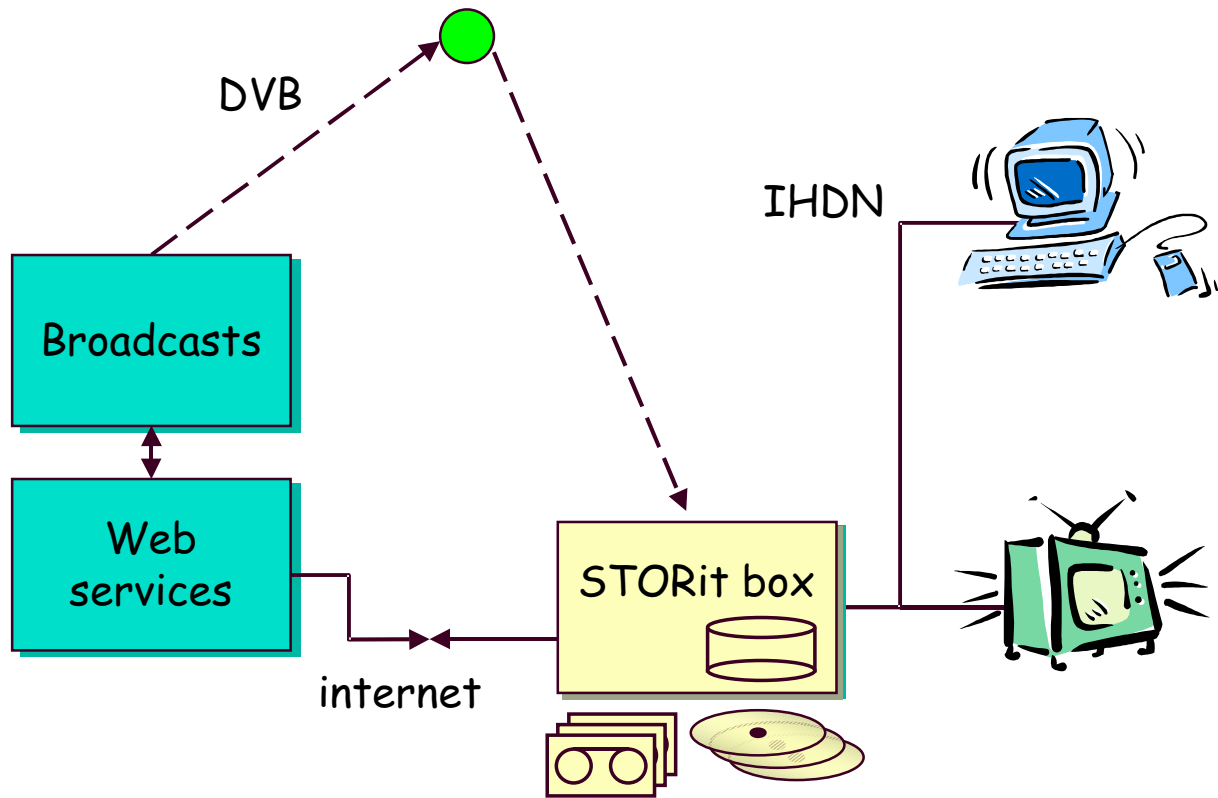


Figure 2.1 : The STORit box provides access to DVB, Internet and local storage, to be used by PCs and (digital) televisions connected to an in-home digital network.

## 3 Requirements

### 3.1 Requirements for automatic recording component

#### 3.1.1 Basic technology outline

The Personal Service Access component (*PSA*), which is currently being developed in wp330, enables the automatic recording of potentially interesting media content (i.e. TV-programs). The underlying selection mechanism is based on the notion of content filtering via profile information. This profile information expresses the interests of a user in an abstract way.

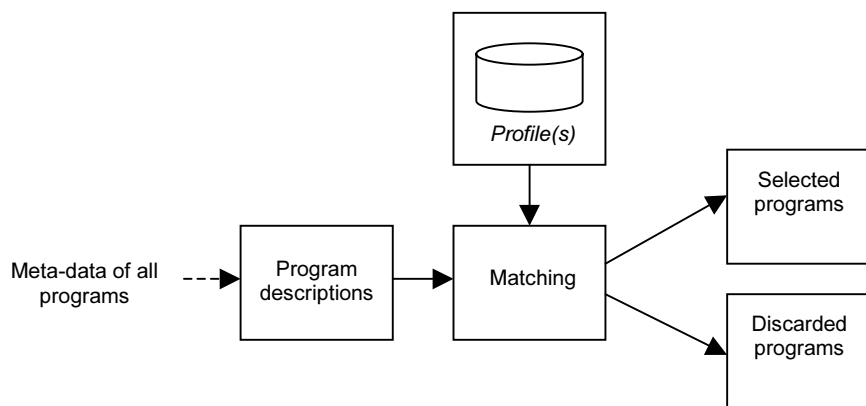


Figure 3.1: The basic principle of content filtering

The basic principle of profile based filtering is depicted in figure 3.1. Content descriptors in textual format (i.e. *meta-data*) are compared (*matching*) with information held in the interest profile. The level of similarity between profile and description is expressed in a matching score. This score will be used to determine whether a program should be selected for recording.

#### 3.1.2 User requirements for profile based recording

In the user requirements, as defined in by wp400 and wp330 in STORit Deliverable #6, we find several requirements relevant for the functionality of the profile based recording component. In general, the user requirements define three types of user interactions:

- **Navigating through content descriptions**  
via descriptions of future and past content and links to web pages, etc.
- **Navigating through the content itself**  
live broadcast as well as stored content
- **Management**  
importing, exporting, and removal of content; automatic and manual filtering of content

The requirements for wp330 fall into the 'Management' category, and are depicted below. Note that the word 'filter' reflects the user's view of the automatic recording functionality.

### 3.1.3 Automatic filtering

#### **Getting descriptions**

The user should be able to reduce the number of descriptions in the information space (*i.e. all the available content*).

#### **Getting content**

The user should be able to delimit the information space from which to record (*i.e. a filter should only record the programs that have an expected high probability of the user's interest*).

#### **Alter filter properties**

The user should be able to set and change filter properties. These are:

- Broadcaster
- Start time
- Language
- Language of sub-titling
- Country of origin
- Time of origin
- Genre & sub-genre
- Keywords (from a fixed list)

These properties can be used for getting and blocking content. Further, it should be possible to alter the filters from a higher level:

- Automatic recording on/off
- Individual filters on/off (using an overview of filters)
- Import/export filters
- Creating a new filter
- Removing an existing filter

### 3.1.4 Manual filtering

#### **Getting content and content descriptions**

The user should be able to reduce the number of descriptions and select content to be recorded. This content can be future content or content that is presently being broadcast. The user should be able to change the selection (in case of resource shortages).

#### **Overruling automatic filtering**

The user must be able to 'overrule' the filter. For example:

- The filter sets a program for recording, but the user is not interested and resets it.
- The filter has not set a program for recording, but the user is interested and sets it anyway.

### 3.1.5 System requirements for profile based recording

Content descriptors and profile information play a crucial role in the profile based recording component. In order to meet its user requirements satisfactorily, the component makes some specific demands on the syntax and semantics of both descriptors and profile.

The profile based recording component derives distinguishing program characteristics from the content descriptors. Therefore, the relevant content descriptors must be;

- *Standardised*, to avoid both ambiguity and the need for complex natural language interpretation,
- *Discriminating*, in such a way that potential interesting programs can be distinguished (automatically) from other programs.
- *Reliable*, in such a way that they are always available and describe the content correctly.

- Available prior to the actual broadcast, to enable timely evaluation and system preparation

Altogether, this means that at least some crucial attributes should be available, such as *title*, *language*, *channel*, *format*, *genre*, and *keywords*.

The profile reflects the system's knowledge about the tastes and preferences of the user. It is accessible for both for system and user. Therefore,

- At least a part of the profile should be expressed in the same attributes as the content itself.

This has two advantages. First, the relevant part of the profile can be manipulated directly via the user interface. Note that, since there is no stringent requirement to express the entire profile knowledge in this format, the use of non-editable profiles (e.g. based on neural nets) is still possible. Secondly, it simplifies the matching process. Next to this, study should clarify whether the use of weighting factors and logical compositions enhance the effectiveness of the profiles.

Some additional system requirements for this component are:

- The software should run on the same PC as the end-user application, developed in WP400
- The software should use the same operating system as the end-user application (Win NT)
- The software should be implemented in the Java programming language.

## 3.2 Fulfilment

In order to enable STORit applications such as filtering, search and retrieval of broadcast content, descriptive information must be delivered to receivers. Descriptors must be coded in a standard way, so that every set top box can parse them and use the information they carry.

Furthermore, since the project also takes into account interoperability with the Internet world, coding of such meta-data could be done using mark-up languages that are currently used on the Web. While this solution has the advantage of providing a text-based representation of descriptors (human readable and easily editable), proper coding must be performed before transmission in a DVB stream in order not to waste bandwidth.

Having said this, an initial analysis of the individual components of the system that is responsible for creating, delivering and using descriptors in a STORit framework can be performed.

## 3.3 Creation of meta-data

The descriptors that must be sent along with the content itself can be divided into (at least) four categories:

- [A] Resolution data
- [B] Descriptive information for programs
- [C] Segmentation data (for search and retrieval)
- [D] Trailers and background information.

Class [A] is used to resolve program identifiers into program locations (URLs) and program group identifiers into program components.

Class [B] contains the description of each program such as title, actors, scores and pointers to extra information sources.

Class [C] is the set of items such as keyframes, textual annotations or other kinds of meta-data, specific for each program, that can be used to allow navigation within content.

Finally, class [D] carries all the multimedia files that can be used to attract viewers to a certain program (e.g. trailers). In addition, program information in the form of Web pages. Each of the above classes can be prepared separately before transmission, possibly with the help of an authoring tool specific for the descriptor class.

For example, in cases [A] and [B], the authoring tool should retrieve information from the broadcaster's database and code it for transmission (using a mark-up language as described in section 5).

Class [C] descriptors may sometimes be generated by means of analysis algorithms that automatically generate segmentation meta-data. This segmentation data may also be manually keyed by the broadcaster in, for example, live broadcasts. Meta-data belonging to this class may be generated in real-time during the transmission of programs (for example, a key-frame extractor, incrementally building a visual summary of the transmitted program).

### 3.4 Delivery of meta-data in a broadcast channel

Once descriptors have been prepared, they can be broadcast in a DVB stream as files in a data carousel. This mechanism implements a 'push' service, in which meta-data is continuously sent to the receiver under control of the service provider.

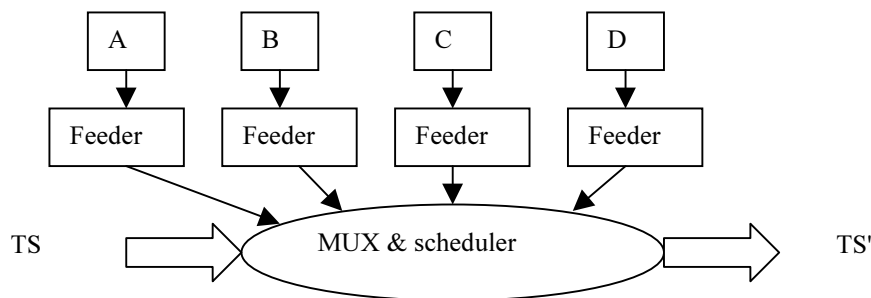


Figure 3.2 - Insertion of meta-data into a TS by class

Figure 3.2 shows how each descriptor class is generated by a separate process (the authoring tool) using a mark-up language. A feeder is then responsible for coding meta-data in an efficient way for transmission.

The process of meta-data insertion can be performed by a remultiplexor using null packets of an existing Transport Stream (TS): this solution implies that no modifications are needed in the existing multiplexors.

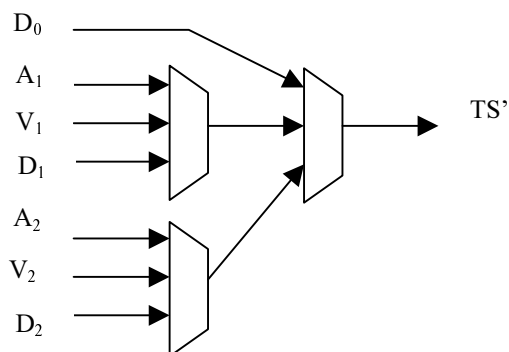


Figure 3.3 - Multiplex structure

If we indicate the annotated Transport Stream with TS', then the desired configuration is shown in figure 3.3.

Here,  $A_i$  indicates the audio stream of the  $i$ -th program,  $V_i$  represents the video stream and  $D_i$  the data stream carrying descriptors for the same program.

As it can be observed, a new program has been created in the multiplex ( $D_0$ ), containing descriptors for the whole transport stream such as resolution data, program meta-data and trailers. Separate descriptor elementary streams are also added to existing programs, in order to carry meta-data related to that program, such as segmentation data and background information.

For backward compatibility, DVB-SI tables are still included in the Transport Stream, resulting in a slight information redundancy.

### 3.5 Consumption of meta-data in the receiver

Meta-data transmitted via DVB must be extracted in the receiver and made available to the viewer and agent applications.

To achieve this functionality, the demultiplexor must be able to monitor the data carousels in the TS and check for new versions of descriptors. Then, it must notify interested applications and pass meta-data to them.

Since meta-data will be transmitted using files in a DSM-CC Object Carousel, the file containing an updated (or new) version of some descriptor will be sent to the receiver Content Manager for parsing, storing or other processing. In terms of Java implementation, this can be achieved with a simple mechanism like the one depicted below.

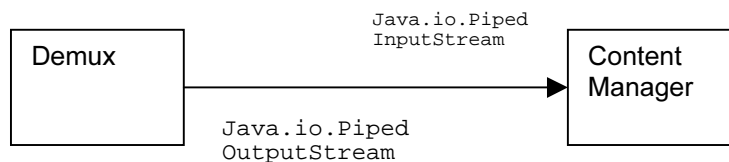


Figure 3.4 - Connection of modules in the receiver

Upon arrival of a new file, the demux writes to a PipedOutputStream, then notifies the Content Manager that, in turn reads from the connected PipedInputStream. This sequence of events is depicted in figure 3.5.

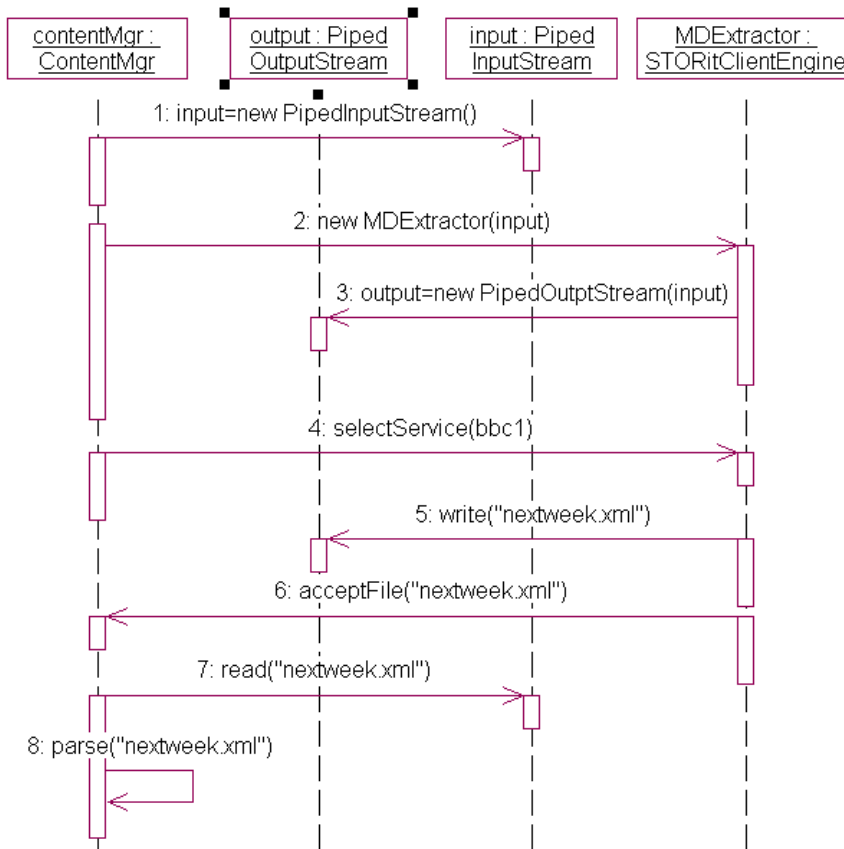


Figure 3.5 - XML extraction sequence diagram

## 4 Existing Standards & Technology

### 4.1 Content descriptions

The algorithms for the profile based recording component assume the classification of content to be compliant with a fixed format. Early experiments with such a component used content descriptions, compliant with DVB-SI. User tests however, revealed some shortcomings of this format. In particular the definition of program genres and sub-genres was not satisfying for most users. The classification, described in the document entitled "Meta-data for TV content", is partly based on the results of these tests.

### 4.2 User profile syntax

The user's interest profile will be partly expressed with the same attributes as used for the descriptions of the content. In addition, we consider the use of logical constructions and weighting factors in order to enrich the semantics of the user profile.

To support importing of user profile parts from the external world, we shall standardise the syntax of the user profile. We are considering the use of the Knowledge Interchange Format (KIF) as standardised syntax for the profile.

### 4.3 Descriptors Representation

As far as descriptor coding is concerned, the use of a mark-up language provides a human readable way of representing meta-data. XML (the eXtensible Mark-up Language) is flexible enough to be used in a STORit system, due to its easy customisation mechanism. Indeed, XML is a declarative language in which the structure of a document is specified in a standard way by means of a "Document Type Definition" (DTD) file: each XML document must reference its accompanying DTD or have it embedded. The DTD contains the rules for building an XML document and parsing it to retrieve the information it carries. Therefore, coding STORit descriptors in XML means defining a STORit DTD.

Some existing XML notations will be briefly analysed in the rest of this section.

#### 4.3.1 RDF

RDF stands for Resource Description Framework and represents an XML notation to describe meta-data on the Web. It provides mechanisms for resource discovery, cataloguing, rating of content, and privacy policies.

RDF is extensible since it is based on so-called "Schemas", that are sets of rules for meta-data representation in a specific domain. These can be successfully exploited by intelligent software agents for automatic filtering and retrieval.

The RDF basic data model consists of Resources, Properties and Statements that can be combined to build a very complete description of the content<sup>1</sup>. Schemas provide dictionaries that can be augmented in time, being therefore scalable.

---

<sup>1</sup> Example of a RDF statement: "The programs that make up the Star Trek Series for 1998 are: Episode #1, Episode #2, Episode #3, they are maintained by BBC and their description can be found at [www.bbc.com](http://www.bbc.com) or [www.startrek.org](http://www.startrek.org)".

### 4.3.2 XHTML

XHTML is another mark-up language specifically tailored for broadcast applications. It is a scalable language in that it provides support for applications that run on different hardware platforms. From small PDAs to Set-Top Boxes and PCs (compact, broadcast and www profiles). The different capabilities are managed by having different DTDs for each hardware profile. XHTML also defines an event model that is intended to support actions taken by the end-user.

### 4.3.3 SMIL

Short for Synchronised Multimedia Integration Language, a new mark-up language being developed by the World Wide Web Consortium (W3C) that would enable Web developers to divide multimedia content into separate files and streams. Audio, video, text, and images are sent to the user's computer individually, and are then displayed together as if they were a single multimedia stream. The ability to separate out the static text and images should make the multimedia content much smaller so that it doesn't take as long to travel over the Internet.

SMIL is based on the eXtensible Mark-up Language (XML). Rather than defining the actual formats used to represent multimedia data, it defines the commands that specify whether the various multimedia components should be played together or in sequence.

### 4.3.4 CDF

CDF (Channel Definition Format) is the format that is currently in use on the Web to enable the so-called 'push' services supported by a variety of browsers. The CDF format defines which items (resources) belong to a channel, how often they are updated, for how long their content is valid. In this way, a publisher can offer a variety of continually updated information services to clients that have subscribed to the channel.

At present, several tools exist for writing and parsing XML files; these processes often make use of the DOM (Document Object Model) API, specified by the W3C consortium.

The DOM API defines interfaces to access document elements and get, or set, attribute values. Implementations of the DOM API are available by several vendors for different programming languages, including Java.

## 4.4 Descriptors Transport

The STORit content description scheme is far more complicated and exhaustive than the standard DVB-SI tables. Therefore, an extension to the tables themselves does not seem a viable solution. Mechanisms for transporting data streams in a DVB channel already exist such as Multiprotocol Encapsulation and the DSM-CC Object Carousel.

The latter, in particular, allows the cyclic delivery of a structured group of files, directories and streams to the receiver. Access to objects carried in the carousel is enabled through a Service Gateway by means of object identifiers (Interoperable Object References, IOR) defined in CORBA. Object data is transmitted in BIOP (Broadcast Inter ORB Protocol) messages that are, in turn, segmented and inserted into MPEG-2 private sections.

IORs contain information about the location of associated objects. The DSM-CC Object Carousel defines interfaces for the manipulation of File, Stream, Directory, ServiceGateway and StreamEvent objects. DSM-CC is linked to a DVB service by means of an entry in the Program Map Table.

On the other hand, Multiprotocol Encapsulation can be used to transmit datagrams in DVB networks: encapsulation is performed by means of DSM-CC sections that are compliant with MPEG-2 private sections.

In either case, the transport of descriptors coded using a mark-up language (and therefore in textual form) could be specifically optimised to save bandwidth.

## 5 Proposals

### 5.1 Proposal for automatic recording of TV-content

This paragraph outlines the implementation approach for a profile based recording component, which meets the requirements stated in section 3. As we shall see, a direct implementation of the basic filtering principles is not desirable; user tastes are simply too complex and too variable to describe them in one single profile. In a short overview, we present a more sophisticated filtering framework in which the profile is split in multiple sub-profiles. This reduces the complexity of the profile as well as the matching algorithms while increasing the flexibility.

#### 5.1.1 Position of the profile based recording component

The position of the profile based recording component in the STORit environment is depicted in figure 5.1.

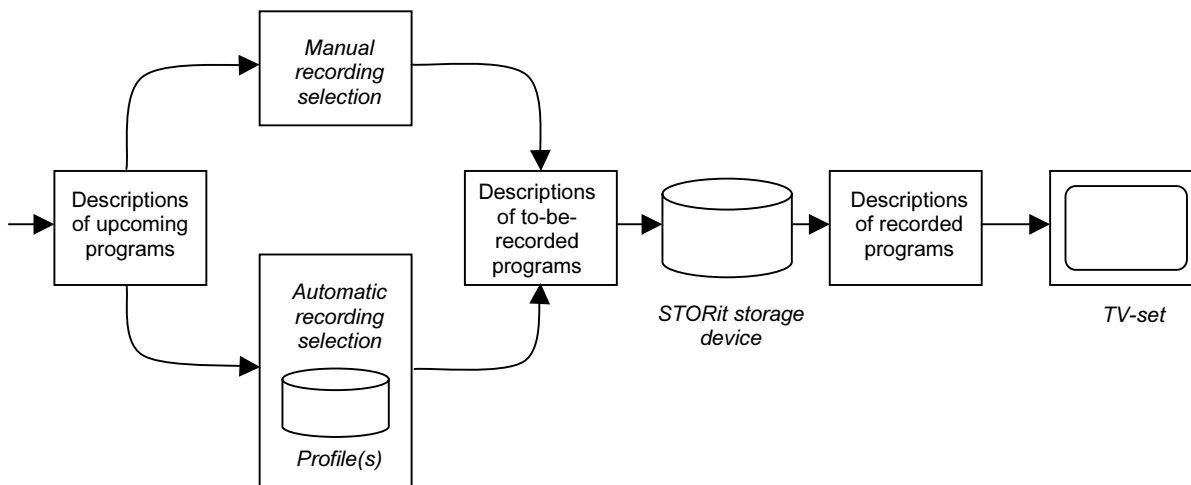


Figure 5.1: The position of the profile based recording component in the STORit environment

At the left-hand side, content descriptions enter the home. Similar to the basic filtering principle, the profile based recording component compares all descriptions with its user profile. If the match for a program is strong enough, and sufficient system resources are available, the program is scheduled for recording. This recording schedule is visible for the user, who may add programs by hand, and may delete automatic selected programs ('overruling').

#### 5.1.2 Agent framework for profile based recording

Users may find it difficult to describe and formulate their own taste, irrespective of the format in which they do this. Therefore, default profiles can be imported, and may be edited via the user interface. However, even for imported profiles the description of a broad 'spectrum' of tastes and preferences might become too complex. For reasons of flexibility we allow the splitting of a user profile in several partial profiles, each with their own speciality (*sports profile*, *movie profile*) or characteristics (*Michael Jackson's profile*).

We defined a *multiple agent system* (MAS) in which each partial profile is implemented as an agent ('juror agent'). A juror agent consists of a partial profile and its private matching intelligence. Each juror agent evaluates an incoming program, after which a weighted average matching score over all the jurors is calculated. Each weighting factor, indicating the juror's influence in the overall filtering process, is based on the juror's success in the past.

The advantage of implementing the component in a MAS is twofold:

- The overall complexity is minimised, by placing the intelligence where it belongs: in the juror, near its corresponding partial profile.
- New partial profiles, together with their optimised matching intelligence, can be added or deleted easily, as long as they are compliant with the internal agent communication protocols.

A simplified overview of the MAS is depicted in figure 5.2. The list at the left-hand side contains the incoming programs, with their descriptions in the box above it. Programs, with average scores exceeding a threshold level will enter the recording list at the right hand side.

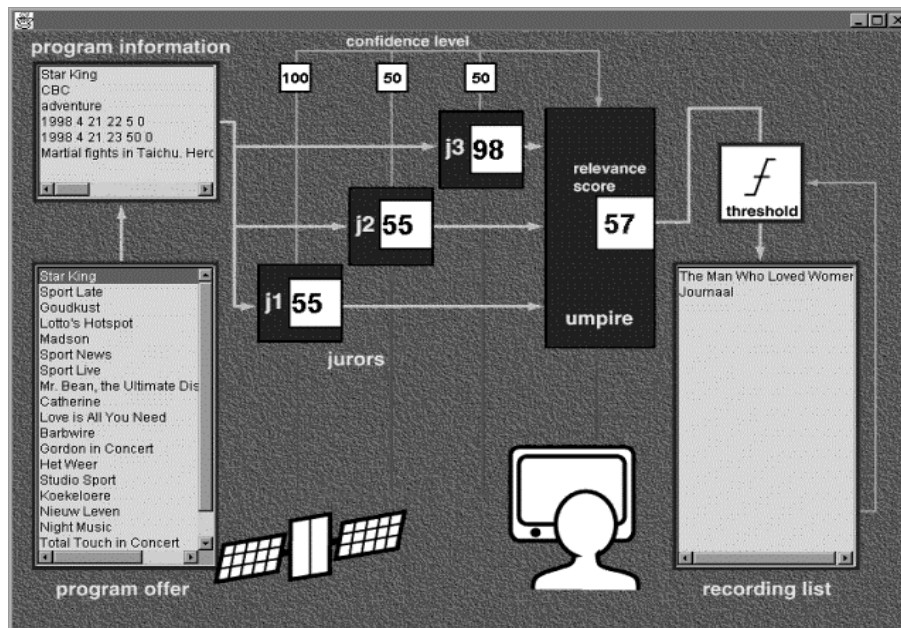


Figure 5.2: Simplified overview of the MAS for profile based recording of TV-content

## 5.2 Visual Descriptors and Indexing

To create an enhanced consumer system that helps a user deal with the information overload of hundreds of television channels additional information is necessary- meta-data. Meta-data can help the user select material worth viewing. Intelligent agents can actively filter this meta-data and only show the programs a user might be interested in, they could suggest programs worth watching, or even fill an entire evening with entertaining programs. To enable the users and agents to make informed decisions, the meta-data should reflect the content of the programs. Most of this data has to come from the broadcaster or content provider, since most of it can not be generated at the home.

Two types of meta-data have been identified, textual descriptors and visual descriptors. The basic elements of textual descriptors are words, whereas the basic elements of visual descriptors are key-frames. In this chapter, we will first focus in detail on the structure of the visual descriptors. Secondly, we will focus on the requirements of the indexing process, which generates the textual and visual descriptors.

### 5.2.1 Visual descriptors

There are many ways of describing the visual content of a certain video segment, and not all aspects can be represented in a limited set of textual or visual descriptors. In other words, we do not believe in a scheme that completely describes the video content, other than the video itself. However, turning to the specific functional requirements of the application, we find that we have the following requirements for the visual descriptors. They should allow for automatic creation of program summaries;

1. allow for browsing video databases on different levels of detail;
2. allow for video filtering and retrieval on the basis of examples;

Although all of the above functional requirements can be implemented (to some extent) using solely textual descriptors, a more advanced (and possibly more efficient) fulfilment of the requirements is based on visual descriptors. Browsing a video database may often require something more than viewing a number of frames from a video program (as it is now in the 'Smash' project). For example, a query (based on textual descriptors) can be refined by means of user feedback and visual browsing of the query results. As another example, the user might want to browse parts of the video database, across programs, which contain visually similar scenes/shots etc. For these more advanced functions, visual descriptors should be available which form a visually *representative* summary of a program.

We consider a visual summary as a collection of the most *representative* key-frames. Since the concept of "representivity" does not have a formal definition, nor does it benefit from a precise measure, we allow the summary to be constructed at different levels of detail. The final goal is to achieve a hierarchical structure of key-frames, which best represents video content.

### 5.2.1.1 Structure for visual descriptors

The visual descriptors are organised in a hierarchy of key-frames. User studies have indicated that at least three levels should be available in such a hierarchy. At the lowest level the key-frames represent program **shots** (detailed), at the intermediate level the key-frames essentially represent program **scenes** (coarse). At the top level, the key-frames represent a program **segment** (global). Figure 5.3 shows this hierarchy.

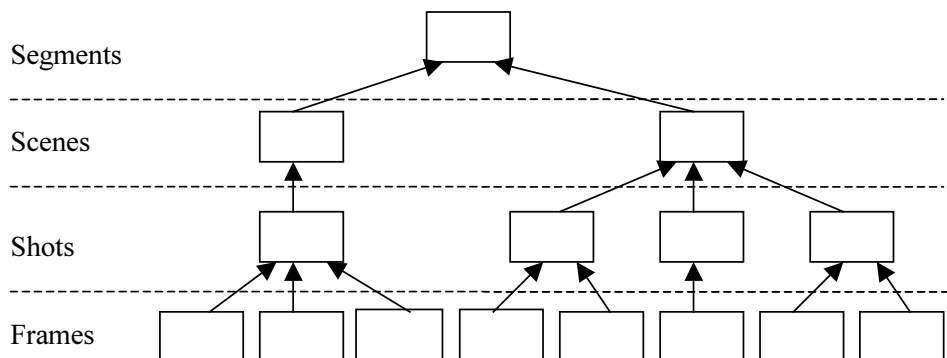


Figure 5.3. Key-frames hierarchy.

The program shot has been defined in the 'Smash' project. The **scene** is an intermediate level; a set of shots that show a visual similarity. The **segment** is a collection of scenes and should *ideally* be equal to an **item**, which is the lowest level of detail for the textual descriptors. The advantage of having both visual and textual descriptors on the same hierarchical level is that retrieval can be based on both types of meta-data. For example, refining a textual search by using item-level key-frames (also cross content browsing can be implemented in this way).

It is important to note that the program segment, which is to be described by the hierarchy of key-frames, is not necessarily the program item. In principle, the whole program can be represented by such a set of organised key-frames. However, we prefer to visually represent the lowest level of the program that is described by textual descriptors in order to benefit from both types of meta-data and have a meaningful upper level of the hierarchy. As such, the visual descriptors can be seen as additional and complementary to the textual descriptors.

It should be noted that for a user at least three layers in the hierarchy should be available, however more layers are possible and even necessary. The reason for having a hierarchy consisting of more layers is that not every user has the same perception of global, coarse and detailed. With more than three layers, it is possible to adapt to the user needs by allowing him to select his own favourite three layers.

## 5.2.2 Indexing

The process of associating meta-data with video content is called *video indexing*, or indexing for short. Various ways of video indexing can be envisaged, depending on the functional requirements of the system. For the STORit service and the associated recording/filtering/playing-back device, it was decided that the indexing scheme should include textual, as well as visual descriptors, each of which enables different components of the overall functionality. While textual descriptors are appropriate for describing the content of large video segments (program groups, programs, and program items - according to the proposed hierarchy of textual descriptors), visual descriptors are more appropriate for detailed content.

### 5.2.2.1 Indexing textual descriptors

Research conducted in the past year has indicated that at the current level of technology, textual descriptors cannot be entered automatically. Therefore, indexing textual descriptors will rely on a manual indexing process on the content provider's side.

Study of the indexing process at the provider's side has revealed that indexing can best be done during the production of the content. During production, many documents are generated that record details such as; shots, scenes, items, actors, and the director. The problem for many production companies and content providers is that such documentation is not standardised. Indexing research can contribute to standardisation of this form of production documentation. Furthermore, indexing is not simply a concern during post-production but should take place during production.

The best form of indexing is entering the textual descriptors by allowing selection from a list of keywords. This will minimise the chance of typing errors. Even for keywords, we propose a large but limited set. However, provisions have to be made to allow for alteration of the keyword list. It should be possible to add and remove keywords, since new keywords will emerge and some keywords might become outdated.

### 5.2.2.2 Indexing visual descriptors

In order to solve the particular problem of automatic visual indexing (representation of a video segment by using a hierarchy of key frames) a more general problem has to be addressed first: visual similarity. The fundamental question is whether visual similarity should be obtained before interpretation of an image (frame) or after interpretation. After interpreting an image, a *conceptual* similarity can be obtained. For instance, the image of a 'Formula One' racing car is somewhat similar to an image of an off-road racing car. Before interpretation, similarity can be obtained by using visual information such as; colour, texture, and shape.

Since it cannot be expected that in the near future the general problem of image and video interpretation will be solved, we have to address the problem of visual similarity by using visual information (features). Although there has been a large amount of work in the area of image analysis, it is not yet known what visual information sufficiently represents an image in order to determine the similarity with other images. MPEG-7 may provide some answers here as 'similarity based retrieval' of images and video is one of the main applications addressed.

#### 5.2.2.2.1 Visual similarity

The problem of visual content similarity has been addressed in the context of image and video searching and retrieval from multimedia databases. Most current techniques rely on low-level information, like colour histograms. By comparing such data from one frame with the other, the

similarity between the frames can be obtained. The main advantage of such techniques lies in their low computational complexity. The effectiveness of these techniques has been proved for some particular cases, in which the visual content can be accurately characterised in terms of dominant colour (like "sunset", "summer landscape", etc.).

It is widely believed that representing images by global information such as colour histograms is not a general solution to the problem of visual similarity between images. Using colour histograms for *regions* in images is a much better approach since this allows for comparisons between images having a more complex composition and allows partial matching. The main problem here is the lack of structural information. What homogeneous regions are where in the image? In other words, how should the image itself be segmented and represented? Although in the areas of image retrieval and video coding (e.g. MPEG 4) structural information becomes increasingly important (blotch representations, object-based coding), there does not yet exist a standard for the *segmentation* of the image itself. Most segmentation algorithms focus on finding "exact" (object) regions (for image interpretation or efficient coding), are rather time consuming, and not successful for all kinds of images. The four requirements for an *ideal* segmentation are:

- it should incorporate structural information (represent the objects in the image),
- it should be applicable to any kind of image,
- it should allow for a (partial) image similarity measure that is scale, translation, and rotation invariant,
- it should be computationally fast.

A representation that exactly satisfies these requirements does not exist. However, an image segmentation method used for determining image similarity does not have to satisfy these requirements exactly. This is because the similarity will be used for ordering the key-frames only and not for more demanding tasks like object recognition or image interpretation. Therefore, a basic, rough, segmentation algorithm can be used that results in regions whose sizes depend on the local structure of the image. A simple and efficient way of doing this is by using the quad-tree-splitting algorithm in which the homogeneity criterion is connected to the local structure of the image. Although we do not expect a perfect match of two sub-trees representing the same "object" in two different images, some invariance on scaling, translation and rotation can be achieved.

#### 5.2.2.2.2 Quad-Tree splitting algorithm

The algorithm we used is the well-known recursion in which, at each step a homogeneity criterion is tested:

1. Let the whole image be the root of the tree.
2. Divide the image into four quadrants. Assign to each quadrant a label  $x_i$ ,  $i \in \{0, 1, 2, 3\}$ , according to an arbitrary rule (e.g. begin with the upper-left quadrant and continue in clockwise manner), using as prefix the label of the parent node.
3. Test the homogeneity criterion for each quadrant. If a cell is homogeneous or the cell dimensions are lower than a given value, **STOP**. Otherwise **GOTO** Step 2.

As we said before, we use homogeneity criteria that capture the local structure of the image. Some of these are:

1. *Pixel value variance*: Let the current node  $Q_p$  represent a rectangular area of  $M_p \times N_p$  pixels. The node, having the pixel values  $\{u_{ij}\}$ ,  $i = 1, 2, \dots, M_p$ ,  $j = 1, 2, \dots, N_p$  and their mean value  $\mu_p$ , is considered to be homogeneous when

$$\sqrt{\frac{\sum_{i=0}^{M_p} \sum_{j=0}^{N_p} (u_{ij} - \mu_p)^2}{M_p N_p}} < \theta, \text{ where } \theta \text{ is a given threshold value.}$$

2. *Edge density*: For the current node we first detect the discontinuity points using a first order differential operator. We decide that at one point  $(m, n)$  there is a discontinuity if the output of the operator:

$$u_{mn} - \sum_{i=-\frac{M_W}{2}}^{\frac{M_W}{2}} \sum_{j=-\frac{N_W}{2}}^{\frac{N_W}{2}} u_{m-i, n-j} c_{ij}, \text{ with } c_{ij} > 0, \text{ is above a given value. Further on, if the}$$

percentage of discontinuity points within the current node  $Q_p$  is smaller than a given threshold, then the node is considered homogeneous.

3. *DCT coefficients variance*: This criterion aims to detect local homogeneity with respect to spectral components. The basic idea is to compute the DCT coefficients for each node and to apply criterion 1 for these values

Based on these criteria, a quad-tree can be constructed that forms a structural description of an image (frame). Given a structural representation of the frames using Quad-Trees, the similarity between images is determined by comparing their individual Quad-Trees. The advantages of using Quad-Trees for similarity are:

- The use of a quad-tree division of images induces some invariance properties, which allow a reliable comparison of similar regions at different scales and positions. At the same time, it ensures that the optimal number of regions is used.
- The homogeneity criteria used in the splitting algorithm (pixel value variance, edge density, and DCT coefficients variance) lead to a quad-tree that captures structural information, although only colour information is explicitly used further.
- Comparing groups of adjacent regions minimises the possibility of false matching. The average region sizes together with the number of similar regions give a more reliable measure of similarity than a simple distance function.

The drawback of the Quad-Tree approach lies in the comparison of the trees, which can be computationally expensive if accurate results are required. To determine these requirements and to design an optimal comparison, extensive user studies are necessary.

#### 5.2.2.2.3 Automatic Visual Indexing

The final organisation of the key-frames in distinctive levels of detail is obtained by using a hierarchical clustering procedure. In hierarchical clustering, the levels are automatically obtained, and the number of levels heavily depends on the complexity of the program. During clustering, a group of similar key-frames is represented by the key-frame being the most similar to each of the key-frames in the group. By joining groups of key-frames, new levels are formed which need less representative key-frames. At each higher level, the similarity between the groups becomes smaller such that, at the highest level, a single key-frame represents the complete program segment. A program containing many similar key-frames can therefore be organised in fewer levels than a program having many dissimilar key-frames.

With this approach, automatic indexing of the visual descriptors is possible for a program segment. However, the representivity of the visual descriptors for that segment as perceived by the users should be carefully studied and evaluated.

It will be the focus of further research to determine whether a segment, such as a program item, can be determined automatically based on key-frame hierarchy.

## 5.3 STORit XML

In the STORit project, we'll be sending data streams along with the digital video streams on DVB channels:

- Location Resolution data; giving where and when programs will be broadcast.
- Group Resolution data, giving information on the composition of program groups.
- Segmentation data, giving information on the segmentation / itemisation.
- Meta-data, describing programs (~ groups) in order to attract viewers.

The resolution information will probably be sent more frequently than the other data. The key to all this information is the UPI, Uniform Program Identifier, connected to each program (or program group). Programs will have to be able to 'identify themselves,' i.e. you will require access to the UPI for the program currently on air. To allow for accurate recording of programs you will need to know;

1. The UPI of a program 'coming up next' so that the start is not missed
2. A frame-accurate means of finding the start and end of a program

In order to be modern and flexible, we should use XML notation for these data streams, though the data actually transmitted may well be compressed in some way.

### 5.3.1 Location Resolution Table

Below is a sample XML code, proposed for use in the location resolution tables. A formal definition can be made once we agree on the format.

```
<LOCATION_RESOLUTION_TABLE>

<!-- here's an example of a program that will be broadcast twice
-->

  <PROGLOC ID="upi:bbc:500021" TITLE="Star Trek: Spock in trouble ">
    dvb://bbc..bbc2;1a639f@19981124T1845D0045
    dvb://bbc..bbc2;1a639f@19981125T1030D0045
  </PROGLOC>

</LOCATION_RESOLUTION_TABLE>
```

This resolution table contains one entry revealing resolution information for one episode, which is broadcast twice.

The table may contain more such entries, but need not always constitute a complete schedule for a given television service. In other words, the resolution information for a given schedule may be sent piece by piece in any number of files. It will be up to the STORit box to assimilate all these files into a single 'resolution database.' This allows for last minute scheduling changes and the STORit box must be able to update the resolution database accordingly.

### 5.3.2 Group Resolution Table

Below is a sample XML code, proposed for use in the group resolution tables.

```
<GROUP_RESOLUTION_TABLE>

<!-- here's an example of a group which is not complete, i.e. has more
episodes than the three listed. The 'REPLACE' attribute suggests how the
actual recordings of the programs might be managed by the STB. A value of
'TRUE' suggests that each successive recording will replace the last.
'FALSE' suggests they do not replace one another.
-->

    <GROUP ID="upi:bbc:5000"  TITLE="Star Trek 1998 series"

        COMPLETE=FALSE  ORDERED=TRUE  REPLACE=FALSE>

        upi:bbc:500021

        upi:bbc:500022

        upi:bbc:500023

    </GROUP>

</GROUP_RESOLUTION_TABLE>
```

This resolution table contains one group entry, identifying the next few episodes of this year's Star Trek series, indicating the list is not complete, is ordered, and the episodes do not replace one another.

### 5.3.3 Meta Data Table

Below is a sample XML code, proposed for use in the meta-data tables.

```
<META_DATA_TABLE>

<!--

-->

    <PROGMD ID="upi:bbc:500021" TITLE="Star Trek: Spock in trouble ">

    <DESCRIPTION>
        Yet another episode of this great series from the sixties.
    </DESCRIPTION>

    <SYNOPSIS>
        After reacting to a distress signal, the
        Enterprise gets trapped in a morphogenetic force
        field. Spock decides to investigate...
    </SYNOPSIS>

    <GENRE>  Science Fiction </GENRE>

    <FORMAT> TV drama </FORMAT>
```

```
<MM_LINK TYPE=IMAGE TITLE="Spock"
URL=http://www.bbc.co.uk/Startrek/img021.jpg />

<MM_LINK TYPE=SOUND TITLE="Hear the Enterprise"
URL=http://www.bbc.co.uk/Startrek/Enterprise.wav />

<MM_LINK TYPE=WEB TITLE="Star Trek home page"
URL=http://www.startrek.com/index.html />

<TRAILER> upi:bbc:500021t </TRAILER>

<PERSONNEL>
  <PERSON ROLE="Camera" NAME="Joe Bloggs" \>
  <PERSON ROLE="Dr. Spock" NAME="Leonard Nimmoy" \>
</PERSONNEL>

<KEYWORDS>
  phaser "transporter room" space
</KEYWORDS>

<PROG_LINK TITLE="Space 1999"> upi:bbc:3000 </PROG_LINK>

<LANGUAGE SOUNDTRACK=English SUBTITLES=TRUE SUB_LANGUAGE=English/>

</META_DATA_TABLE>
```

This table contains one entry, similar entries would exist for each program or program group that is relevant for, say, the next two days.

### 5.3.4 Segmentation Table

Below is a sample XML code, proposed for indicating the segmentation of programs.

```
<SEGMENTATION_TABLE>

  <PROGSEG ID="upi:bbc:900345" TITLE="Match of the Day: Glasgow
  Rangers - Celtic ">

    <SEGMENT NUMBER=1 TITLE="First goal" TIMECODE=00:12:13 />

    <SEGMENT NUMBER=2 TITLE="Red card" TIMECODE=00:41:30/>

    <SEGMENT NUMBER=3 TITLE="Second goal" TIMECODE=00:88:10 />

  </PROGSEG>

</SEGMENTATION_TABLE>
```

This table would be expanded as the program progresses and more goals are scored.

## 5.4 Channel Definition Format

If descriptors are to be carried via the DSM-CC Object Carousel, then we need a way of organising the XML files that contain the meta-data. A possible solution is using CDF (Channel Definition Format – see section 4) as a Table of Contents for our descriptor files. Here, we propose to use CDF to;

- indicate the XML files that must be parsed by the receiver when looking for a particular descriptor class (e.g. resolution tables),
- maintain schedule information so that file updates are checked only when appropriate,
- provide an indication for the expiration of the validity of the information contained.

Overleaf, a CDF file is given as an example.

```
<!DOCTYPE Channel SYSTEM "dvb://bbc..bbc1/DSM://bbc/dttds/cdf.dtd">
```

```
<!-- This is a description of an information channel that carries  
different kinds of meta-data for STORit purposes, using W3C standard CDF  
(Channel Definition Format) notation.
```

The main channel has three subchannels as follows:

- 1) resolution data channel,
- 2) metadata channel,
- 3) trailer channel.

Both channel definitions in CDF format and referenced XML documents are assumed to be carried in a DSM-CC Object Carousel (inside a DVB stream) as files.

Notice that the definition of the CDF format used in this document is also carried in the object carousel as the W3C Document Type Definition (DTD) file "cdf.dtd", as stated in the 1st line of this document.

The notation for references to files in an DSM-CC Object Carousel follows DAVIC1.3 specifications (part09, chapt. 9.3).  
-->

```
<Channel HREF="dvb://bbc..bbc1/DSM://channels/bbcPrograms.cdf"  
  IsClonable=YES >  
  <LastMod VALUE="19981213T1325+0100" />  
  <Title VALUE="BBC Programs December schedule"/>  
  <Abstract VALUE="All the programs of BBC1, BBC2 and BBC3" />  
  <Author VALUE="D.Melpignano"/>  
  <Schedule>  
    <EndDate VALUE="19981231T2400+0100"/>  
    <IntervalTime HOUR=2 />  
    <EarliestTime HOUR=7 />  
    <LatestTime HOUR=22 />  
  </Schedule>  
  <Logo HREF="dvb://bbc..bbc1/~//logos/bbc1.png" />
```

```
<!-- First subchannel: Resolution data -->
```

```
<Channel HREF="dvb://bbc..bbc1/~//channels/resolutionData.cdf"  
  IsClonable=NO >  
  <LastMod VALUE="19981213T1330+0100" />  
  <Title VALUE="Resolution Data Channel" />  
  <Abstract VALUE="Channel carrying tables to resolve UPIs  
into URLs">  
  <Schedule>  
    <EndDate VALUE="19981220T2400+0100" />  
    <IntervalTime HOUR=4 />  
    <EarliestTime HOUR=0 />  
    <LatestTime HOUR=24 />
```

```
</Schedule>
```

```
<!-- Here we assume that resolution data is carried in separate XML files,
in order to be more maintainable: each file follows the syntax defined in
the "STORit XML" document draft 0.1, by E. Montie. -->
```

**(SEE SECTION 5.3)**

```
<Item HREF="dvb://bbc..bbc1/~channels/tables/nextweek.xml" />
<Item HREF="dvb://bbc..bbc1/~channels/tables/nextmonth.xml" />
<Item HREF="dvb://bbc..bbc1/~channels/tables/bbc1.xml" />
<Item HREF="dvb://bbc..bbc1/~channels/tables/bbc2.xml" />
</Channel>
```

```
<!-- Second subchannel: Program Metadata -->
```

```
<Channel HREF="dvb://bbc..bbc1/~metadata/metadata.cdf"
  IsClonable=NO >
  <LastMod VALUE="19981213T1400+0100" />
  <Title VALUE="Metadata Channel" />
  <Abstract VALUE="Channel carrying descriptive program
  information">
  <Schedule>
    <EndDate VALUE="19981220T2400+0100" />
    <IntervalTime HOUR=4 />
  </Schedule>
```

```
<!-- Notice that categories as format, genre and subgenre are defined in a
separate file named "storitCategory.xml" as CategoryDef elements according
to the W3C CDF spec. -->
```

```
<Item HREF="dvb://bbc..bbc1/~metadata/seriesMD.xml" >
  <Title VALUE="Information on series, serials etc"/>
<CategoryHREF="dvb://bbc..bbc1/~metadata/
storitCategory.xml#series" />
</Item>
<Item HREF="dvb://bbc..bbc1/~metadata/moviesMD.xml" >
  <Title VALUE="Information on movies." />
  <Category HREF="dvb://bbc..bbc1/~metadata/
storitCategory.xml#movies" />
</Item>
<!-- Add any other references to metadata resources. -->
</Channel>
```

```
<!-- Third subchannel: Trailers -->
```

```
<Channel HREF="dvb://bbc..bbc1/~trailers/trailer.cdf"
  IsClonable=NO >
  <LastMod VALUE="19981213T1330+0100" />
  <Title VALUE="BBC Trailers Channel" />
  <Abstract VALUE="Programs until Dec, 20th" />
  <Schedule>
    <EndDate VALUE="19981220T2400+0100" />

    <IntervalTime HOUR=4 />
  </Schedule>
```

```
<!-- XML files containing trailers can reference Java applications
contained in the object carousel. -->
```

```
<Item HREF="dvb://bbc..bbc1/~trailers/moviesTR.xml" >
```

```
        <LastMod VALUE="19981213T1335+0100" />
        <Title VALUE="Movies" />
        <Abstract VALUE="Some commentary text" />
    </Item>
    <Item HREF="dvb://bbc..bbc1/~/trailers/sportsTR.xml" >
        <LastMod VALUE="19981213T1335+0100" />
        <Title VALUE="Upcoming sports events" />
        <Abstract VALUE="Some commentary text" />
    </Item>
</Channel>
</Channel>
<!-- End of main channel definition document. -->
```

### 5.4.1 Meta-Data API

In order to handle descriptors, a meta-data API has to be defined that allows applications to:

1. build a descriptor tree (add child descriptors, allow navigation)
2. get/set descriptor properties
3. write the descriptors tree content into XML
4. parse an XML file to retrieve the descriptors tree
5. pass descriptor data from one module to another.

A set of interfaces must be defined for the objects that manipulate descriptors and for the descriptors themselves. Here an interface is proposed containing methods to handle the most common operations on meta-data.

The following interface defines generic operations to deal with program components and compositions, including items and groups.

Public Operations:

---

**getComposite () : MetaData**

Returns the object reference if the object is composite, null otherwise..

**add (component : MetaData) : boolean**

Adds child metadata.

**delete (component : MetaData) : boolean**

Deletes child metadata.

**accept (visitor : Visitor) : boolean**

Accept a visitor to be used for navigation in the metadata tree.

**setProperty () :**

Sets a property on the metadata object. Uses reflection.

**getProperty () : Object**

Gets a property from the metadata object using reflection.

**toString () :**

Writes the metadata object as a String: depending on the implementation, can be used to produce an XML output.

It is proposed here, that Program Groups, Programs and Program Items (the three main descriptor classes) all implement the same meta-data interface. This is shown in the class diagram of figure 5.4.

ANNOTATION CLASSES

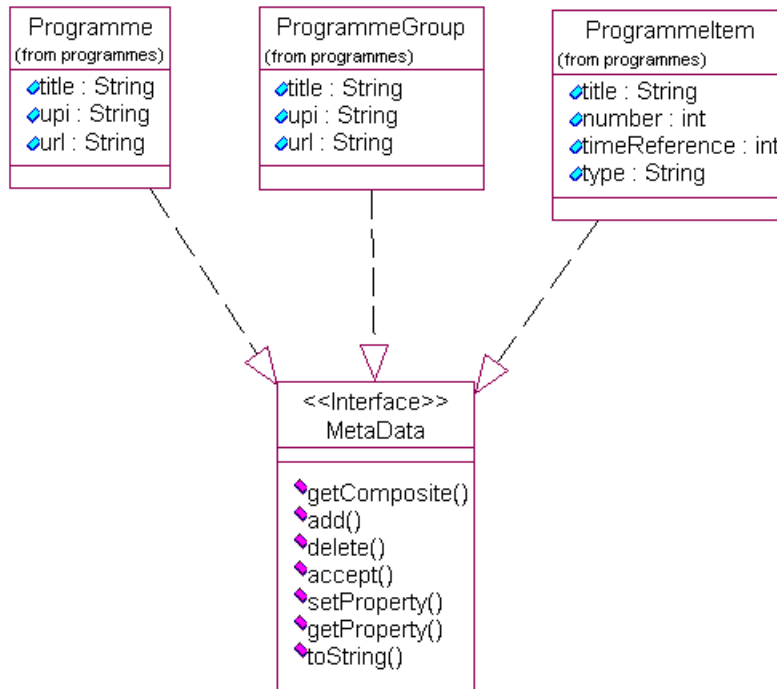


Figure 5.4 Meta-Data class diagram.