# The ASSET Architecture—Integrating Media Applications and Products through a Unified API

**By M. Cordeiro, P. Viana, J. Ruela, M. Body, B. Cousin, D. Bommart, G. Ferrari, W. Bernet, M. Strambini, E. Müller, M. Laurentin, B. Algayres, S. Daulard, I. Höntsch, and T. Marx**


M. Cordeiro


P. Viana


J. Ruela

***Applications and products currently available for the broadcasting market are vertically integrated or proprietary. They are based on components requiring specific and costly development to interoperate and typically rely on a single manufacturer or system integrator. Hence, they are not fully compliant with broadcasters' requirements. Architectural Solutions for Services Enhancing digital Television (ASSET) is a European-funded project. Its main goal is to overcome the limitations of custom-specific implementations in a digital system for TV content creation. These limitations are generally due to the misfit of interfaces between software layers, proprietary application program interfaces (APIs) of equipment from different vendors, and the lack of a generalized middleware for multimedia content management with openly defined interfaces. Besides presenting the ASSET-proposed architecture and concepts, this paper describes the prototype under development to test and demonstrate the project proposals.***

The main goal of the Architectural Solutions for Services Enhancing digital Television (ASSET) project[1,2] is to overcome the limitations of custom-specific implementation in a complete digital system for TV content creation. These limitations are due to lack of connectivity and interoperability between equipment and applications; lack of generalized middleware for multimedia content management and exchange with openly defined interfaces; and the use of different types of databases with respect to access and data model. Technical solutions available on the market are integrated or proprietary and are usually expensive to develop. They are not sufficient in serving the requirements of end-users; in particular, broadcasters.

The ASSET project has defined and developed a software architecture and the corresponding technologies necessary for the unified management of digital TV content. This covers the complete operational workflow, including acquisition, creation, editing, control, storage, broadcasting, publishing, and archiving of digital TV content. The project therefore defines and/or implements (1) a harmonized data model and interfaces for accessing and manipulating the multimedia objects covering the essence (audio and video) and the associated metadata, based on existing standardization work; (2) a command and control system between the different ASSET components, as well as a middleware layer, based on distributed technology, in order to expose an interface for controlling the different ASSET devices and servers; and (3) a reliable way to interchange audiovisual program material, system, and descriptive metadata between the different ASSET components as an integral part of a production and content management system.

The ASSET project utilizes open standards and emerging concepts and technologies such as MXF;[3] standard data models for describing essence; XML;[4] and distributed system technologies for defining the

concept of an ASSET middleware. XML is used for exchanging command and control information, such as service identification and parameters, through a set of predefined application program interaces (APIs). MXF is the proposed solution for the exchange of content between ASSET frameworks, and the implementations shall follow the standard recommendations. However, although the project supports the use of MXF, the proposed architecture is agnostic of the coding format and does not exclude other solutions.

A prototype, based on a typical workflow in a newsroom platform, demonstrates and validates the benefit of the ASSET approach.

## ASSET Architecture

The ASSET architecture is based on a software framework composed of a set of three standard interfaces and protocols for applications and products working together in an integrated environment. Applications communicate with the framework using the ASSET Public API, which allows them to communicate with any other application; access the public services provided by the framework; and use additional functionalities implemented by third-party integrators as aggregated services.

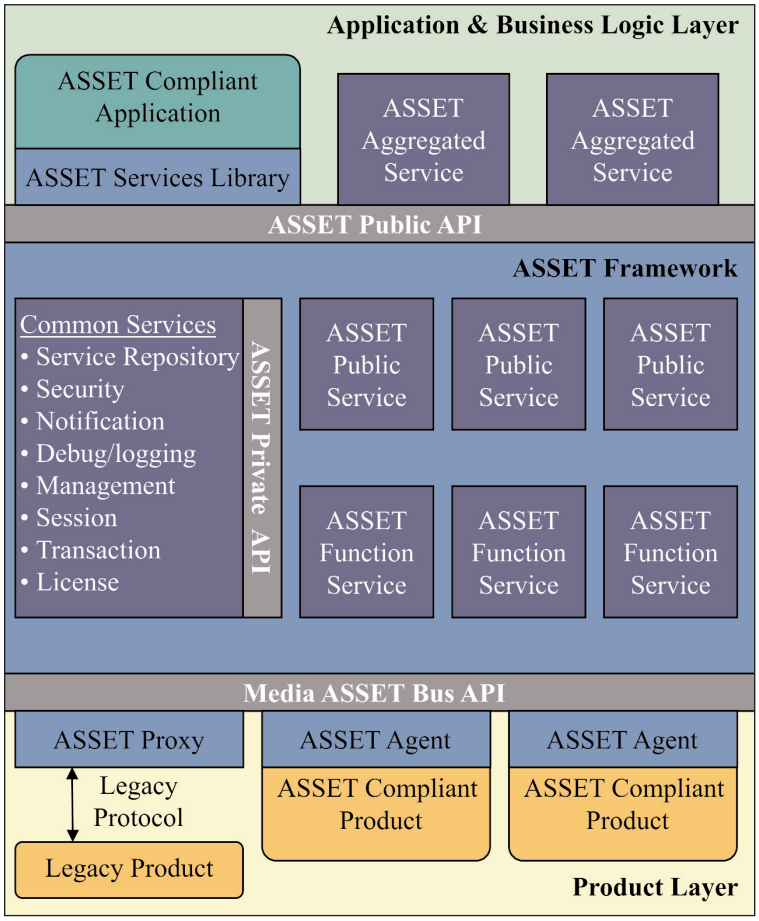


*Figure 1. ASSET architecture.*

Products from different manufacturers are integrated in the ASSET framework either natively, using an ASSET agent, or via an ASSET proxy. They are controlled and managed through the Media ASSET Bus API, based on XML service schema definitions. This API ensures an easy and seamless integration of devices and products in the framework.

An ASSET Private API is defined to access core services of the framework that enable the workability of this integrated environment. This API is only used internally in order to guarantee the overall system integrity. Figure 1 illustrates the different components of the ASSET architecture.

### *ASSET Components*

The ASSET architecture defines a number of concepts, components, and functions that enable the implementation of an ASSET-compliant framework. The core of the framework consists of three main components. The ASSET Public Services expose the mandatory services of the framework to the applications and to the aggregated services through the Public API. These services provide a minimum set of multimedia functionality, sufficiently rich and extensible in order not to limit the system efficiency. They ensure the consistency and integrity of the system by handling the internal logic such as access right, resource allocation, etc., required for each operation.

The ASSET Common Services provide implementation of key infrastructure requirements such as security, logging, notification, and resource management. This allows a uniform and single implementation of these services throughout the framework. They expose a private API that can only be used by ASSET Public Services.

The ASSET Function Services provide an abstraction of functionalities, such as encoder, recorder, and player, to the ASSET Public Services. They hide the specificities of the different interconnected products. For

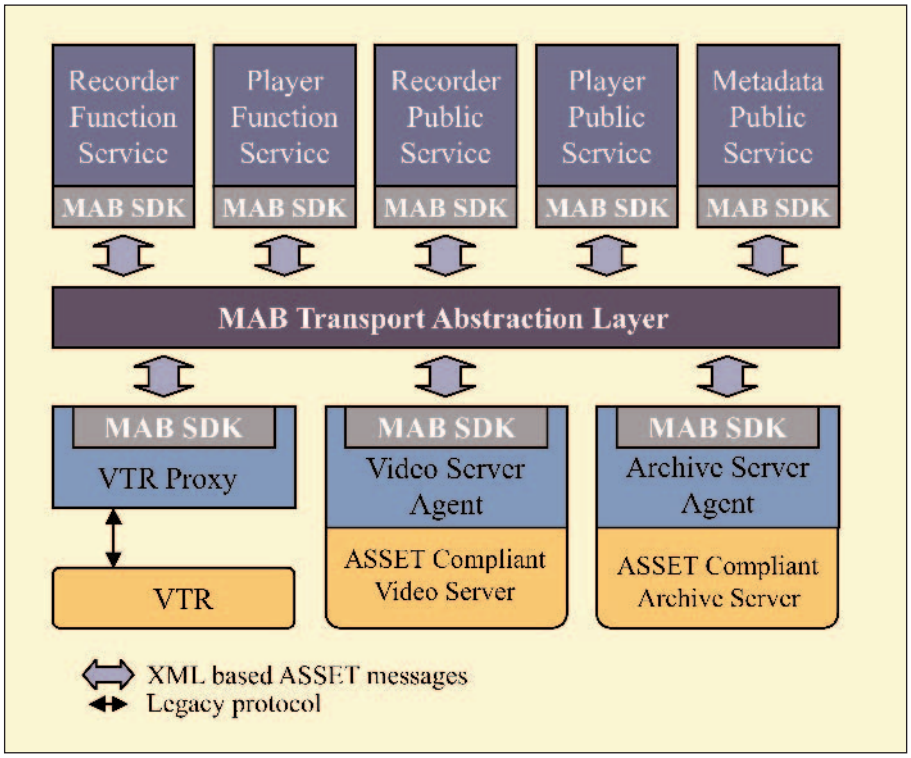


*Figure 2. Media ASSET Bus.*

example, for a public service, a VTR output and a video server output are considered as two system-wide logical output ports.

At the application and business logic layer, three components are introduced. The ASSET Compliant applications are the top-level ASSET software components. They use the Public API to access services provided by the framework and, optionally, by ASSET aggregated services.

The ASSET Services Library is a software component included in, or linked with, an application, which makes it compliant with the ASSET framework and gives it access to the ASSET public services. The ASSET Aggregated Services implement additional business logic on top of public services, or even other aggregated services. They register in the framework as new services available for ASSET-compliant applications. For example, complex workflows may be specified as aggregated services and then made available to other connected applications or aggregated services.

At the product layer, product is a manageable hardware or software component that implements one or several common functions (Most of the video server products implement a recorder function, a player function, and a storage function.) and logical components such as logical ports, repositories, and so forth. Products are compatible with the ASSET framework if (1) an ASSET-compliant product is managed by the framework through a built-in ASSET agent; or (2) a legacy product is not, or cannot have, a built-in ASSET agent. The ASSET framework can nonetheless manage such a product through an external software module called an ASSET Proxy. For example, a VTR is not capable of including a built-in ASSET agent and has to be connected through an ASSET Proxy.

### *Media ASSET Bus*

Most of the ASSET framework is built on top of a software bus called the Media ASSET Bus, or MAB. The goal of the MAB is to provide support for the integration of the widest range of products within the ASSET framework, independent of the underlying operating system environment and protocols.

The MAB defines a set of standard interfaces and synchronization processes that ensure a seamless interoperability between ASSET components and products. These interfaces allow any third-party media application or product to be integrated with the MAB by developing a simple software adapter in an agent or a proxy. A MAB software development kit (SDK) is provided in order to facilitate this task. It gives a uniform way of connecting devices, registering services, and exchanging messages within the ASSET framework.

The integrated environment offered by the MAB is based on a transport abstraction layer (TAL) that takes care of message exchanges. Messages have XML format, thus ensuring flexibility and adaptability of the ASSET solution. The concept of the Media ASSET Bus is illustrated in Fig. 2. Different ASSET components and products are interconnected via software adapters on top of the MAB SDK.

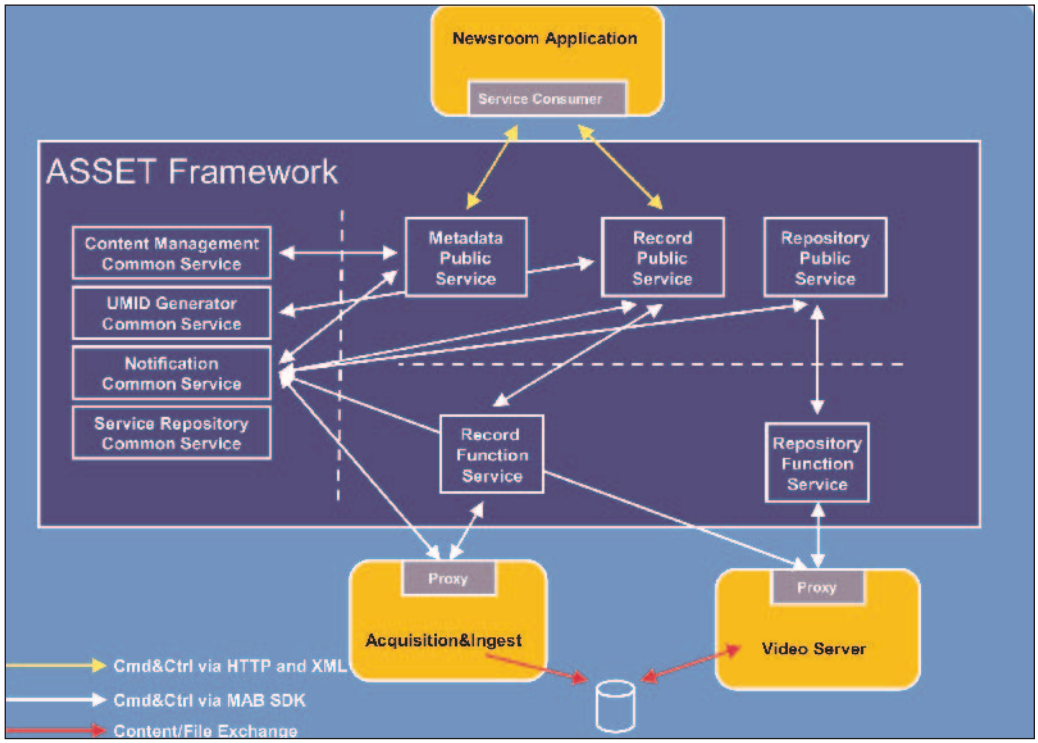


*Figure 3. ASSET demonstration platform components.*

## Demonstration Platform

The purpose of the demonstration platform, which is being implemented as part of the ASSET project, is to show the feasibility and simplicity of integrating different broadcast systems via the ASSET middleware. It should also serve as a reference system, where end users, administrators, and integrators can test and verify the implemented functionality against the requirements, thus validating the result of the project.

The demonstrator uses one client and two providers of different manufacturers to emulate a simple workflow of a news platform. The client is a newsroom application that controls the ingest process and lists the content stored on the online video server. The providers are an ingest server and an online storage system provided by a video server. Both the providers and the client application are connected to the ASSET middleware. All the command and control operations and data exchange are controlled by components of the ASSET middleware. Interaction between the different components used in the demonstrator is presented in Fig. 3.

### *Demonstration Platform Components*

The demonstration platform will make the following set of public, common, and function services available:

*Public Services:* (a) Record PS provides an API to start, stop, and restart the recording of the pre-programmed feeds. (b) Metadata PS is the central access point for accessing, creating, and modifying any kind of metadata in the system, such as content management metadata and descriptive metadata. (c) Repository PS provides the interface for the applications to access and modify content stored in any repository (online, nearline).

*Common Services:* (a) Service Repository CS provides a mechanism for common and function services to register to the framework so that they can be addressed and accessed by other services or applications. (b) Notification CS provides a subscription mechanism for other services and applications to be notified of events such as creation and modifications to objects. (c) UMID Generator CS provides a unique material identifier according to SMPTE 330M.[5] (d) Content Management Metadata CS maintains the link between a UMID and the repositories where the material is located.

*Function Services:* (a) Record FS provides an abstraction layer (entities, data exchanged) for controlling audio/video input ports. For example, sending different commands to control the recording of a media asset. In the demonstrator scenario, the Record PS uses the record function to control the input port of the acquisition server. (b) Repository FS provides an abstraction layer for controlling all the storage repositories of a system. The repository for the demonstrator is an online storage video server that stores the material ingested by the acquisition service.

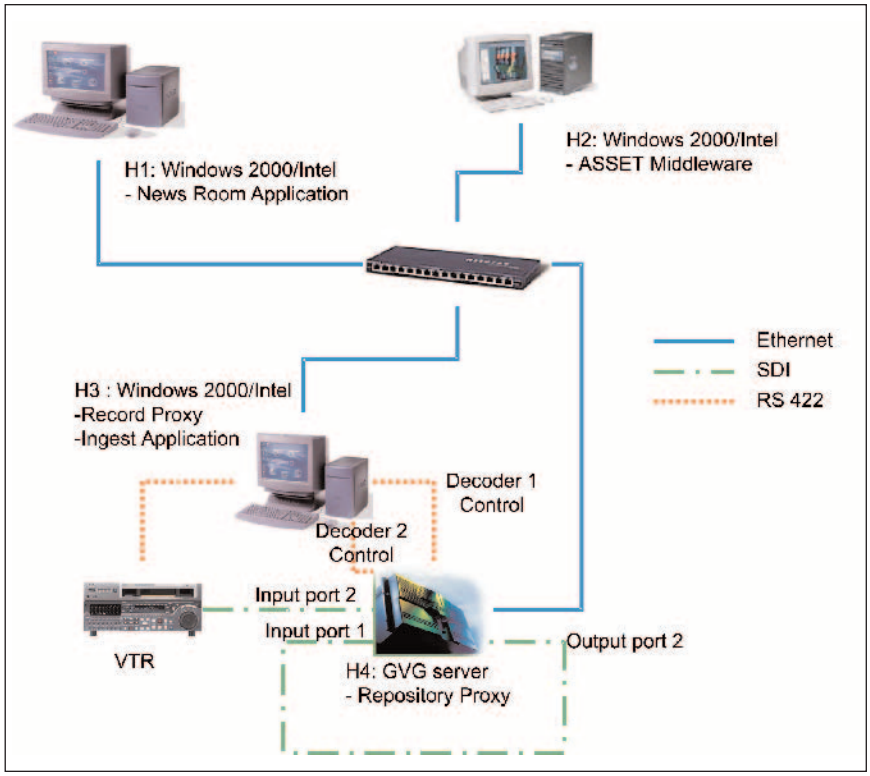


*Figure 4. ASSET network architecture for the demonstrator platform.*

## Demonstration Hardware Architecture

The demonstrator components are connected using different network technologies as presented in Fig. 4. The newsroom application displays the status of the ingest operation and lists all content stored on the online video server. Incoming material (A/V feeds) are digitized with different technical qualities. High-resolution copies are recorded on the video server. Each time a recording is initiated, a new and unique UMID will be associated with the new feed. The video server informs the ASSET framework about the new content creation and status changes.

## Register Record and Repository Functions—Use Case

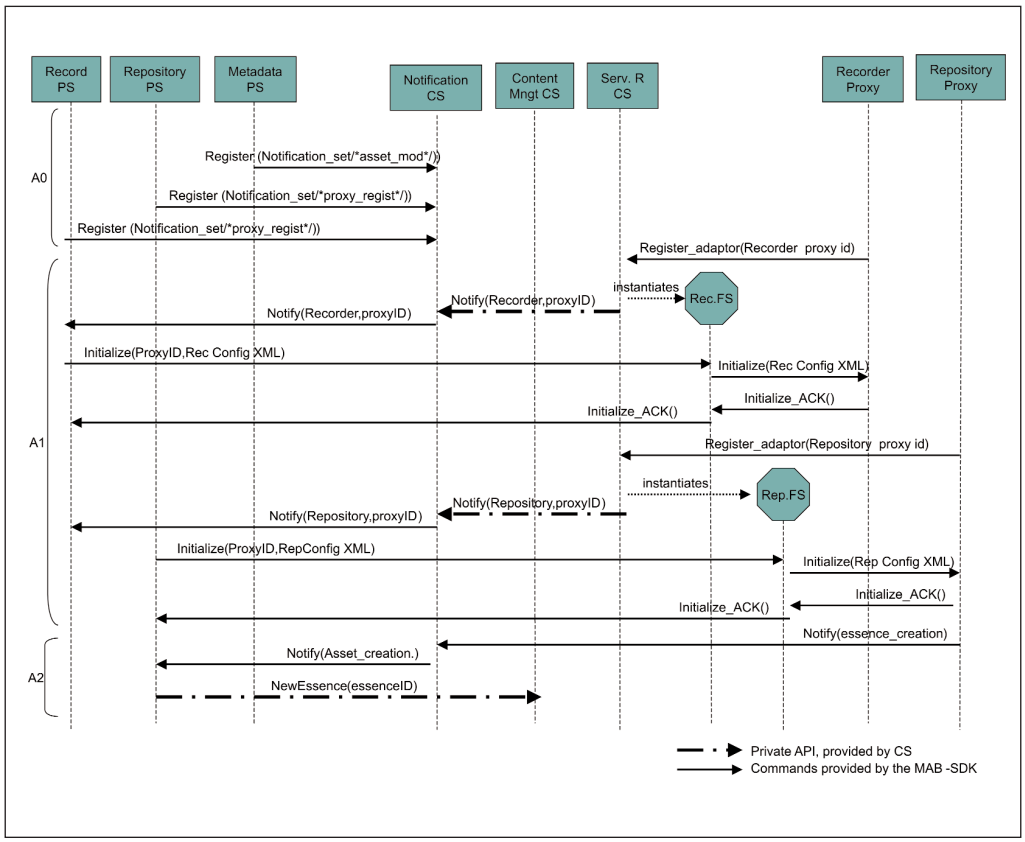


*Figure 5. ASSET demonstration registration scenario.*

Some use cases were developed to model the behavior and the interaction between different modules in the demonstrator. Figure 5 presents the Record and Repository FS registration into the ASSET framework. The following steps are executed:

A0—All the PS subscribe for notifications to the Notification CS in order to receive notifications in which they are interested: the Metadata PS to receive all modifications regarding the creation, deletion, move, and modification of content; and the Record and Repository PS to receive notifications about the

corresponding newly registered functions.

A1—The Record and Repository proxies are registered to the ASSET framework. The Service Repository CS creates the corresponding function services acting as client stubs for the proxies. After that, the corresponding PS are notified about the new registrations through the Notification CS. Generally, the public services would read the predefined configuration from the Configuration Management CS. In the demonstrator scenario, the configuration for the proxies are stored in configuration files read by the corresponding PS. The PS initializes the corresponding proxy by calling the function services' Initialize command, passing the configuration in the XML data parameter.

A2—After its registration, the Repository proxy sends a notification describing its content. This notification is received by the Metadata PS, which updates the Content Management CS information regarding the content location (repository) and media assets structures.

## Conclusion

This paper gives an overview of the work developed within the scope of the ASSET project. The software architecture and the main concepts are described, and a simplified prototype that is expected to test and validate the project approach is presented. This work is expected to improve the interconnection between equipment and media applications in digital TV environments, solving some of the problems users and system integrators currently face.

## References

1. IST-2001-37379—Architectural Solutions for Services Enhancing digital Television, http://mog.inescporto.pt/ist-asset.
2. Paula Viana et al., "A Unified Solution for the Integration of Media Applications and Products in Broadcaster Environments—The ASSET Architecture," *Proc. of the NAB Conference 2003*, pp. 172-176, April 2003.
3. Proposed SMPTE Standard, 377M-2003, "Material Exchange Format (MXF) File Format Specification," www.smpte.org.
4. W3C, Extensible Markup Language (XML) 1.0 (Second Edition), W3C Recommendation, World Wide Consortium, www.w3c.org/TR/REC-xml, Oct. 2000.
5. SMPTE 330M-2000, "Unique Material Identifier (UMID)," www.smpte.org.

*First published in the IBC2003 Conference Proceedings, Amsterdam, The Netherlands, Sept. 11-15, 2003.* 

## THE AUTHORS

**Mário Cordeiro** graduated with a degree in electrical and computer engineering from the University of Porto, Portugal, in 2000. He was a researcher at INESC Porto from September 2000 to April 2003, where he worked on several projects in the area of distributed systems, graphical user interfaces, and MXF. In May 2003, Cordeiro joined MOG Solutions, where he is currently developing middleware services and graphical user interfaces.

**Paula Viana** (pviana@inescporto.pt) received B.S. and M.Sc. degrees in electrical and computer engineering from the University of Porto, Portugal, in 1989 and 1994, respectively. She is an adjunct professor at the Polytechnic Institute of Porto, where she teaches courses in telecommunications. She is also a senior researcher in the telecommunications and multimedia unit at INESC Porto. Viana's main research activities and interests are in the area of networked audiovisual systems.

**José Ruela** (jruela@inescporto.pt) received a degree in electrical engineering from the University of Porto, Portugal, in 1971, and a Ph.D. in electrical engineering from the University of Sussex, U.K., in 1982. He is an associate professor at the University of Porto, where he teaches courses in data communications and computer networks. He is manager of the telecommunications and multimedia unit at INESC Porto, an R&D institute affiliated with the University of Porto. His main research interests are in resource management, quality of service, and performance evaluation in high-speed networks.

### ASSET Consortium

The work presented in this paper was developed, from June 2002 to September 2003, in the framework of ASSET (Architectural Solutions for Services Enhancing digital TV), a European-funded project (IST-2001-37379). The ASSET partners are Compaq-HP Group, France (D. Bommart); Thomson Broadcast Systems, France (S. Daulard); Dalet a.n.n., Germany (W. Bernet, E. Müller); INESC Porto, Portugal (M. Cordeiro, P. Viana, J. Ruela); INRIA, France (M. Body, B. Cousin); Institut für Rundfunktechnik, Germany (I. Höntsch, T. Marx); Front Porch Digital Inc., France (M. Laurentin, B. Algayres); SHS Multimedia, Italy (G. Ferrari, M. Strambini).