



**DAVIC 1.4.1. Specification Part 3**

**Service Provider System Architecture and Interfaces**

**(Technical Report)**

**NOTICE**

Use of the technologies described in this specification may infringe patents, copyrights or intellectual property rights of DAVIC Members or non-members.

Neither DAVIC nor any of its Members accept any responsibility whatsoever for damages or liability, direct or consequential, which may result from use of this specification.

© Digital Audio-Visual Council 1998.  
Published by Digital Audio-Visual Council  
Geneva, Switzerland

## CONTENTS

<i>Foreword</i> .....	<i>v</i>
<i>Introduction</i> .....	<i>vi</i>
<b>1. Scope</b> .....	<b>1</b>
<b>2. References</b> .....	<b>1</b>
<b>2.1 Normative references</b> .....	<b>1</b>
<b>2.2 Informative references</b> .....	<b>1</b>
<b>3. Definitions</b> .....	<b>2</b>
<b>4. Acronyms and abbreviations</b> .....	<b>2</b>
<b>5. Conventions</b> .....	<b>2</b>
<b>6. Architecture</b> .....	<b>3</b>
<b>6.1 The Server Reference Model</b> .....	<b>3</b>
<b>6.2 Broadcast Server</b> .....	<b>5</b>
6.2.1 Broadcast Server Playlist Management .....	5
6.2.2 Broadcast Server Timing Requirements .....	6
<b>6.3 Protocol Profiles</b> .....	<b>6</b>
<b>6.4 Domains</b> .....	<b>6</b>
<b>6.5 Internal Server Architecture</b> .....	<b>7</b>
<b>7. Service Interfaces</b> .....	<b>7</b>
<b>8. Service Elements</b> .....	<b>8</b>
<b>8.1 Service Element Groups</b> .....	<b>8</b>
8.1.1 Application Group .....	9
8.1.2 Content Group .....	9
8.1.3 System Group .....	10
<b>8.2 Service Element Descriptions</b> .....	<b>10</b>
8.2.1 Application Group .....	10
8.2.2 Content Group .....	12
8.2.3 System Group .....	15
<b>9. Networked Server Objects</b> .....	<b>16</b>
<b>Annex A Conceptual Server Model (Informative)</b> .....	<b>19</b>
<b>A.1. Requirements</b> .....	<b>19</b>
<b>A.2. Advantages of an Open Server Model</b> .....	<b>19</b>
<b>A.3. Basic Concepts</b> .....	<b>20</b>
<b>A.4. Standard Service Element</b> .....	<b>21</b>
A.4.1. Control Plane .....	22
A.4.2. User Plane .....	22
A.4.3. Management Plane .....	22
<b>A.5. Standard Service Element Functionality, Interfaces, and APIs</b> .....	<b>22</b>
A.5.1. The Standard Service Element interfaces .....	22
<b>A.6. Server Reference Model Service Elements</b> .....	<b>22</b>
<b>A.7. The Layered Communications Model</b> .....	<b>23</b>

<b>A.7.1.</b> Layer 5 Protocols.....	23
<b>A.8.</b> Network Implementations of a Service Provider System .....	24
<i>Annex B Service Provider Instance (Informative)</i> .....	25
<i>Annex C Content Provision (Informative)</i> .....	26
<b>C.1.</b> Scope.....	26
<b>C.2.</b> Definitions.....	26
<b>C.3.</b> Content Installation and Management Objectives.....	26
<b>C.4.</b> Supply Chain .....	27
C.4.1. Real Time Distribution .....	27
C.4.2. Non-Real Time Distribution .....	28
C.4.3. Multicast Distribution .....	28
<b>C.5.</b> The Content Package .....	29
<b>C.6.</b> The Service Package Installation Interface .....	31
<b>C.7.</b> Loading Content onto the Server from Service Provider .....	31
C.7.1. Methods of Loading.....	31
C.7.2. Version Control.....	32
<b>C.8.</b> The Content Server Element.....	32
C.8.1. Content Element .....	32
<i>Annex D VoD Scenario (Informative)</i> .....	34

## Foreword

The Digital Audio-Visual Council (DAVIC) is a non-profit Association registered in Geneva in 1994. The objective of DAVIC is to promote the success of interactive digital audio-visual applications and services through specification of open interfaces and protocols.

The DAVIC 1.4 Specification was developed by representatives of DAVIC member organizations. It is a public document based on submissions from members and non-members in response to the public Calls For Proposals which were issued in October 1994, March 1995, September 1995, December 1995, March 1996 and September 1996. The specification has full backward compatibility with the earlier versions [DAVIC 1.0](#), [DAVIC 1.1](#) and [DAVIC 1.2](#). [DAVIC 1.3](#) has been available on the Internet since June 1997. [DAVIC 1.3.1](#) is a point release issued in Milan, March 1998.

DAVIC 1.4 is a single specification consisting of 14 parts.

- [Part 1: Description of Digital Audio-Visual Functionalities](#)
- [Part 2: System Reference Models and Scenarios](#)
- [Part 3: Service Provider System Architecture](#)
- [Part 4: Delivery System Architecture and Interfaces](#)
- [Part 5: Service Consumer System Architecture](#)
- [Part 6: Management Architecture and Protocols](#)
- [Part 7: High And Mid-Layer Protocols](#)
- [Part 8: Lower-Layer Protocols and Physical Interfaces](#)
- [Part 9: Information Representation](#)
- [Part 10: Basic Security Tools](#)
- [Part 11: Usage Information Protocols](#)
- [Part 12: System Dynamics, Scenarios and Protocol Requirements](#)
- [Part 13: Conformance and Interoperability](#)
- [Part 14: Contours: Technology Domain](#)

The DAVIC PAS (Publicly Available Specification) forwarded to ISO/IEC JTC 1 for transposition into an international standard is a subset of the DAVIC 1.4 specification consisting of the normative parts (2, 6-12 and 14). In addition, the essential informative Part 1 which provides categorised sets of user and market requirements and has two informative annexes which are closely integrated with the normative technology conformance details provided in Part 14 is proposed as an ISO/IEC JTC 1 Technical Report.

All versions and corrigenda of DAVIC specifications are available from the DAVIC web site.

## Contact Information

DAVIC Secretariat  
c/o Societa' Italiana Avionica Spa  
Strada Antica di Collegno, 253  
I-10146 Torino - Italy  
Tel.: +39 11 7720 114  
Fax: +39 11 725 679  
Email: [secretariat@davic.org](mailto:secretariat@davic.org)  
DAVIC Home Page: <http://www.davic.org>

## Introduction

DAVIC specifications define the minimum tools and dynamic behavior required by digital audio-visual systems for end-to-end interoperability across countries, applications and services. To achieve this interoperability, DAVIC specifications define the technologies and information flows to be used within and between the major components of generic digital audio-visual systems. Interoperability between these components and between individual sub-systems is assured through specification of tools and specification of dynamic systems behavior at defined reference points. A reference point can comprise one or more logical (non-physical) information-transfer interfaces, and one or more physical signal-transfer interfaces. A logical interface is defined by a set of information flows and associated protocol stacks. A physical interface is an external interface and is fully defined by its physical and electrical characteristics. Accessible reference points are used to determine and demonstrate compliance of a digital audio-visual subsystem with a DAVIC specification.

The Parts of the DAVIC 1.4 Specification can be classified into four major groups. A summary of each part under each of the four headings follows.

### Requirements and Framework (Parts 1-2)

Part 1 provides a detailed listing of the functionalities required by users and providers of digital audio-visual applications and systems. It introduces the concept of a contour and defines the IDB (Interactive Digital Broadcast), EDB (Enhanced Digital Broadcast), and Institutional Multimedia Retrieval (IMR) functionality requirements which are used to define the normative contour technology toolsets provided in Part 14.

Part 2 defines the normative DAVIC technical framework. It provides a vocabulary and a Systems Reference Model, which identifies specific functional blocks and information flows, interfaces and reference points.

### Architectural Guides (Parts 3-5)

Parts 3, 4 and 5 are technical reports which provide additional architectural and other information for the server, the delivery-system, and the service consumer systems respectively.

Part 3 defines how to load an application, once created, onto a server and gives information and guidance on the protocols transmitted from the set-top user to the server, and those used to control the set-up and execution of a selected application.

Part 4 provides an overview of delivery systems and describes instances of specific DAVIC networked service architectures. These include physical and wireless networks. Non-networked delivery (e.g. local storage physical media like discs, tapes and CD-ROMs) are not specified.

Part 5 provides a service consumer systems architecture and a description of the DAVIC Set Top reference points defined elsewhere in the normative parts of the specification.

### Technology Toolsets (Parts 6-11)

The next six parts are normative. They specify and comprise the technology toolsets and relevant protocols across the entire audio-visual creation and delivery chain.

Part 6 specifies the information system model used for managing DAVIC systems. In particular, this part defines the managed object classes and their associated characteristics for managing the access network and service-related data in the delivery system. Where these definitions are taken from existing standards, full reference to the required standards is provided. Otherwise a full description is integrated in the text of this part. Usage-related information model is defined in Part 11.

Part 7 defines the technologies used for high and mid-layer protocols for DAVIC systems. In particular, this part defines the specific protocol stacks and requirements on protocols at specific interfaces for the DAVIC content, control and management information flows.

Part 8 defines the toolbox of technologies used for lower layer protocols and physical interfaces. The tools specified are those required to digitize signals and information in the Core Network and in the Access Network. Each tool is applicable at one or more of the reference points specified within the delivery system. In addition a detailed specification is provided of the physical interfaces between the Network Interface Unit and the Set Top Unit and of the physical interfaces used to connect Set Top Boxes to various peripheral devices (digital video

recorder, PC, printer). The physical delivery system mechanisms included are copper pairs, coaxial cable, fiber, HFC, MMDS, LMDS, satellite and terrestrial broadcasting.

Part 9 defines what the user will eventually see and hear and with what quality. It specifies the way in which monomedia and multimedia information types are coded and exchanged. This includes the definition of a virtual machine and a set of APIs to support interoperable exchange of program code. Interoperability of applications is achieved, without specifying the internal design of a set top unit, by a normative Reference Decoder Model which defines specific memory and behavior constraints for content decoding. Separate profiles are defined for different sets of multimedia components.

Part 10 defines the interfaces and the security tools required for a DAVIC 1.4 system implementing security profiles. These tools include security protocols which operate across one or both of the defined conditional access interfaces CA0 and CA1. The interface CA0 is to all security and conditional access functions, including the high speed descrambling functions. The interface CA1 is to a tamper resistant device used for low speed cryptographic processing. This cryptographic processing function is implemented in a smart card.

Part 11 specifies the interface requirements and defines the formats for the collection of usage data used for billing, and other business-related operations such as customer profile maintenance. It also specifies the protocols for the transfer of Usage Information into and out of the DAVIC System. In summary, flows of audio, video and audio-visual works are monitored at defined usage data collection elements (e.g. servers, elements of the delivery system, set-top boxes). Information concerning these flows is then collected, processed and passed to external systems such as billing or a rights administration society via a standardised usage data transfer interface.

### **Systems Integration, Implementation and Conformance (Parts 12-14)**

Part 12 is a normative part which defines system dynamic behavior and physical scenarios. It details the locations of the control functional entities along with the normative protocols needed to support the systems behavior. It is structured as a set of protocol walk-throughs, or “Application Notes”, that rehearse both the steady state and dynamic operation of the system at relevant reference points using specified protocols. Detailed dynamics are given for the following scenarios: video on demand, switched video broadcast, interactive broadcast, and internet access.

Part 13 is an informative report which provides guidelines on how to validate the systems, technology tools and protocols through conformance and / or interoperability testing.

Part 14 provides the normative definition of DAVIC Technology Contours. These are strict sets of Applications, Functionalities and Technologies which allow compliance and conformance criteria to be easily specified and assessed. DAVIC 1.4 contains the full details of three contours introduced in Part 1 of this Specification. Part 14 specifies required technologies and is a mandatory compliance document for contour implementations.

## Service Provider System Architecture and Interfaces

### 1. Scope

The DAVIC Service Provider System architecture and interfaces section describes the entities and interfaces of the server portion of the DAVIC system model. The server reference model defines the architecture for Content Service Provider and End-User Service Provider Systems. The Content Service Provider System supports common system services, a content transfer interface(A10) and repository for content. The End-User Service Provider System supports common system services, a content transfer interface(A10), a repository for content, an application runtime interface (A9), and services that support the functionalities of DAVIC applications such as movies on demand, teleshopping, etc.

This Part is organized as follows:

<b>Architecture</b>	This section describes partitionings for Server System, Security and Administrative Domains, End-User Service Provider System, Content Service Provider System.
<b>Service Interfaces</b>	This sections lists the normative DAVIC service interfaces that are supported by the Service Provider System.
<b>Service Elements</b>	This section describes DAVIC Service Elements, which represent implementations of the Service Interfaces.
<b>Networked Server Objects</b>	This section illustrates the interaction of Client and Service Elements for the purpose of Stream control and Download.
<b>Annex A: Conceptual Server Model</b>	This section presents a conceptual basis for the Service Element inheritance model.
<b>Annex B: Service Provider Instance</b>	This section illustrates an example Service Provider System instance using four core Service Elements.
<b>Annex C: Content Provision</b>	This section describes the flow of content from source to End-User Service Provider, and presents general Content and Service administration concepts.
<b>Annex D: Scenarios</b>	This section illustrated server-related message sequences for selected DAVIC applications.

The Service Provider System architecture refers to server-related information that is described in the following sections of the [DAVIC 1.4](#) Specification:

- Part 7: High Layer and Mid-Layer Protocols. The Service Element interfaces are defined in Part 7.
- Part 9: Information Representation. The file format for content transfer is defined in Part 9.
- Part 12: Dynamics, Reference points and interfaces for A9 (server to delivery system). Scenarios and message sequences are described in Part 12.

### 2. References

#### 2.1 Normative references

Part 3 of the [DAVIC 1.4](#) specification is not normative and the references used in this part cannot be regarded as normative unless they are specified as such in other, normative parts.

#### 2.2 Informative references

The following documents contain provisions which, through reference in this text, constitute provisions of this Specification. At the time of publication, the editions indicated were valid. All referenced documents are subject to revision, and parties to agreements based on this Specification are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards. The TSB (Telecommunication Standardization Bureau) maintains a list of currently valid ITU-T Recommendations.

1. ISO/IEC 13818-6: Information Technology - Generic coding of moving pictures and associated audio information: Part 6 - Extensions for Digital Storage Media Command and Control (DSM-CC).
2. ISO/IEC 14750 Interface Definition Language (IDL).
3. Common Object Request Broker: Architecture and Specification, version 2.1 August 1997 (Note: known as OMG CORBA 2.1)  
available at <ftp://ftp.omg.org/pub/docs/formal/97-09-01.pdf>
4. IETF RFC 1305 Network Time Protocol version 3.
5. US Coastguard Global Positioning System.

### 3. Definitions

This Section defines new terms, and the intended meaning of certain common terms used in this Specification. Part 2 [Annex A](#) defines additional terms and, in some cases, alternative interpretations that are appropriate in other contexts. The definitions in the annex were derived from various sources: some are direct quotes, others have been modified.

Supplementary definitions in Part 2 [Annex A](#) are not normative and are provided for reference purposes only. (For convenience, copies of the normative definitions below are included in the annex.)

There are no normative definitions for this part.

### 4. Acronyms and abbreviations

Part 2 [Annex A](#), B and C contain a complete set of acronyms and abbreviations used throughout the [DAVIC 1.4](#) Specification. The following acronyms and abbreviations are used in this Specification:

ASI	Asynchronous Serial Interface
CA	Conditional Access
CORBA	Common Object Request Broker Architecture
DSM-CC	Digital Storage Media Command & Control
DVB	Digital Video Broadcast (European standards body)
ECM	Entitlement Control Message
EMM	Entitlement Management Message
GPS	Global Positioning System
IDL	Interface Definition Language
IOP	Inter-ORB Protocol
IIOB	Internet Inter-ORB Protocol
NTP	Network Time Protocol
ORB	Object Request Broker
RPC	Remote Procedure Call
SI	Service Information

Also refer to Part 2 of [DAVIC 1.4](#).

### 5. Conventions

The style of this Specification follows the general guidelines of ISO/IEC 0001 : 1993: Information Technology Rules For Presentation Of ITU-T | ISO/IEC Common Text.

## 6. Architecture

### 6.1 The Server Reference Model

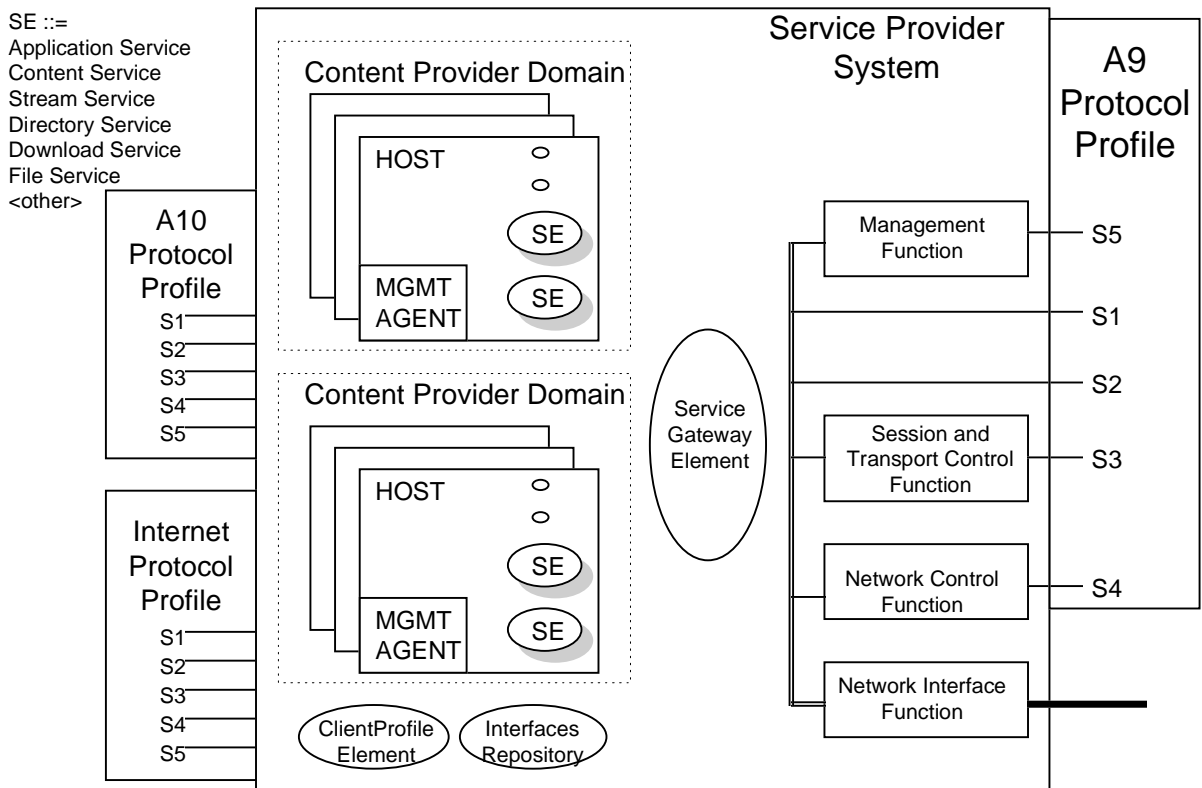


Figure 6.1-1 Distributed Server Reference Model

The Distributed Service Provider System specification is defined as a Domain containing a scaleable number of DAVIC Service Elements(SE) and a scaleable number of physical hosts. A common set of Service Elements apply to the Service Provider Domain as a whole. These are the ServiceGateway, the InterfacesRepository, and the ClientProfile elements. The ServiceGateway is the only entry point into the Service Provider System by an outside client. Entry is gained by invocation of the ServiceGateway attach() operation. The Interfaces Repository ensures the coherence and completeness of all public interfaces and data types in the Service Provider System. The ClientProfile Element is a repository for the configurations of clients that may access this system.

In addition, a common set of functions apply to Service Provider Domain as a whole. These are the Session Control Function (SessionGateway) on the S3 interface, the Network Services Function on the S4 interface, and the Management Function on the S5 interface.

All other Service Elements are allocated to sub-Domains. These are called Content Provider Domains. The Content Provider is the owner of the domain, and is allotted a percentage of the Service Provider System resources, e.g., a percentage of storage, computational and port bandwidth capacity of the system.

The model is likened to a department store, where the manager provides basic facilities, floor space for the stores within, and a directory at the entry. The store tenants are free to setup their wares in whatever way they best see fit, in the allotted floor space.

The Domain Concept is described in the CORBA 2.1 specification of the OMG. In the case of the DAVIC Service Provider System, the domain is both a security domain and an administrative domain. It is a security domain in that authentication and authorization is required at the ServiceGateway entry point. It is an administrative domain in that a Management Function applies to the system as a whole. The Service Provider domain also offers a common name graph, where all Service Elements are nodes

which may be traced from the ServiceGateway using CORBA logical path names (CosNaming::Names).

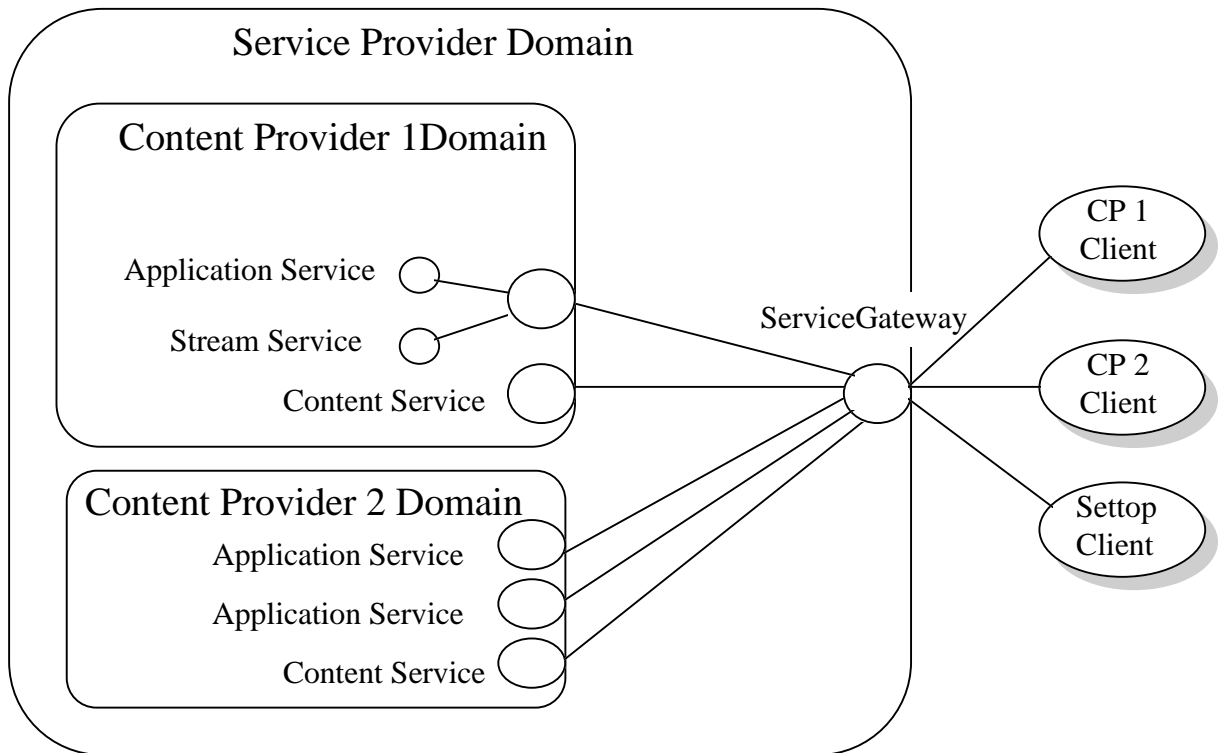


Figure 6.1-2: Service Provider and Content Provider Domains

Each Content Provider in the system is given a sub-domain within which to operate. The Content Provider may elect to setup this sub-domain as a security domain. The Service Provider may also act in the capacity of one of the Content Providers of the system, thus using up a percentage of the system resources in order to offer application services to end-users. For example, the Service Provider acting as a Content Provider may offer Movies-on-Demand using allotted resources of the system, and another Content Provider may offer Home Shopping. The Content Provider may create multiple Application Service Elements or Content Service Elements and bind these by name with the ServiceGateway. The Content Provider may create Content Provider sub-Domains where Other Content Providers may have access. Access rights to Domains is determined by the manager of the Domain, through the Domain, Group and Member APIs. Thus a Content Provider can grant access to other Content Providers or clients. The Domain architecture with Access Roles and Access Privileges enables a wide-variety of business models for the arrangement of contracts between Service Providers and Content Providers, between Content Providers and other Content Providers.

The Content Provider domain does not necessarily align with host boundaries, i.e. the capacity of a host may be allotted to more than one Content Provider.

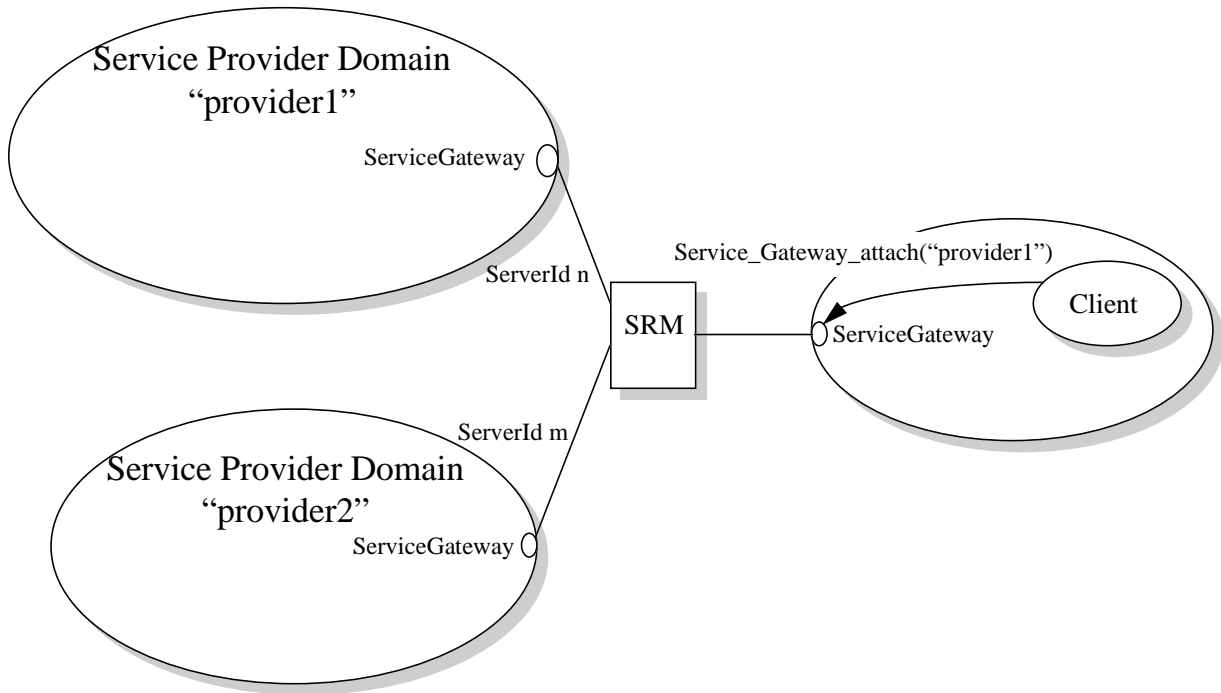


Figure 6.1-3 : Access to Service Provider Domains

Service Domains are linked by DSM-CC Session and Resource Manager. A common name space permits navigation between multiple Service Domains. The DSM-CC ServiceGateway attach() or Session attach() is used by clients to enter a new Service Domain.

## 6.2 Broadcast Server

This logical entity is an extension of the general Distributed Server Reference Model which adds the ability to support playlists for accurate control of the play out of concatenated Service Elements. It therefore inherits the functionality of the Service Gateway from the Distributed Server and extends it to support the playlist functionality.

It also encapsulates other specific Service Elements useful in the broadcast case. These are targeted at supporting the need for Multi-Program Transport Streams (MPTS) used in unidirectional broadcast environments (Transport Stream Multiplexor), provide a real-time Stream Service Element that converts one form of real-time incoming data into a standard S1 flow and another specific extended type of Stream Service Element that plays out data carousels. These are not new entities to DAVIC specifications, but merely explicit instances of generic Service Element types already defined.

### 6.2.1 Broadcast Server Playlist Management

Two extensions to the generic Distributed Server are required to support the concept of a playlist, required for the continuous service nature of broadcast delivery systems:

1. A Playlist Client maintains a schedule of service element launch points and signals the Broadcast Server when the timing so requires. This element is able to communicate with the service elements from within the Broadcast Server across the A9\* reference point, from outside the Broadcast Server but directly with it across A9, or via the core network. i.e. Its location in the reference model is not explicitly defined. The exact nature of its operation and the message set used to interact with the Broadcast Server is not defined in DAVIC 1.3.
2. A Playlist Server is the other half of the process and is an extension of the Service Gateway Element that acts as a repository for the list of Service Elements to be delivered contiguously once provided by the Playlist Client.

### 6.2.2 Broadcast Server Timing Requirements

Since SPS entities and their sub-elements may be locally interconnected using LAN type technologies, or remotely distributed in a WAN-like arrangement, two different network time strategies are defined. For locally interconnected machines, Network Time Protocol (NTP version 3 - IETF RFC 1305) is used to ensure synchronization of all locally connected elements to a local time server. For physically diverse systems, Global Positioning System (GPS) timers should be used to synchronize the time servers.

The accuracy of the time maintained on all elements of the system that are required to have any knowledge of the time of day should be within  $\pm 1$ ms of GPS time.

### 6.3 Protocol Profiles

A Protocol Profile defines the complete network stack used for interfacing between networked objects. The Protocol Profile is specified in the CORBA 2.1 specification of the OMG. The DAVIC A9 reference point using ATM is one such Protocol Profile. An A10 interface (for instance with satellite communications) is another such Protocol Profile. Various Internet Protocol Profiles also exist, which enable Internet access by the Service Element instances. Due to the possible existence of multiple Protocol Profiles, it may be necessary to implement bridges or half-bridges between the Protocol Profiles as is described in the CORBA specification.

### 6.4 Domains

In a DAVIC distributed environment, domains are used to encapsulate the networked objects in the system. DAVIC Domains have the following characteristics:

Each Domain will have members identified by a unique Principal Id, and groups categorized as reader, writer, broker or manager. These categories correspond to DSM-CC permission groups in the DSM::Access class. New members can be added to the Domain, a member can be added to a Group, and members can be granted privileges.

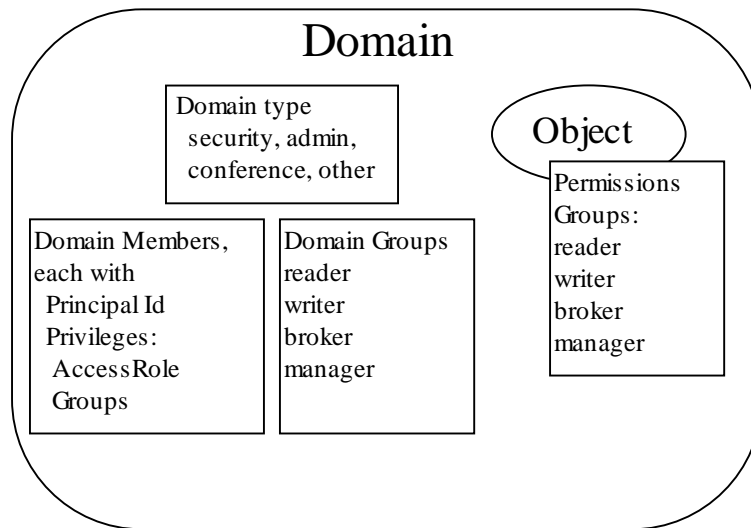


Figure 6.4-1: Domain Attributes and Interfaces

Each object in the Domain shall have the DSM-CC Access Interface, which specifies a permissions attribute identifying which reader, writer, broker or manager groups have access to the object. Furthermore, each operation of the object specifies whether reader, writer, broker or manager privileges are required to invoke that operation.

Interfaces are specified in [DAVIC 1.4](#) for Domains, Groups in Domains, and Members in Domains. The operations of these interfaces enable the Domain itself and all objects in the Domain to control access and operation invocation rights.

### 6.5 Internal Server Architecture

Part 2 of the [DAVIC 1.4](#) specification defines the A9\* reference point - an interconnection interface defined to allow for consistent interoperability between various hosts and Service Elements in a Distributed Server. It is called A9\* as it is very similar in functionality and operation to the A9 reference point where the Distributed Server interfaces to the Core Network and/or Access Network. It has some added functionality, however, that is common to Distributed Server implementations but not common in Core Networks or Access Networks. In particular, support is added for 10/100BaseT Ethernet connections, DVB ASI as well as the ubiquitous ATM (these are all defined fully in Part 8). However, the general aim is to ensure that consistent information flows, protocol stacks and physical interfaces are available for interconnection of various parts of the Distributed Server.

While the diagrams above all show logical architectures, a physical architecture for the Distributed Server is shown in *Figure 6.5-1: Distributed Server - Physical Architecture* below. This indicates the presence of some of the service elements particular to the Broadcast Server as well as the general Distributed Server and the ability to incorporate a Transport Multiplexor that can provide an MPTS for transfer across A11/A10, A9\* and A9 on S1 flows. It also shows the concept of cascading instances of a SPS with multiple A11/A10 interfaces linking them for a more generalized and complex instance of a Distributed Server.

Extensions to the S1 flow are provided to support a lower cost physical interface network than ATM for non-video/audio streams. This will allow low bandwidth server hosts/service elements (e.g. ECM server, SI servers, etc.) to provide their TS packets to a Transport Stream re-multiplexor without the need for an ATM network or separate physical point to point links.

This network is defined for use across the A9\* reference point as well as on A11/A10 where there is no ATM Core network separating the paired SPS entities. In particular, Ethernet (10BaseT and 100BaseT) is supported and uses the UDP/IP stack.

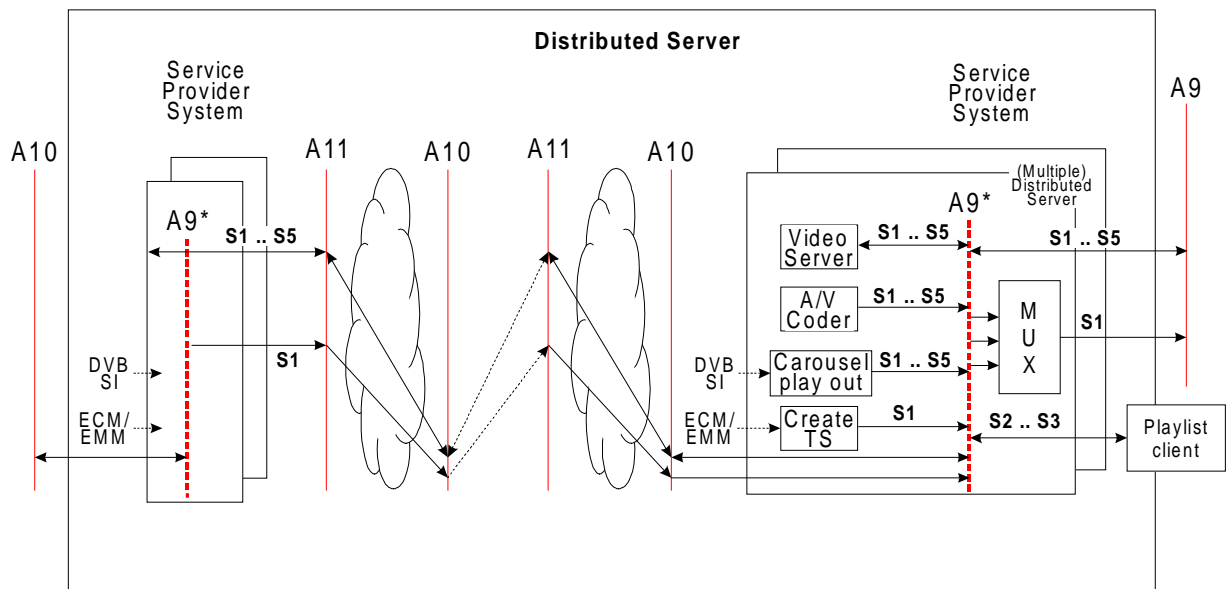


Figure 6.5-1: Distributed Server - Physical Architecture

### 7. Service Interfaces

In DAVIC, a Service Element represents a specialized active entity, such as a process, that is implemented at the Server. The Service Element can receive requests over its declared interface. The service interface specifies a group of operations, each of which may be invoked over the DAVIC S2

interface. The service interface may declare inheritance of other interfaces, thus expanding its operation set.

[DAVIC 1.4 Part 7](#) references as normative the following service interfaces from DSM-CC:

DSM::Base

DSM::Access

DSM::Stream

DSM::Event

DSM::Directory

CosNaming::NamingContext

CosNaming::BindingIterator

DSM::File

DSM::Session

DSM::ServiceGateway

DSM::Download

DSM::Interfaces

DSM::Service

DSM::SessionGateway

[DAVIC 1.4 Part 7](#) defines as normative the following service interfaces:

DAV::Stream

DAV::Domain

DAV::Group

DAV::Member

DAV::Content

The definition or references for the APIs of the above interfaces is specified in Part 7.

In addition, DAVIC implementations may optionally use other DSM-CC User-to-User interfaces, and declare new application-specific interfaces.

## 8. Service Elements

Service Elements run in the Service Provider system and provide interface(s) of operations that Clients can invoke. The Service Element communicates through one or more network stacks, e.g., DAVIC A9 or DAVIC A10 reference points. The Service Elements may therefore be grouped as residing in a major architectural partition. Some Service Elements, such as Directory, may reside in more than one major group.

### 8.1 Service Element Groups

The groupings are System, Application and Content. These groupings also define the partitioning of Service Elements for Content Service Provider and End-User Service Provider, as shown below:

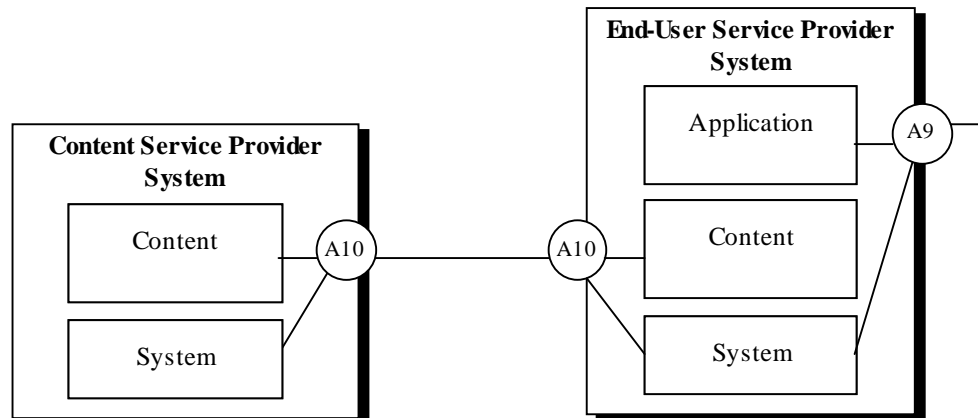


Figure 8.1-1: Application, Content and System Groups

### 8.1.1 Application Group

<b>Application Service</b>	The Application Service uses the A9 interface to the End-User Client at application runtime. It may inherit any of the application group Service interfaces. It may also define new application-specific interfaces and operations. It is a goal of DAVID to accommodate a wide range of applications and to give the information provider a vehicle by which new, value-added services may be created for DAVID systems.
<b>Stream Service</b>	The Stream Service provides the control interface for continuous media streams. The stream may be video, audio, or data. The DAVID Stream Service Element inherits DSM::Stream and DSM::Event.
<b>Create TS Service</b>	This special instance of the Stream Service Element is a real-time converter of incoming data into a Transport Stream compliant with the general S1 flow defined in the DAVID specifications. A particular example of this element may be a real-time audio/video encoder.
<b>Data Carousel Service</b>	The Data Carousel is a special instance of a Stream Service Element that provides for the repeat playout of regular data where such data is played out as an MPEG-2 Transport Stream compliant with the S1 flow definition..
<b>Download Service</b>	The Download Service allows the server to download operational or application code to the client
<b>Directory Service</b>	The Directory Service provides path name to service bindings. With the Directory Service, hierarchical organizations of Directories can be created, and Clients can browse and access Services. The Directory Service is inherited by both the ServiceGateway and Content Services.
<b>File Service</b>	The File Service provides read/write access to a File.

### 8.1.2 Content Group

<b>Content Service</b>	The Content Service uses the A10 interface for transfer and loading of Content. It is the repository of content for use by Application, Stream, Download and other Services. The Content Service inherits Directory Service functionality.
<b>Directory Service</b>	In the Content Group, the Directory Service provides path name to File and Directory bindings. With the Directory Service, hierarchical organizations of Directories can be created, and Server-side Clients can browse and access Files.
<b>File Service</b>	The File Service provides read/write access to a file. In the Content Group, the File provides static storage for use by Application, Stream, Download and other Services.

### 8.1.3 System Group

---

<b>Service Gateway</b>	The Service Gateway is a broker where services register to make their existence known and deregister when they are decommissioned. It is the means by which the client discovers the existence of a service. It is the means by which a client activates an instance of a service and deactivates an instance of a service. The ServiceGateway inherits the Directory interface.
<b>SessionGateway Service</b>	The SessionGateway Service performs the Session Control function Of the Server Reference Model. It enables a service or ServiceGateway to add and delete resources by invoking operations that will in turn negotiate resources with the Session Resource Manager (SRM) by calling U-N add and delete resource messages. A Session Gateway also performs translation between User-to-User RPCs and User-to-Network Messages (e.g. Session Establishment messages).
<b>Client Profile Service</b>	The Client Profile Service allows the server to maintain a profile of its clients to better assure that each service instance provided by the server matches the capabilities of its client.
<b>InterfaceRepository Service</b>	The InterfaceRepository Service insures completeness and consistency of all interfaces of the Service Provider Domain. It is used to define and verify new Service Element interfaces as they are added to the system.

---

## 8.2 Service Element Descriptions

The Service Elements are described here in terms of their DAVIC behavior. Where not described by DAVIC, Service Elements based on DSM-CC interfaces shall exhibit the behavior specified by DSM-CC.

### 8.2.1 Application Group

#### 8.2.1.1 Application Service Element

The Application Service is a general purpose service that enables applications to operate via the definition, manipulation, and exchange of objects over their lifecycle. By being a general facility, it accommodates a wide range of real applications, and is the key vehicle by which the DAVIC application functionalities (listed in Part 1 of [DAVIC 1.4](#)) are implemented. It gives the information provider a means by which new, value-added services may be created for DAVIC systems through instantiations and sub-classes of the Application Service Element.

A specific service inherits the DSM::Base class. The base class provides functions which all service objects should support. The functions often relate to the lifecycle. The two classic examples are:

- a) deletion of an object reference, which deletes the message connection to the object but not the object itself, and
- b) deletion of the object itself.

The Base class provides a common interface, with consistent semantics, to control the lifecycle.

The application service is the collection of interfaces on which all applications and some other services build. Since each service will add application-specific functions, the functions of the application service must be basic. The functions, which are common to all services, are those which relate to lifecycle, such as create. While not all objects are able to create other objects, a client always invokes functions on an object through an object reference. The base class provides a function to delete the object reference. Also certain clients should be able to delete the object itself. The base class provides function to delete the object, subject to access control.

#### 8.2.1.2 Stream Service Element

The Stream Service element is a repository and source for streams. It exports the interface through which the client device controls the media stream. The interface provides functions to:

- control the advance of a media stream, via pause and resume. The interface also provides a scale value, which can be positive or negative. The sign of the scale value determines the direction of the stream. The magnitude of the scale value determines the stream rate.
- inquire about the state of the stream object. The functions return the state of the stream state machine, plus the current time value and the scale value. The solution also places these values in the media stream.
- optionally place events into the media stream.

### 8.2.1.3 Create TS Stream Element

The Create TS Stream Element is a generic object that accepts data in an unspecified format, but which then builds an MPEG-2 Transport Stream as defined in ISO/IEC 13818-1 and accompanies the data stream with appropriate PSI to allow the remainder of the SPS and delivery system to locate the components and integrate them with the remainder of the delivered multiplex. Examples of the Create TS Stream element could include an EMM Generator or real-time MPEG-2 video encoder.

### 8.2.1.4 Data Carousel Stream Element

Data other than Audio/Visual data is required for preparation of a MPTS for distribution via Hertzian networks. Such data streams include ECM and EMM streams, DVB SI (or ATSC SI/EPG) and proprietary data such as for Electronic Program Guides. It is not always appropriate to transfer this data using a Transport Stream, particularly when data is repetitive and/or must have specific timing relationships to the final MPTS generated (e.g. ECMs and the linked use of Control Words in the scrambling engine).

The broadcast of repeated data sections that conform to the section syntax provided in ISO/IEC13818-1 which includes all of MPEG-2 PSI, DVB SI, ATSC SI/EPG and DVB CA (ECMs and EMMs) will be performed using an entity called the “Data Carousel Stream Element”. The output of the data carousel must be an MPEG-2 Transport Stream as defined in ISO/IEC 13818-1 and include all necessary PSI to locate the content streams.

### 8.2.1.5 Download Service Element

The download service provides an interface to transfer data from the server to a client. A download phase is incorporated into the overall session lifecycle, although it may not be necessary to perform download in all cases.

The scope of the download service is to download an executable image from the server to the client. While the service is capable of transferring any arbitrary data, this additional use is outside the scope of the [DAVIC 1.4](#) specification. In addition, the decision as to what executable image to download is outside the scope of the download service.

The download service assumes that configuration of the client with respect to the access network is complete. In the session-oriented scenario where a control connection is required, it is assumed that this connection has been established. Both session-oriented (point-to-point) and broadcast procedures are supported in a DAVIC system; in the former case operation over a reliable network is assumed. At the conclusion of the download procedure, the executable image is installed in the client and available for immediate execution.

### 8.2.1.6 Directory Service Element

The directory service provides an interface for binding logical names and object kinds to object references. The object reference is an addressing structure that holds object location information. The directory can bind any object kind, thus it is suitable as a general purpose name service. The a graph of directories and other objects is typically constructed by the content provider as a result of loading content. The logical path names can be navigated by Clients, and individual names can be resolved to obtain a Client/ Service Element instance connection. Refer to DSM-CC for more details. Service Elements which inherit the directory interface include Content and ServiceGateway.

**8.2.1.7 File Service Element**

The file service exports an interface for client access to files. The client can be not only an STU but also another service element. The content service, for example, can adopt the interface for content transfer. The semantics of the access are random access. The client provides the file origin and data size. The file service abstracts file access. The scope of the service is:

- To control the lifecycle. The service inherits DSM::Base interface lifecycle functions.
- To open and close a file. The service inherits the DSM::Access interface.
- To read and write the file.

**8.2.2 Content Group**

**8.2.2.1 Content Service Element**

The Content Service Element is responsible for the support function of managing content, including loading and unloading content, between the server and the content provider as well as among the service elements.

**Content Provider Domains**

The Service Provider makes a contract with one or more Content Providers to enable them to use storage, computational and delivery resources of the Service Provider System, for the purpose of making applications available to End-Users. The Content Provider is therefore given a fixed Server resource usage ceiling, e.g., a certain amount of disk storage to store content on. The Content Provider is the Owner of this space, and can subdivide the space as it sees fit. Within the space, The Content Provider can store files and run executables. It may even sublet to other Content Providers if the contract permits. In addition, on a per Session basis, the Content Provider may be allotted computational and network bandwidth resources for an Server-side application to run and communicate with an End-User, e.g. a Set-top End-User.

The Content Transfer therefore occurs between one Content Provider’s allotted Domain in one Service Provider System and a Domain of the same or authorized Content Provider in another Service Provider System.

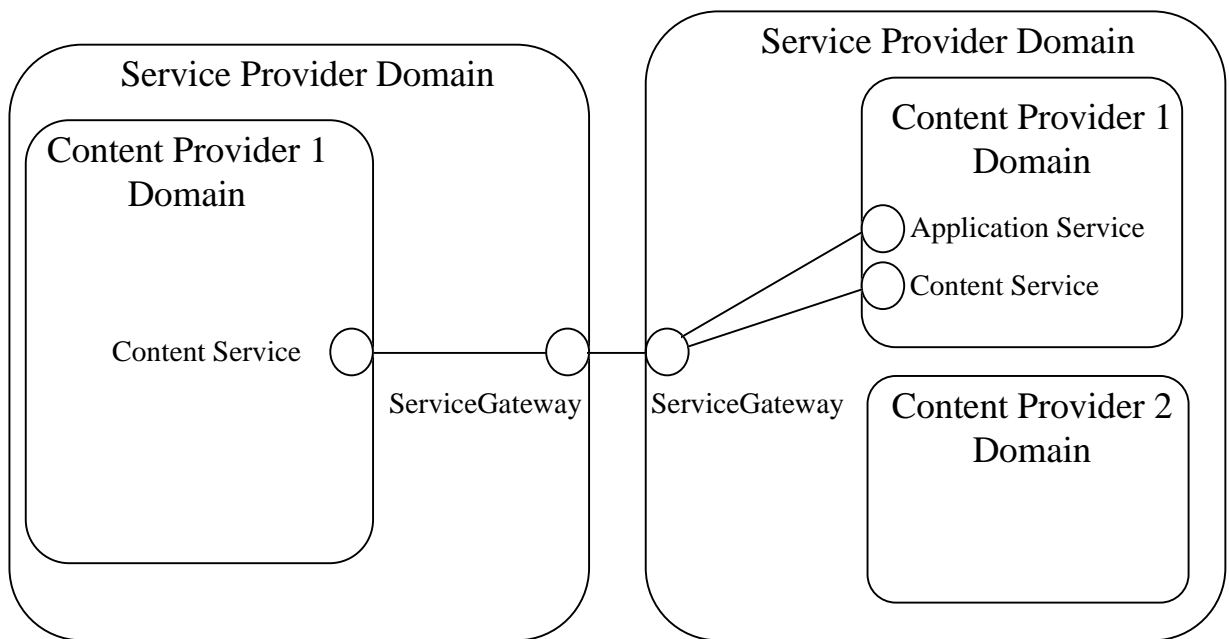


Figure 8.2-1: Logical Paths for Content Transfer

The Content Provider has total flexibility to set up Directory hierarchies within the space allotted. The Content Provider can setup Content and Application Services and register them with the ServiceGateway, using DSM::Directory\_bind(). The Content Provider is the Owner of all nodes in its Directory hierarchies. By definition, the nodes are private unless that Owner sets permissions granting access. Thus, the Content provider can make public to all Set-tops certain paths, and also can set access control for a limited set of End-Users on other paths. The entire system starting from the ServiceGateway is a graph of Service paths, some of which are accessible and others which are not, depending a combination of the identification, authentication and authorization of the End-User.

The Interfaces of applications may also be public or private. The Content Provider again decides whether these are to be made public, and if so, places them in the system Interface Repository. Object attributes and relationships, i.e., schema, can also be published in the system Interface Repository.

### **Content Transfer Assumptions**

The Content Package is used to transfer content between Service Provider Systems. The format of the Content Package is described in Information Representation, Part 9.

### **Content Format Requirements**

The Content Package shall have the following properties for each public content object:

- CosNaming:pathname (the affected node that the load instructions will be applied to)
- Comma-separated string containing load instructions:
  - DSM::Directory operations such as “DSM::Directory::bind”
  - DAVIC instruction “DAV::verify” to cause a verification of content integrity, virus check, etc.
  - DAVIC instruction “DAV::execute” to cause a Server-side executable to run.
  - The load instruction format shall be “<load instruction>,<load instruction>, ...”
- the content itself, as specified by Information Representation, Part 9.

Each object shall have a world-wide unique identifier.

An application executable can be transferred in as content, placed at the pathname indicated, and executed.

A minimal application might consist of a server-side executable and a download object. More complex applications might have MPEG stream objects, multiple download objects, Set-top accessible files, etc. An existing application is notified that content has been updated. Multiple content objects of one Content Package are loaded as a single transaction.

## **Content Transfer Using the A10 API**

The A10 API is defined as the DAVIC Content interface. This interface consists of four operations: copy(), modify(), move() and load().

A new Content Package is transferred from one Domain to the other using the copy() operation of the DAVIC Content interface. An update Content package is transferred using the modify() operation of the DAVIC Content interface.

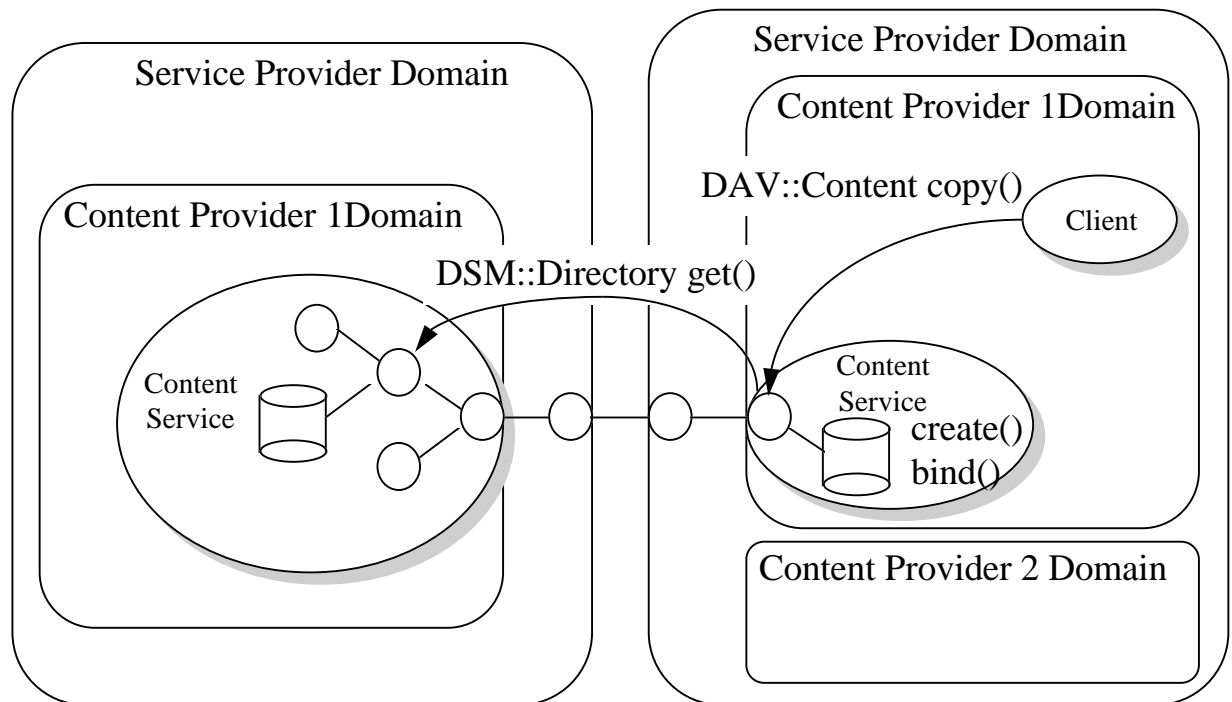


Figure 8.2-2: Content Copy

## The Content Loader

After the Content Package has been transferred from one Domain to the other, it can be unpacked by the load() operation of the DAVIC Content interface. The load() will process each known object in the Content Package by performing one or more load instructions at the given path name with the given content data.

The Content Package may contain DAVIC content types and private content types. The DAVIC types are unpacked. The private types are not processed. The Content Package shall have a pathname where it will be sourced from, indicated as the second input parameter of the copy() and modify() operations. The Content Package shall have a pathname where it will reside, indicated as the first input parameter of the copy() and modify() operations. After the load() of DAVIC content types is completed, the proprietary application can unpack and load the private content types, if there are any.

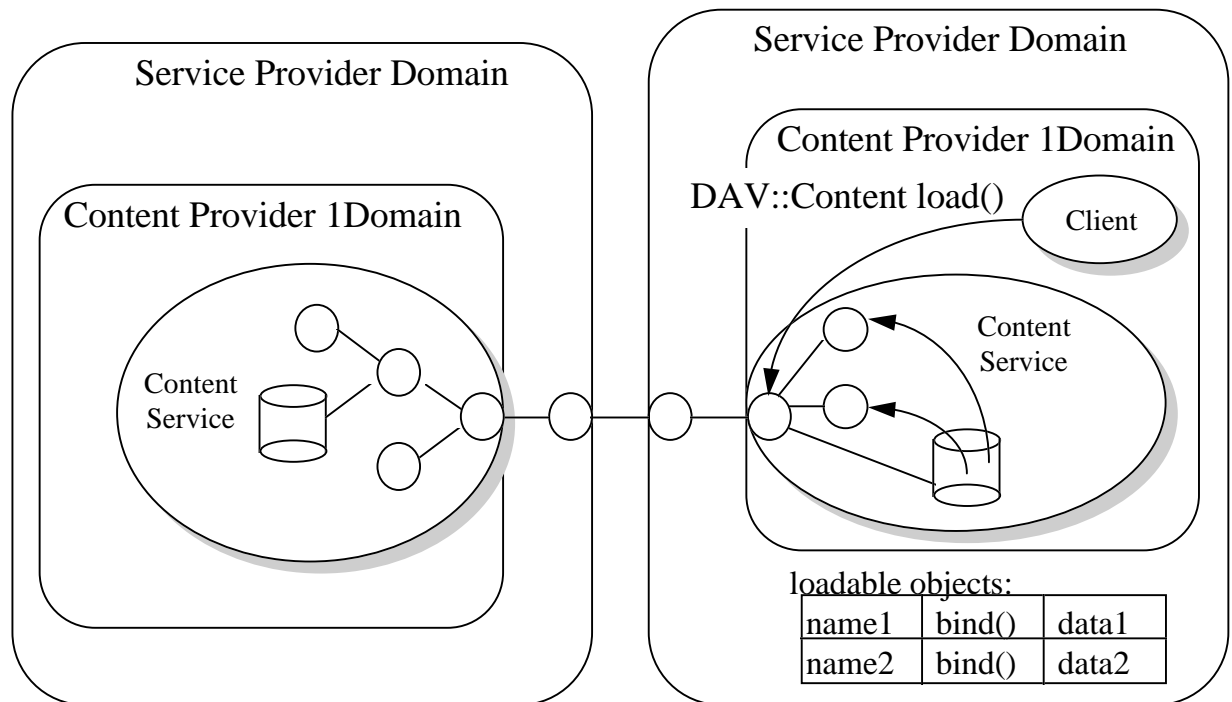


Figure 8.2-3: Content Loading

## 8.2.3 System Group

### 8.2.3.1 Service Gateway Element

The Service Gateway Element is a broker that provides the following functions:

- Organize the Service Provider Domain where services register (or install), make their existence known, and deregister when they are decommissioned.
- Provide a means for the client to discover the existence of a service (i.e. an application service for movies on demand). This is accomplished by browsing the service domain.
- Provide a means by which a client activates and deactivates an instance of a service. Included is a Session Service that facilitates session establishment and management.
- Cooperate in the setting up of connections between services and client entities to transition the execution focus from the Service Gateway to other services, e.g., to the Stream Service after a client has selected a particular movie.
- Match a client with the services that it is authorized to browse.

The service gateway is usually the first element the STU observes. In providing its services, the service gateway must:

- support a flexible description of the service domain. The gateway adopts a name context interface which provides a name graph to organize the service domain.
- install services in the service domain. A client which is a service, that is an object which exports an interface, publishes itself through the service gateway.
- allow a client to browse the service domain. The objective is often to locate a specific service, known through the name bound to the name graph.
- establish a session. Since service gateway supports the session protocol, it expedites the session establishment.
- empower a function that either filters the available services in the browse phase to those which can inter-operate with a client, or selects the service in the activation phase. The client profile interacts with this feature.

### 8.2.3.2 Session Gateway Service Element

The Session Gateway Service enables a service or Service Gateway to add and delete resources by invoking operations that will in turn negotiate resources with the Session Resource Manager (SRM) by calling U-N add and delete resource messages. A Session Gateway also performs translation between User-to-User RPCs and User-to-Network Messages (e.g. Session Establishment messages).

### 8.2.3.3 Client Profile Service Element

The motivation for the client profile is interoperation between a service and its client, so the primary objective behind the client profile is to allow a service to determine if interoperation is feasible. The client profile describes the configuration of the client device, such as the hardware and software that resides on the device. The second objective relates to download. If the software that resides on the client does not inter-operate, but the client allows software download, the service can often download certain software to achieve interoperation. The implication is that the service must obtain the client profile before the download phase, or as the first step of the download phase.

The scope of the client profile is:

- To describe the client device such that a service can decide if interoperation is feasible with the client's resident software. This is likely the first event when download is possible, and is the only choice when download is not possible.
- To describe the client device for the situation where the service can or must download software to the device. In this case the client profile provides information about hardware features, such as instruction set, before it selects the correct software to download.

Once a session phase completes and a connection phase completes, the next phase between a client and a service is often the download phase. The client or the service can elect to skip the phase, but there is an interface for the situation where the client and the service allow download. Because the service must know the client profile to decide what to download, the transfer of the profile is the first step within the download phase.

Note that the concept of a user profile, which describes a human being and not a device, is outside the scope of the profile interface. Also the configuration information which software can access through the management plane (the S5 interface) is outside the scope of the client profile service.

### 8.2.3.4 Interface Repository Service Element

The interface repository service provides operations for verifying consistency and completeness of the public interfaces in use by a Service Domain. It inherits from DSM::directory and DSM::Interfaces. The Directory provides a catalog of IDL files. The DSM::Interfaces operations provide methods to perform the following:

- define or publish a new interface
- check the correctness of an interface and its consistency with the other interfaces in the Service Provider System.
- remove a definition from the repository
- show inherited kinds and constructed types defined by an interface.

## 9. Networked Server Objects

The following diagram illustrates the interaction of a client and DAVIC Service Elements for the purpose of playing an MPEG-2 video stream.

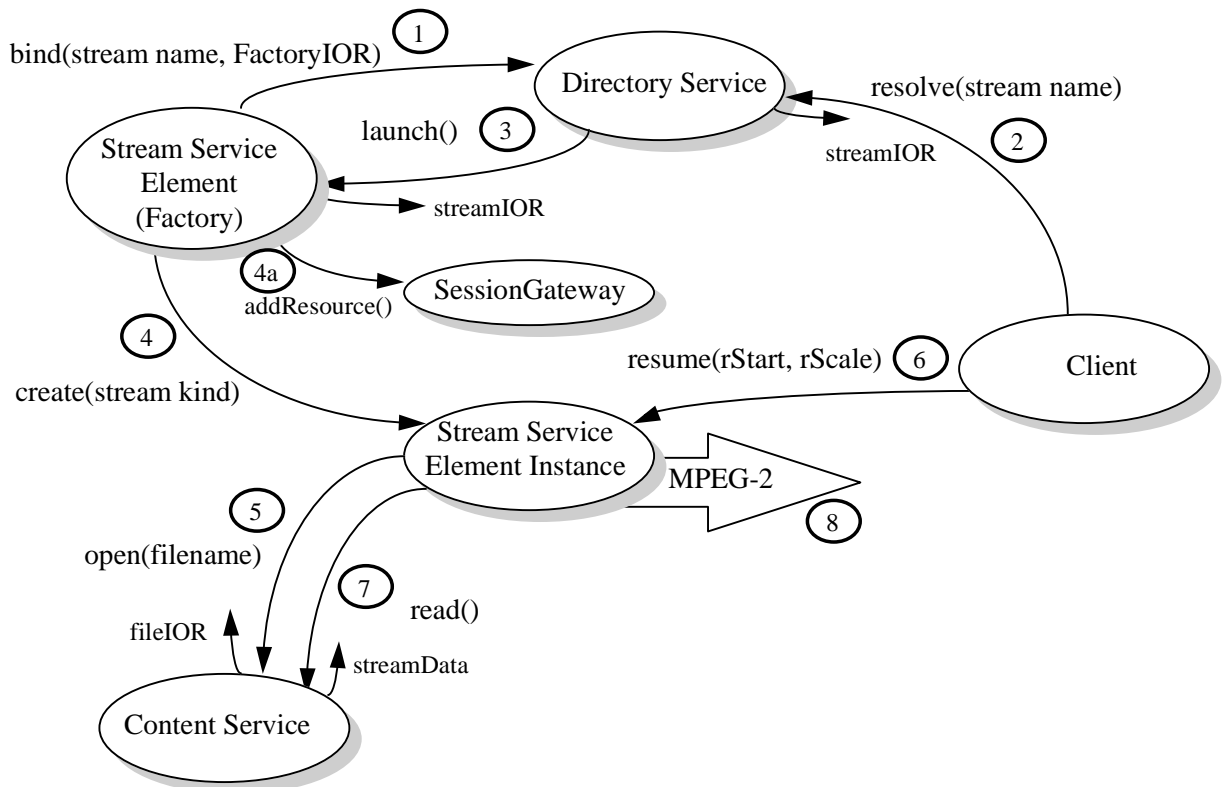


Figure 9-1: Networked Objects Supporting Stream Functionality

1. A Stream Service Element with the capability to create Stream Service Element Instances is referred to as a Factory. The Factory will use the DSM::Directory bind() operation to bind a Factory Inter-operable Object Reference (IOR) to a Directory Service. The Directory Service will contain a binding list of names to Inter-operable Object References.
2. After choosing a name (by menu or other means), a Client will invoke DSM::Directory resolve().
3. Having received the resolve(), the Directory will send a DSM::Service launch() request to the Factory.
4. The Factory will create() a Stream Service Element Instance and a unique Inter-operable Object Reference for it. It will send DSM::SessionGateway addResource() to the SessionGateway to cause the Network to allocate sufficient bandwidth for the requested connection.
5. The Stream Service Element Instance will open a file containing the encoded MPEG Stream.
6. The launch() returns with the IOR of the Stream Service Element Instance. The IOR is passed back to the Client in the resolve() reply. The Client then invokes one of the Stream interface operations, e.g., DSM:: Stream resume(), on the Stream Service Element Instance.
7. In response to the resume(), the Stream Service Element Instance begins reading the MPEG file.
8. The MPEG Transport is delivered over the network to the Client.

The following diagram illustrates the interaction of a client and DAVIC Service Elements for the purpose of downloading and running an application. In this case, the Application Service Element Instance is assumed to inherit multiple interfaces, one of them being DSM::Download. As can be seen, the sequence is similar to that of the Video Stream Play above, with the exception that MPEG Private section is used to download the client-side of the application. Following the Download, the Client may invoke operations on the Application Service Element instance for other interfaces it supports.

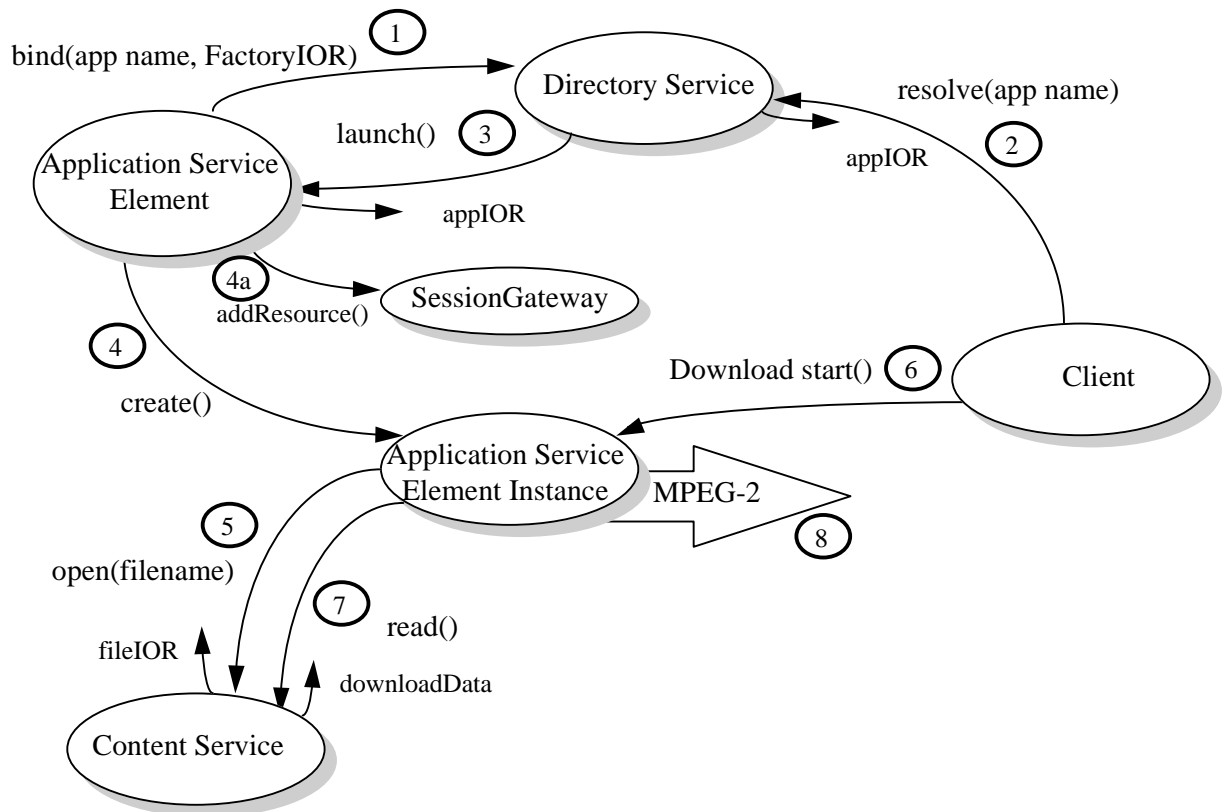


Figure 9-2 : Networked Objects Supporting Download Functionality

## Annex A

# Conceptual Server Model (Informative)

### A.1. Requirements

1. The server model must be an open systems model. Therefore the server will not only be defined as a single entity, but also as a collection of well-defined service elements.
2. At the same time, the server reference model must be implementable and server elements must inter-operate so as to present an interoperable view to the Delivery System and the STU, the other two critical pieces of an end-to-end solution. This interoperable interface to the Delivery System must be mappable to individual interfaces of server elements.
3. A DAVIC-compliant server element is defined with a minimum set of generic functions and interfaces, to allow plug and play at the A9 interface as well as expansion and evolution of the DAVIC system model.
4. A given application will define the services it needs. If an application does not need the functionality or capabilities of a service, then the service element containing that functionality or capabilities need not be provided in that instantiation of the model.
5. A DAVIC-compliant service element may implement a superset of functions and interfaces for a service element, so long as it includes the minimum functions and interfaces for that server element as specified by DAVIC.

### A.2. Advantages of an Open Server Model

The Service Provider system offers a variety of services to support multimedia-oriented applications. The services could be offered by one logical Server box with several functional modules, where the interface to the one box is the primary view from the Network, the STU or other Servers. On the other hand, the Service Provider's system may be defined as a set of distributed service elements each with a distinct function and interface.

At first glance, the distributed model appears to have many interfaces which need to be defined. In fact, many of the attributes of each interface are highly similar to those of the other interfaces. For example, communications stacks are typically defined using the ISO 7 layer model. Layer 7 is the application layer, which provides common services to the applications that reside above it. Layers 6-1 represent the protocol stack which may be common to all of the distributed modules. If there is more than one protocol stack in the system, there may be 2 or 3 choices of core protocol stacks, not one per distributed module. In addition, all modules may have an identical registration/activation interface and a common signaling interface. A desirable distributed model would therefore define these common interfaces for all modules to use, while each module would define a few layer 7 interfaces to the outside world.

DAVIC has adopted an object-oriented, distributed server model. The benefits and advantages are as follows:

- **Interoperability.** Service Elements can be defined as interoperable units, meaning vendors can produce these units, interface them with a DAVIC-compliant system, and run applications on them immediately.
- **Evolution.** New services can be added as new applications place requirements for unforeseen services.
- **Scalability.** Many copies of highly used elements can be implemented in a Service Provider's system, enabling the support of many simultaneous application user contexts.
- **Modularity.** Basic operations are defined in a modular manner, with each element responsible for a specific role. For example, a Service Gateway performs connection-oriented functions and a Stream Service performs continuous media delivery and control.
- **Flexibility.** Service elements may be combined in different ways to form sets of services which support the needs of applications and environments.

- **Value Added Opportunity.** Each element has a minimal set of functions and interfaces. Vendors may add value by improving performance of the minimal functions, adding functions, or adding interfaces. A vendor may add services that inherit from the basic service-element and protocol classes. A service provider may define new instance configurations of the services to support value-added applications.
- **Commonality.** Through an object-oriented inheritance framework, many functions and interfaces are defined once and used by all derived classes.

### A.3. Basic Concepts

A basic concept that shapes the design of the service domain is the distinction between a service that contains another service, but does not realize the interface of the contained service, versus a service that does not contain another service. The classic example of a service which contains another service is the Service Gateway function. The Service Gateway object exports interfaces that allow a client to navigate the name space and select a service. It does not export the interface of services it contains. If the Service Gateway contains a File Service, for example, the Service Gateway does not export the File Service interface. The concept of containment is shown in *Figure A.3-1: Concept of containment*.

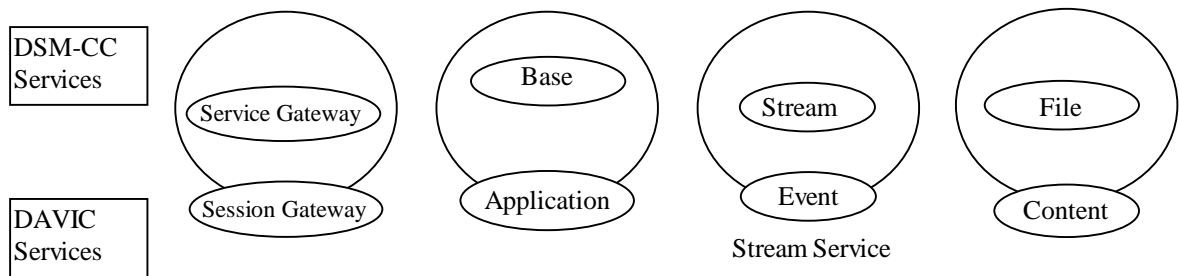


Figure A.3-1: Concept of containment

A service exports a certain interface. The interface which is visible to the STU is often a combination of simple abstract classes. The existence of a simple class tree, where a single class abstracts a single concept, means that it is often possible to reuse the interface. The Name Graph class provides functions to explore a symbolic name space and select a service bound to a specific symbolic name. The Service Gateway interface is one example of an object inheriting the class. A Video Service that organizes film titles as name graphs could also leverage the class.

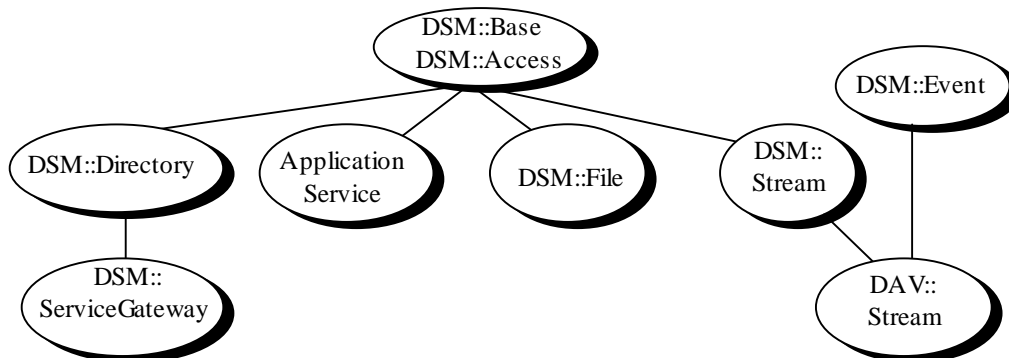


Figure A.3-2: DSM-CC Class Hierarchy

It is important to stress that the STU does not have to understand the subclass on which a concrete service builds. A client can elect to treat the service as a single flat interface. A client that does understand the class structure, however, can elect to create an instance that exports just a fraction of the

complete interface. The motivation might be access control. The access control rules could be that an STU can create a Service Gateway instance that exports browse functions, while a video service can create a Service Gateway instance that also exports installation functions. *Figure A.3-2: DSM-CC Class Hierarchy* on page 19 shows the DSM-CC abstract classes that our model uses to realize services.

### A.4. Standard Service Element

The standard service element is an abstract entity that observes the object hierarchy shown in *Figure A.4-1: Service Element Object Hierarchy*.

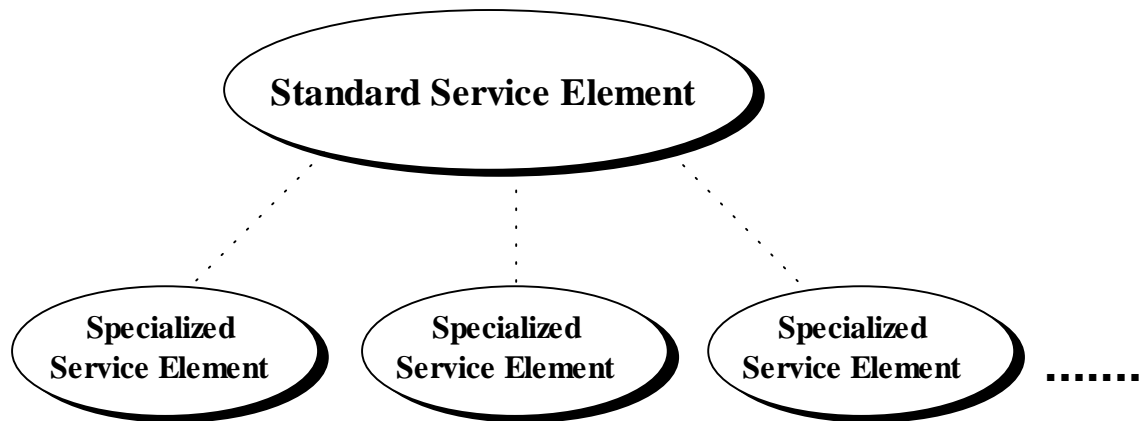


Figure A.4-1: Service Element Object Hierarchy

It acts as the root class for all common methods, and has Control, User, and Management Planes:

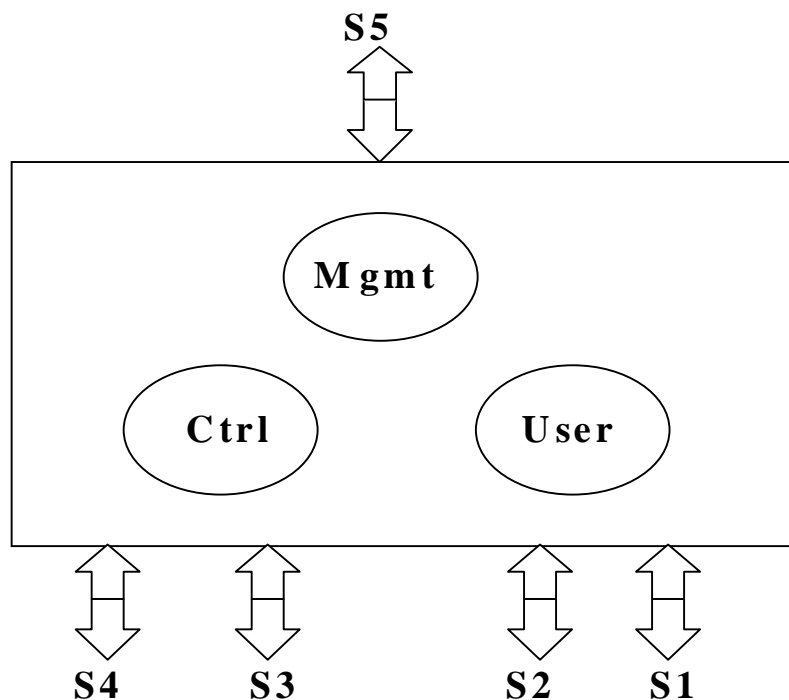


Figure A.4-2: The Standard Service Element

The derived service elements inherit all Standard Service Element protocols and interfaces. The actual implementation of the interfaces for each element will depend on its application context.

### A.4.1. Control Plane

The **Control Plane** is used for session and connection control-related functions and interfaces.

#### A.4.1.1 Connection Control

Connection Control relates to the set-up, tear-down and dynamic control of network resources that a Server needs to perform its functions. ITU/T Q.2931 signaling is used to perform these functions.

#### A.4.1.2 Session Control

A session is a logical entity that is related to the user's interaction with the Server. A session may require multiple connections and dynamic network resource management over its lifetime. These are the functions of Session Control.

As an example, if a user is browsing through a navigator during a session, the user may request an application, say a sports video, that requires different network resources than are currently assigned to it. Connections may need to be torn down, set up, or re-negotiated, depending on the network requirements of the application requested.

### A.4.2. User Plane

The **User Plane** is used for communication between Service Elements in different service domains and between clients (in STUs) and Service Elements. The nature of these bi-directional communications may be highly specific to the Service Element. Typical messages in this plane include the VCR commands for a video server.

### A.4.3. Management Plane

The **Management Plane** is used for configuration, accounting, performance measurement, fault management and security.

## A.5. Standard Service Element Functionality, Interfaces, and APIs

### A.5.1. The Standard Service Element interfaces

S1:	User Plane: This interface is typically used to carry high-bandwidth, bulk data over a transport stream such as MPEG-2. It may be used for replies to RPC requests from the client on S2, e.g. by using the MPEG-2 private section.
S2:	User Plane: This interface is used for requests from any client. It may be used for RPC replies to RPC requests on S2. Clients may be other Service Elements.
S3:	Control Plane, Session. This is used for managing sessions
S4:	Control Plane, Connection. This is used for User-to-Network and Network-to-User Signaling.
S5:	Management Plane, Management. This is used for management interfaces and MIB's.

Consult Parts 7, 8, and 12 for complete explanations of S1 to S5.

## A.6. Server Reference Model Service Elements

Specialized Service Elements inherit all Standard Service Element interfaces, and can be instantiated. Therefore, these elements need only define those interfaces that are defined as NULL in the Standard

Service Element, i.e., S5. In addition, these elements will extend and specialize their functionality primarily through S2 User-User primitives. Part 7 of [DAVIC 1.4](#) lists the functions provided by the elements, the citations of the standards that describe how the functions are achieved, and the constraints on their use in a DAVIC system.

## A.7. The Layered Communications Model

All the interfaces for server elements will follow the OSI reference model.

- Layer 7 defines the specific service interface
- Layer 6 is IDL for S4-S1 interfaces
- Layers 5-1 represent the protocol stack of which there may be several DAVIC protocol stacks. In addition, application download permits download of a DAVIC standard or proprietary protocol stack for use by an application.

Protocol stacks for S1-S5 are detailed in Part 12 of [DAVIC 1.4](#).

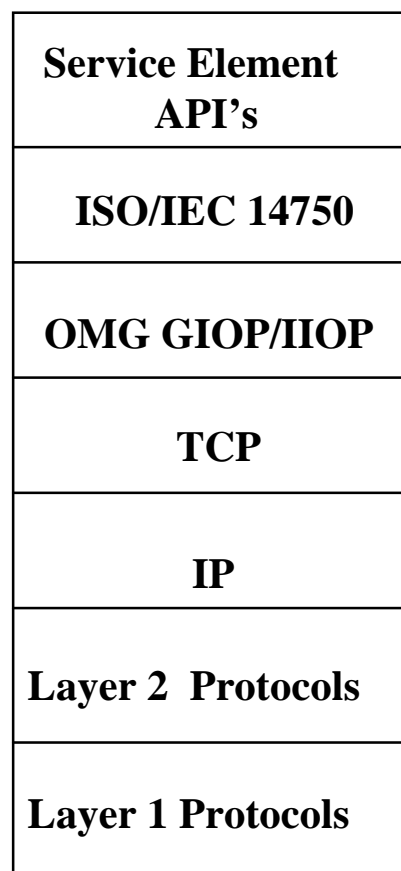


Figure A.7-1: Protocol Stack for S2 and S3 for Server Elements

### A.7.1. Layer 5 Protocols

This layer of the protocol Stack is used for messaging and remote procedure call (RPC) over the transport layer. Two approaches have been identified in order to provide end-to-end interoperability between system entities:

1. Use of the [OMG CORBA 2.1](#) specification for negotiation of Layer 5 protocols and gateway approaches to interoperability, as well as the default Internet Inter-Orb Protocol (IIOP).
2. Use of DSM-CC for run-time selection of the RPC/Messaging mechanism.

ISO/IEC 14750 IDL at Layer 6 must be mappable to Layer 5, and the approach must be consistent with the Layer 3/4 protocol suite.

## **A.8. Network Implementations of a Service Provider System**

In a DAVIC system, the following principles hold:

- Any and all Service Elements may connect to a network.
- There may be more than one network.
- There may be a private network within the Service Provider's domain, in addition to the public network.

## Annex B

### Service Provider Instance (Informative)

The Service Provider System can be represented in many different instances because of the open, distributed design of the Server Reference Model. Figure 3.5 could be used to represent one instance where all the Service Elements are implemented on a single server. In this case only one instance of the Network Interface, Network Control, and Session Control functions are needed and the internal S1 - S4 flows exist only on the one server.

A Service Provider System may also be implemented with the four Service Elements each operating on a separate physical server (or group of servers). In this case, shown in *Figure B-1: Example Instance of Service Provider System*, the Network Interface, Network Control, and Session Management functions are instantiated as needed in each Service Element. The internal S1-S4 information flows are carried across the Service Providers Private network. The description of these internal flows and interfaces is beyond the scope of [DAVIC 1.4](#).

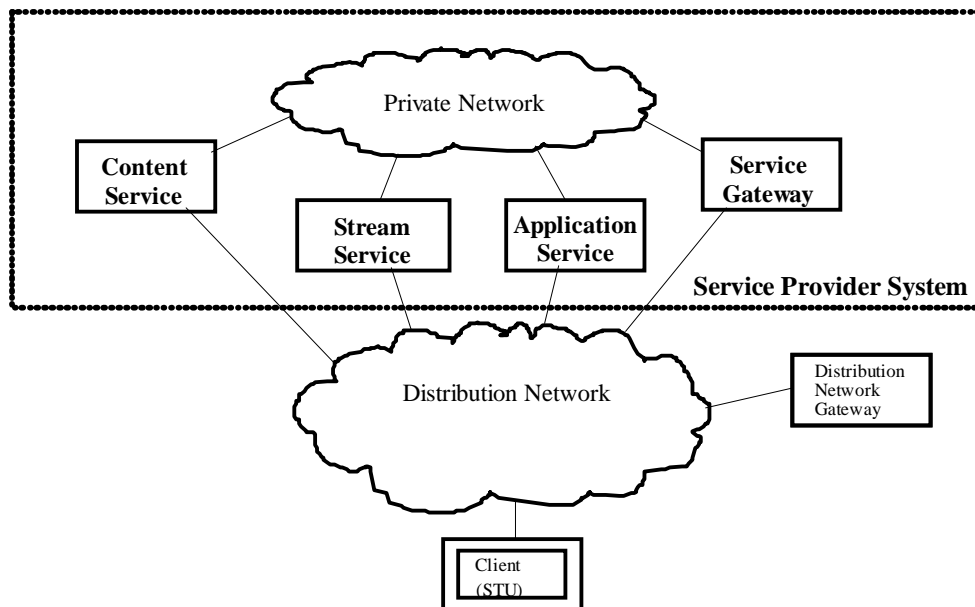


Figure B-1: Example Instance of Service Provider System

## Annex C

# Content Provision (Informative)

### C.1. Scope

This section of the document describes the installation and update of service offerings and the management of associated assets on a Service Provider system. The section is split into two parts:

- Content packaging
- Content delivery

### C.2. Definitions

The *Information Provider* is the creator of a service offering and the *Service Provider* system is the entity responsible for providing end user access to the service.

The *content packaging specification* describes the structure and format of the data which will be passed in a *content package* or *container* from the Information Provider to the Server operated and maintained as part of the delivery infrastructure by the Service Provider.

Content packages will be updated regularly by the Information Provider, and the *content delivery specification* describes the interface which will be used for the delivery and maintenance of content packages.

### C.3. Content Installation and Management Objectives

An Information Provider will treat each DAVIC compliant system as a distribution channel. For efficiency, the Information Provider wants to be able to supply data in a content package which is formatted and structured identically for all Service Providers. The content of each package will vary according to the recipient, even for what appears to be the same service offering. For example, a movie may be supplied with a separate audio stream with a language track for dubbing, or the video content may be edited for airlines or other specialized environments.

The Service Provider will receive content packages from many Information Providers, and also wants a standardized format of package for ease of automation of installation and update. The Information Provider may allow all of the assets contained in the package to be resident on the Service Provider system, or in some of the assets may be retained on the Information Provider system, and released on the receipt of a properly authenticated request from an end user. For example, in a Home Banking application, the executables might be stored at the Service Provider system, but account data could be supplied in an encrypted form as required by the end user.

The Service Provider system operator must have operational control over the service installation and update procedures for quality of service purposes. The operator must also have operational freedom (subject to contractual obligations and agreements) regarding the location and replication of assets used by services in order to manage service performance, availability, and cost of operation. Finally, as Service Provider systems will be composed of elements of hardware and software from various vendors, a unified view of asset management across various servers is essential. Service Provider system developers and operators will need standardized APIs and utilities for such tasks as service installation and update, content usage monitoring, and resource usage monitoring.

The Content Delivery Interface therefore must provide a means of loading or transferring to the server all data needed to run an application, including descriptive data, executables, MHEG objects and multimedia object data. The content delivery interface will support multiple media for delivery of data, including physical, and online transmission.

## C.4. Supply Chain

The success of a video service is closely tied to the process of developing new content and making it available to customers. A simplified version of this process is shown in *Figure C.4-1: Supply Chain from Source to Server*. Before source media (e.g., a newly released movie) can be made available over a DAVIC network, it goes through a number of steps, varying with the nature of the media (i.e., the process for a movie will be different from the process for an interactive multimedia game). In all cases, the process is required to convert the original medium into a multimedia data form, and to provide operational information about the data such that proper resources will be allocated for all phases of data storage and transmission.

In the case of an interactive application (such as a Home Banking application), the Studio will be replaced by an application development group, but the overall process will be essentially the same from the point of view of the Information Provider and the Service Provider. In many interactive applications (such as games or home shopping applications), a studio may well be involved along with the application developer, to provide an integrated multimedia application involving video clips of varying length, and executable code objects of varying complexity. A large application could include very many objects (especially if MHEG is being used), and an automated method of installation and maintenance is essential.

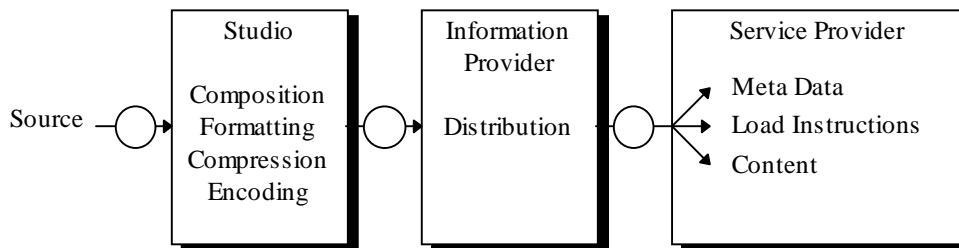


Figure C.4-1: Supply Chain from Source to Server

As an illustrative example, we follow the steps shown in *Figure C.4-1: Supply Chain from Source to Server* for a new movie. The initial box shows the source media, for example a celluloid print of the movie. At the next step, the image undergoes MPEG encoding, a process that both digitizes and compresses the image for storage and transmission. In order for the video server to transmit the movie in response to a customer request, it must store not only the movie, but also information about the movie. This data, with information describing the movie, would be prepared here as well. For example, the same movie may be stored in both 2 Mbps and 6 Mbps versions, or with multiple language audio tracks. Additional information (sometimes known as *meta-data*), will be added to help the Service Provider classify the movie to allow various searches by the end user. The type of movie (comedy, thriller, adult, etc.), the censor's rating, the names of the cast and the director are all examples of meta-data. When the production of content and associated meta-data is complete, it is packaged and sent to the Service Provider.

**Service data** is ancillary information which is required to deliver a complete service. Examples include product pricing, availability, and ordering information in a home shopping service. Service data will often be generated in a different location to that which develops the rest of the content.

As noted above, content may arrive at the server in a number of forms, including tapes or discs. Content loaded into servers may be used for both real time and non-real-time applications as described in the following examples.

### C.4.1. Real Time Distribution

*Figure C.4-2: Real-time online distribution* shows the transmission of content from a broadcast source through the network (represented by the circle) to a server owned by a Service Provider. The information can be sent directly to a Stream Service for redirection to a localized broadcast network.

Additionally, it may be stored in a Content Service, for processing to enable VCR-function capability in subsequent interactive video-on-demand sessions.

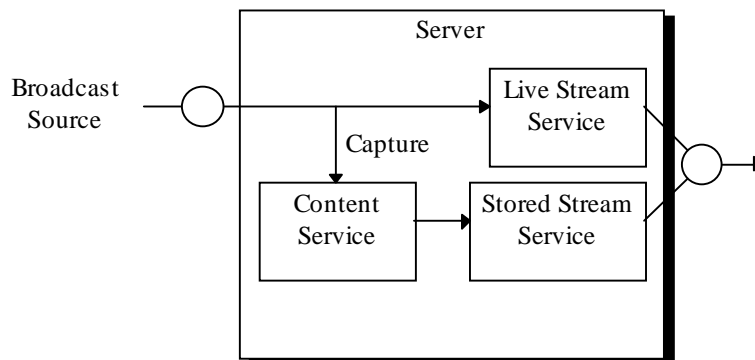


Figure C.4-2: Real-time online distribution

### C.4.2. Non-Real Time Distribution

*Figure C.4-4: Batch distribution* shows the transfer of content from one Service to another Service. For example, content may be copied from a Distributor to the Server's Content Service, moved from the Server's Content Service to the Server's Stream Service, etc.

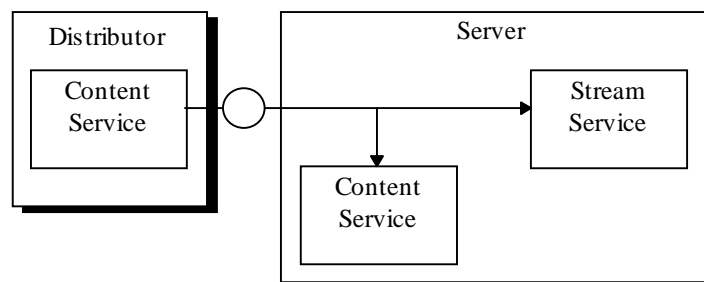


Figure C.4-4: Batch distribution

### C.4.3. Multicast Distribution

*Figure C.4-6: Distribution to multiple servers* highlights the fact that the same content provider may be using the network to supply movies simultaneously to multiple server sites. This would be the case when a new film is released for distribution.

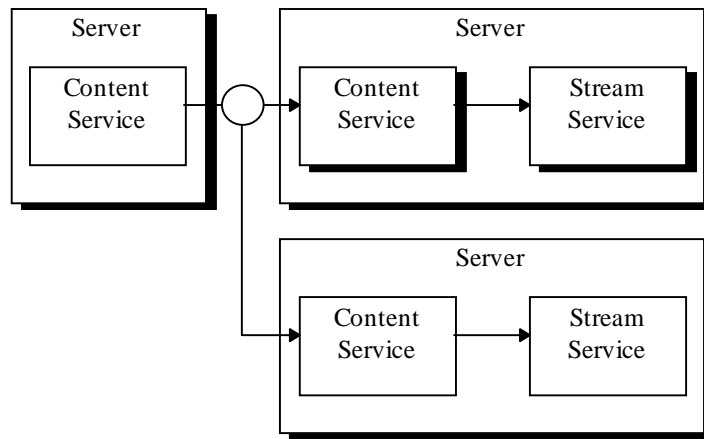


Figure C.4-6: Distribution to multiple servers

### C.5. The Content Package

*Physical Media Dimension*

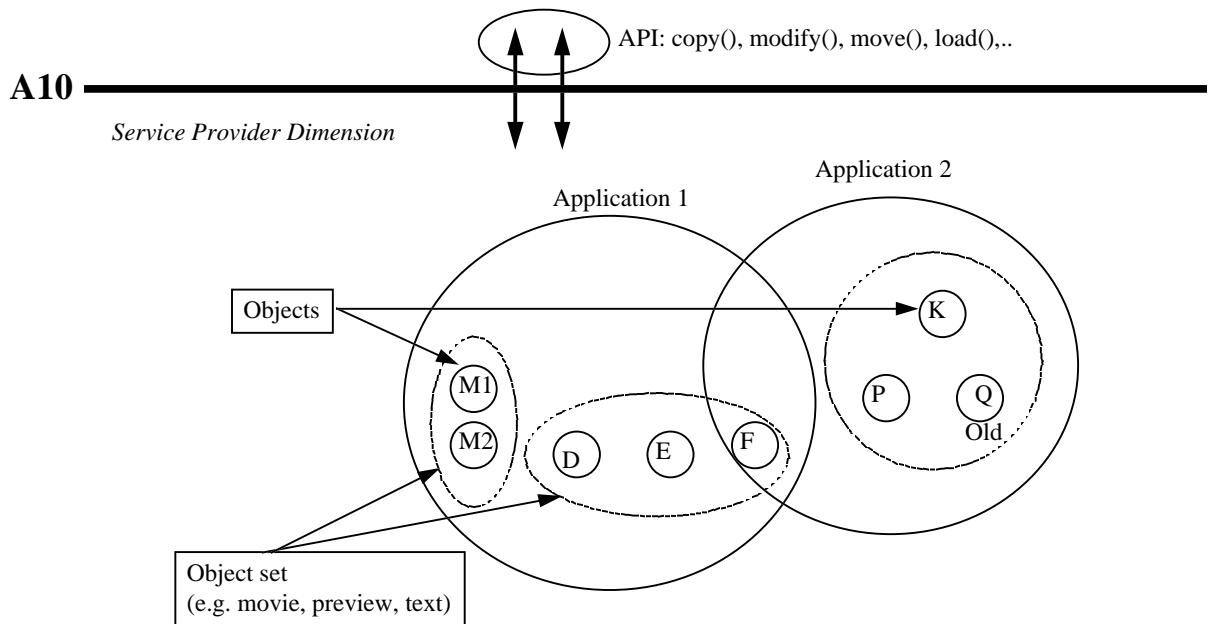


Figure C.5-1: Content Package

Content is made of objects. A television program or an executable application is a collection of objects.

Figure 22 shows some possible relationships between objects, application sets and object sets. Before the actual container requirements are outlined the figures will be discussed to clarify the concepts involved.

Figure 22 shows two applications. Each is composed of objects. In this example, application 1 is a movie on demand (MOD) application and application 2 a movie guide. The objects are located in the

Service Provider System, but the exact locations are unimportant. Objects D, E, and F make up an object set that compose a movie and related information. Objects M1 and M2 are the scripts for running the MOD application. Application 2 is an on screen guide. It is allowed to share some of the objects of application 1. For example, object F is a movie description that the guide uses.

To add or update the set of applications, content may be added or changed by the transmission of a container from the Information Provider to the Service Provider.

Figure 23 shows the same Service Provider system after three separate containers have been installed. What has changed? The movie object set comprising objects A, and B from container ID\_7 have been added to the existing MOD application. A new application (number 3) has been added from container ID\_106. In addition, two isolated objects, S and Q have been installed from container ID\_15, which might contain pricing information. This data may be maintained by a different Information Provider department, which provides service data in a separate container. Object Q replaces a Q object in application 2, whilst object S is simply added to application 3 to make it a complete application.

Applications may be built over time. There is no requirement that a container be a complete stand alone application. When multiple containers are used to supply different parts of an application, the Service Provider will need a tool to validate that an application is complete before it is released to the end user. There must be a table of contents object which can be used for validation of completeness and to validate that the correct revision level has been installed for each object. Until a table of contents object is supplied, the application cannot be validated. The table of contents file may itself be updated, and may be used to validate that the correct release level of each object has been received by the Service Provider.

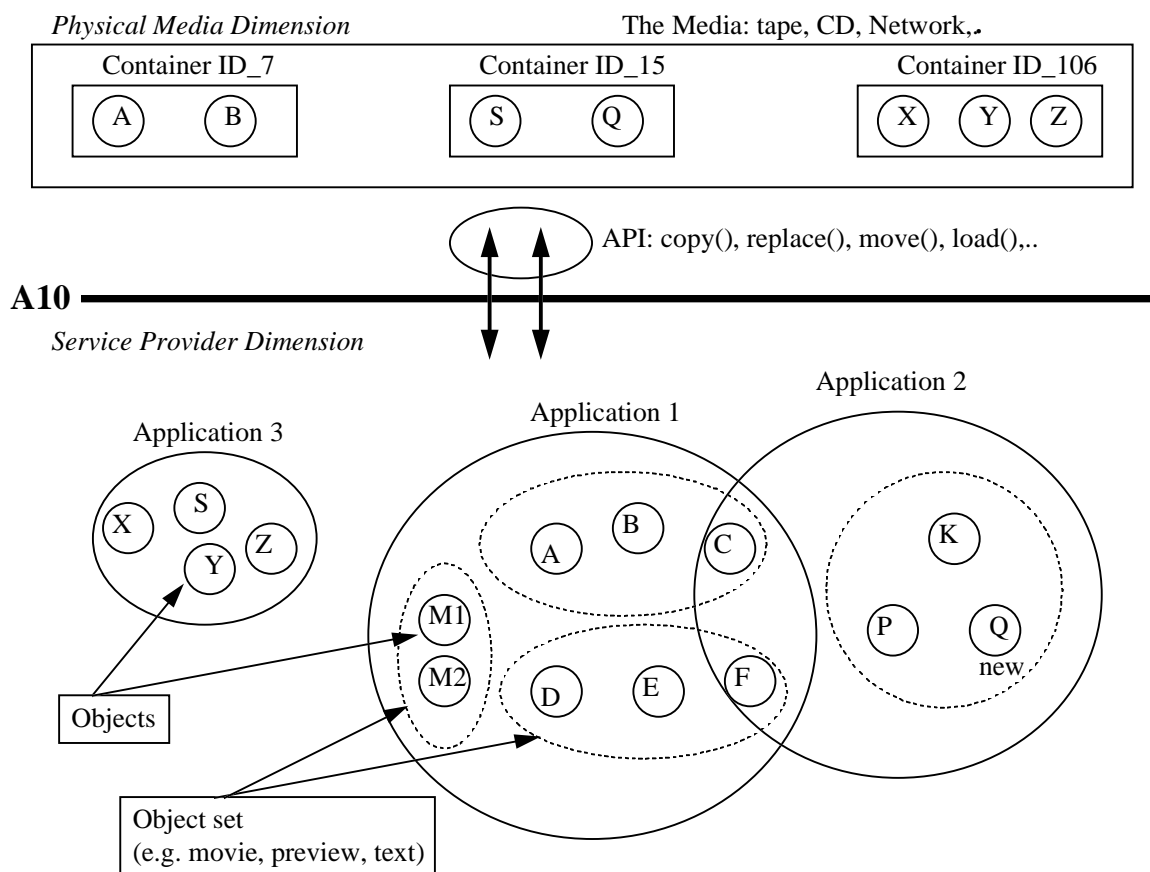


Figure C.5-2: Content Package after update

The exact location of the objects on the server is not defined by DAVIC. Each SPS will install containers in their own way. The physical location of the installed objects is not primary. The important point is that the objects are related to each other by name and/or number and are known by

the applications that need them. In this way a script executing on a STU may call up an object by using its name. The script does not need to be concerned that the object is part of one object set or another. Naturally, the sharing of objects between applications requires a strict naming or numbering convention with possible global registry. This is beyond the scope of this document, but the SMPTE Draft Standard P18.010 (*Universal Labels for Unique Identification of Digital Data*) addresses this requirement for movies.

It is the Server's responsibility to understand object access rights defined by the Information Provider, object names and other object related parameters. It is the servers responsibility to install the objects in the correct physical location(s) as a function of the Service Provider's private needs.

Applications may be well-known or proprietary in nature. A well-known application is one that has a published specification so that content creators may augment it. For example, a Movie on Demand application may be defined to allow for movies to be added by any provider that conforms to the MOD movie object set. On the other hand, a particular shopping application may be "closed". For this case, only the owner of the content creation policies or their agents may create application components as they reside on the SP platform. But in either case, the same API and Container definitions apply.

## C.6. The Service Package Installation Interface

Application publishing is the step which assembles content, meta-data, service scripts and code, and service data into a content package which will be used for service installation and update by a Service Provider system operator.

The Information Provider will use content creation tools (not specified by DAVIC) to both create and edit media content such as video clips or graphic stills. Original content may be created at this stage, or it may be imported from elsewhere. Authoring tools are used to create both the client side of a service (e.g. set top script or a series of MHEG objects) and a server side component.

The Information Provider will also provide meta-data and service data to compile a complete content package.

The Service Provider system complex includes the four DAVIC service elements (Content, Service Gateway, Stream, and Application) as well as application and asset repositories, and a service registry.

## C.7. Loading Content onto the Server from Service Provider

Additions, deletions and replacement of objects are accomplished via the A10 API primitives.

The A10 reference point is the logical interface through which the DAVIC Service Provider domain loads and updates Content Service packages. Three essentially different methods of delivery of content exist, namely content on physical media, real-time content over a communications link, and non-real-time content over a communications link.

Figure 22 and Figure 23 show the domains on either side of the A10 interface. The A10 point provides for the loading of container information into a Service Provider System. The Content Service is used by the Service Provider to locate objects within the container and move them into the Service Provider System. The Information Provider supplies a named container with one or more of the following,

- Complete application
- Object set(s)
- Individual object(s)

The example shows several important concepts. Firstly, the container is a folder or package that is used to transport objects from the content creator or distributor to the SPS. A container may contain all or part of an application. Container objects are installed onto a SPS by use of the A10 API. This document, as part of the [DAVIC 1.4](#) specification, does not define the API beyond its general requirements.

### C.7.1. Methods of Loading

Content can be loaded as follows:

### C.7.1.1 Non Real-time Loading

#### Physical Media

An industry-standard physical media containing the Content (including associated metadata and service data) for the specific content service package is loaded onto the server. Once the media has been checked for integrity, the local process can be fully automated to eliminate operator intervention).

#### Communications Link

The Content service package, not fully defined in the [DAVIC 1.4](#) specification, is retrieved from a remote server, the transfer process either being initiated by the receiving server or the Content system, using a mechanism to be agreed between the two parties.

### C.7.1.2 Real-time Loading

The content arrives over a communications link in real-time, with associated objects embedded within the MPEG multiplex private data part. Such content may be immediately streamed to customers and/or stored in the Service Provider domain for later (interactive) access, depending on parameters set in relevant objects within the stream.

### C.7.2. Version Control

Performing asset updates to occur in an atomic fashion (i.e. all asset changes occur before any changes are available to an application in order to achieve asset set consistency), and keeping asset and application versions aligned are important issues. In addition, it will be necessary for an Information Provider to be able to ascertain the exact complete configuration of his application within a specific DAVIC-compliant Service Provider domain. This can be done by synchronizing the Configuration Management information available between the Service Provider domain and the Information Provider over the A10/S2 interface using methods like database replication.

## C.8. The Content Server Element

### C.8.1. Content Element

- S1: Optional
- S2: install (Add) remove (Delete) update (Replace) interrogate
- S3: standard element
- S4: standard element
- S5: MIB Service Group

#### C.8.1.1 Function

The content server element has two main functions: service installation, and service management.

#### C.8.1.2 Service Installation

Service installation enables the information provider to:

- add new (install) service packages
- remove existing (de install) service packages
- modify existing service packages

#### C.8.1.3 New Service Installation Process

- load STU executable(s) into application boot load component of gateway element
- load server executable(s) into appropriate application element

- load asset containers onto appropriate servers (stream or application elements)
- register the service package (executable) with gateway element
- register asset container access services with gateway element

Note: Stream and application service element specific asset container loading methods must be provided for, since the type of atomic assets stored in the containers is best / only understood by stream or application elements.

#### **C.8.1.4 Existing Service Removal Process**

- de-register the service package (executable) with gateway element
- de-register asset container access services with gateway element
- unload STU executable(s) from application boot load component of gateway element
- unload server executable(s) from application element(s)
- unload asset containers from servers

#### **C.8.1.5 Service Update Process**

- install a new version of a Service
- install new versions of existing individual assets
- modify database and object attribute values
- reconstruct directory hierarchies

#### **C.8.1.6 Executables**

- de register the service package (executable) with gateway element
- replace STU executable on the application boot load component of gateway element
- replace server executable(s) on application element(s)
- register the service package (executable) with gateway element

#### **C.8.1.7 Assets**

- mark asset access service as unavailable in gateway service registry
- add / delete / replace atomic assets (perhaps through a 'source / version' control tool)
- mark asset access service as available in gateway service registry

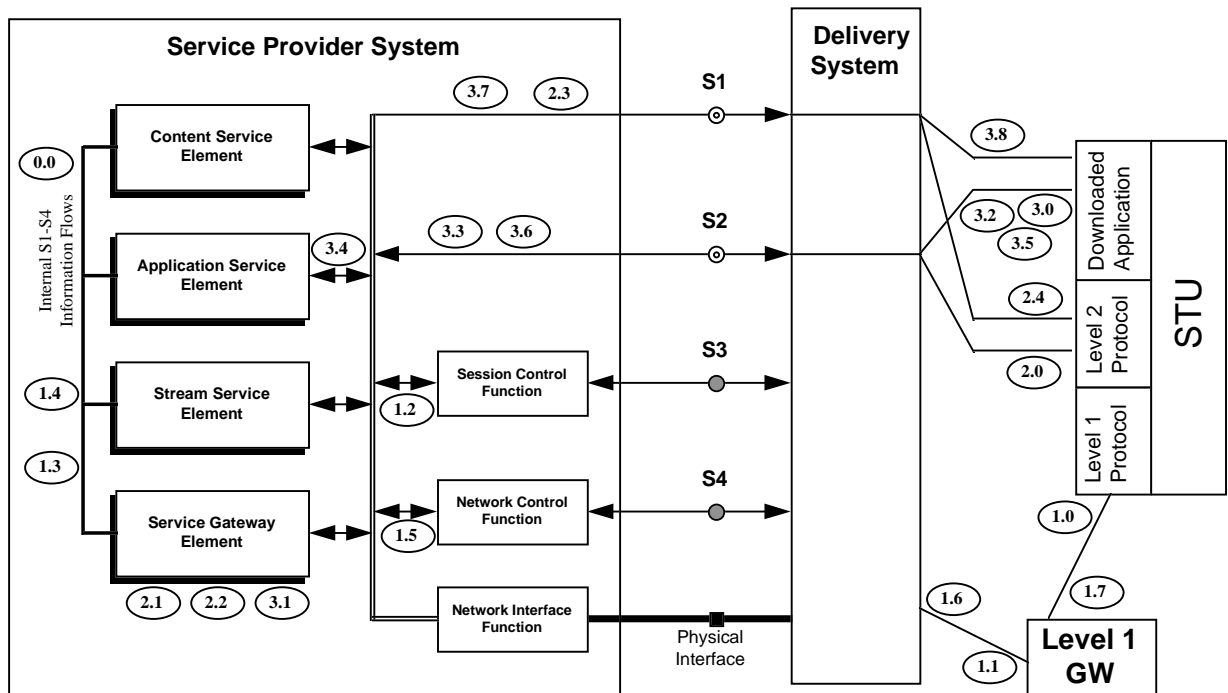
A combined executable and asset update will involve both of the above in an integrated fashion.

## Annex D VoD Scenario (Informative)

This annex is not an integral part of this specification.

This annex is used to describe scenarios for the DAVIC applications. The scenarios specify which interfaces are used between the Service Provider and the Network, and between the Service Provider and the STU. They show a sequence of messages and transactions that occur in sample situations over the information flows S1-S4.

### Simple VOD Scenario #1



Assumptions for this implementation instance of the Server Reference Model. This reflects just one possible implementation and is not normative.

1. The Download Service is contained within the Stream Service Element.
2. The Service Gateway Element is the only element that contains the session management function.  
Default resources are assigned by the Service Gateway Element when the initial session is set up.  
This includes identifying where S2 and S1 terminate and their default bandwidths.
3. Initially S2 terminates at the Service Gateway Element and S1 at the Stream Service Element.
4. The connection management function is available within the Stream Service, Service Gateway and the Application Service Elements.

#### Video On Demand Scenario # 1

##### Step 0: Server Initialization

0.0 All Service Elements register with the Service Gateway using register().

##### Step 1: Session Set up

1.0 STU engages in Session Set up Protocol with L1 Gateway

1.1 L1 GW engages in Session Set up Protocol with Service Gateway over S3

- 1.2 The S3 interface message terminates at the Service Gateway
- 1.3 The Service Gateway coordinates Session Management over the internal S3 interface
- 1.4 The Service Gateway coordinates Connection Management over the internal S2 interface
- 1.5 All Connection Management gets negotiated through the Network Control Function.
- 1.6 Session is confirmed by the Service GW to the L1 GW
- 1.7 Session is confirmed by the L1 GW to the STU

At this point all necessary Server and Delivery System Network Resources have been allocated for the Session requested by the STU. The STU has a handle to the Service Gateway. Also there is an S2 channel to the Service Gateway and an S1 from the Stream Service Element.

#### Step 2: Level 2 Download

- 2.0 STU engages in L2 download protocol with the Service Gateway
- 2.1 The Download Message goes through the S2 interface to the Service Gateway
- 2.2 The Service Gateway scans registration tables and profiles then directs the download message to the Stream Service Element
- 2.3 The Download Image is downloaded from the Stream Service Element over S1
- 2.4 The Download Image is sent through the Delivery System to the STU over S1

#### Step 3: Navigating and Using Level 3 Services

- 3.0 The STU engages in a navigation session with the Service Gateway over S2
- 3.1 The list of registered services is retrieved by the Service Gateway
- 3.2 The STU chooses a specific Application Service and binds to it\*
- 3.3 Messages arrive over the S2 interface to the Application Service
- 3.4 The Application Service responds to the STU commands -- a movie is selected and the handle for the Stream Service Element storing the movie is returned over S2\*
- 3.5 The STU uses the handle to send a specific commands to control the movie over S2
- 3.6 The Stream Service Element receives the command to play a movie over S2
- 3.7 The movie stream is sent over S1
- 3.8 The movie stream is delivered to the STU by the Delivery System Network

\* Renegotiation of S2 network resources is required to allow new physical channels if the Service Gateway Element is physically separate from the Application Services Element or Stream Service Element.