ETSI TS 103 320 V1.1.1 (2015-05)

Digital Video Broadcasting (DVB);
GEM Companion Screen Service Framework

EBU
OPERATING EUROVISION

DVB
Digital Video Broadcasting

Reference
DTS/JTC-DVB-350

Keywords
companion screen, GEM, UPnP

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Contents

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://ipr.etsi.org).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This Technical Specification (TS) has been produced by Joint Technical Committee (JTC) Broadcast of the European Broadcasting Union (EBU), Comité Européen de Normalisation ELECtrotechnique (CENELEC) and the European Telecommunications Standards Institute (ETSI).

NOTE: The EBU/ETSI JTC Broadcast was established in 1990 to co-ordinate the drafting of standards in the specific field of broadcasting and related fields. Since 1995 the JTC Broadcast became a tripartite body by including in the Memorandum of Understanding also CENELEC, which is responsible for the standardization of radio and television receivers. The EBU is a professional association of broadcasting organizations whose work includes the co-ordination of its members' activities in the technical, legal, programme-making and programme-exchange domains. The EBU has active members in about 60 countries in the European broadcasting area; its headquarters is in Geneva.

European Broadcasting Union
CH-1218 GRAND SACONNEX (Geneva)
Switzerland
Tel: +41 22 717 21 11
Fax: +41 22 717 24 81

The Digital Video Broadcasting Project (DVB) is an industry-led consortium of broadcasters, manufacturers, network operators, software developers, regulatory bodies, content owners and others committed to designing global standards for the delivery of digital television and data services. DVB fosters market driven solutions that meet the needs and economic circumstances of broadcast industry stakeholders and consumers. DVB standards cover all aspects of digital television from transmission through interfacing, conditional access and interactivity for digital video, audio and data. The consortium came together in 1993 to provide global standardization, interoperability and future proof specifications.

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

Introduction

Enhanced TV DVB-services, based on the "Digital Convergence" paradigm, are becoming available today in a variety of forms in the digital marketplace. The market has seen an explosion in the distribution and consumption of audio and video content through a variety of connected devices, like smart phones, tablets, PCs, and hybrid STBs and TVs (typically connected to the broadcast and to the broadband channels).

It is recognized that a new, emerging trend is expanding the focus of interactive services from the main TV screen only to a wide range of different connected companion screens, extending the range of possibilities and providing new levels of engagement to end users.

The commercial success of personal and portable devices, often used as a second or companion screen by the user to search and retrieve information or additional content while watching traditional TV services, creates new opportunities to provide compelling services based on interactions among users, devices and content.

For the DVB GEM Middleware (specified in ETSI TS 102 728 [1]), a deeper investigation of the evolution of interactive TV services has happened, envisaging how those services will integrate companion devices to meet user's expectations in the near future.

Based on clear use cases to support interactions with the main TV screen and related content consumption also from companion screens, a set of GEM extensions has been defined which are described by the present document.

These extensions are called the *GEM Companion Screen Service Framework*.

The GEM *Companion Screen Service Framework* includes a service capable of delivering synchronization information as defined by ETSI TS 103 286-2 [2]: "Digital Video Broadcasting (DVB); Companion Screens and Streams; Part 2: Content Identification and Media Synchronization".

1 Scope

The present document specifies the *GEM Companion Screen Service Framework*, which is addressing the Phase 1 and 1+ of DVB's companion screen requirements. These requirements ask for extensions of the GEM middleware specification ETSI TS 102 728 [1], to support information exchange and synchronization between the companion screen and the primary service.

The *Companion Screen Service Framework* provides the infrastructure for GEM companion services that offer their functionality to companion devices in the home network. GEM companion services allow broadcasters and content providers to dynamically provide companion services that integrate screen devices, such as mobile phones, tablets, PCs etc. into the viewing experience.

GEM *companion services* can be deployed via regular GEM applications and enable the broadcaster and content provider to dynamically add companion services, thus augmenting the experience of the viewer to companion devices.

These companion devices can communicate with the GEM companion services via standardized protocols (e.g. UPnP, REST, WebSockets) or can chose to implement proprietary communication protocols.

The framework defines a common discovery mechanism based on the UPnP Application Management Service for these services.

The companion screen service framework as defined by the present document is orthogonal to GEM profiles and versions and can be used on all GEM platforms and derived platforms (MHP, OCAP, BD-J).

2 References

2.1 Normative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found at http://docbox.etsi.org/Reference.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are necessary for the application of the present document.

[1] ETSI TS 102 728: "Digital Video Broadcasting (DVB); Globally Executable MHP (GEM) Specification 1.3 (including OTT and hybrid broadcast/broadband)".

NOTE: Available at http://www.etsi.org/deliver/etsi_ts/102700_102799/102728/01.02.01_60/ts_102728v010201p.pdf.

[2] ETSI TS 103 286-2: "Digital Video Broadcasting (DVB); Companion Screens and Streams; Part 2: Content Identification and Media Synchronization".

NOTE: Available at http://www.etsi.org/deliver/etsi_ts/103200_103299\10328602/01.01.01_60/ts_10328602v010101p.pdf.

[3] ISO/IEC 29341 (September 2014): "UPnP ApplicationManagement:1 Service".

NOTE: Available at http://upnp.org/specs/ms/UPnP-ms-ApplicationManagement-v1-Service.pdf.

[4] IETF RFC 2616 (1999): "Hypertext Transfer Protocol -- HTTP/1.1".

NOTE: Available at http://www.ietf.org/rfc/rfc2616.txt.

[5] IETF RFC 6455 (December 2011): "The WebSocket Protocol".

NOTE: Available at http://tools.ietf.org/html/rfc6455.

[6] ETSI EN 300 468 (V1.14.1): "Digital Video Broadcasting (DVB); Specification for Service Information (SI) in DVB systems".

NOTE: Available at http://www.etsi.org/deliver/etsi_en/300400_300499/300468/01.14.01_60/en_300468v011401p.pdf.

[7] ETSI TS 102 809: "Digital Video Broadcasting (DVB); Signalling and carriage of interactive applications and services in Hybrid broadcast/broadband environments".

NOTE: Available at http://www.etsi.org/deliver/etsi_ts/102800_102899/102809/01.02.01_60/ts_102809v010201p.pdf.

[8] ISO/IEC 29341-1:2011: "Information technology -- UPnP Device Architecture -- Part 1: UPnP Device Architecture Version 1.0".

NOTE: Available at http://upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.0.pdf.

[9] IETF RFC 2818 (2000): "HTTP Over TLS".

NOTE: Available at http://tools.ietf.org/html/rfc2818.

[10] IETF RFC 7159 (2014): "The JavaScript Object Notation (JSON) Data Interchange Format".

NOTE: Available at http://tools.ietf.org/html/rfc7159.

[11] Java API for JSON (JSON in Java).

NOTE: Available at http://www.json.org/java/index.html.

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] "Principled Design of the Modern Web Architecture", ROY T. FIELDING and RICHARD N. TAYLOR, May 2002.

NOTE: Available at http://www.ics.uci.edu/~taylor/documents/2002-REST-TOIT.pdf.

[i.2] ETSI TR 103 286-1: "Digital Video Broadcasting (DVB); Companion Screens and Streams; Part 1: Architecture".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the following terms and definitions apply:

application: functional implementation realized as software running in one or spread over several interplaying hardware entities

association: process or act of establishing a link between a primary TV device and a companion screen application

NOTE: The link does not need to be exclusive or permanent.

companion screen: personal and usually portable device such as a smart phone, a tablet PC or a laptop with broadband connectivity

NOTE: It is sometimes referred also as "second screen".

companion screen application: software application that executes on a companion screen and enhances the main TV screen experience

companion screen event: event sent to companion screen application related to a change in primary device state

companion screen service framework: extension of GEM that is integrated in the GEM platform to expose primary screen functions to companion screens

NOTE: The companion screen framework is specified by the present document.

companion service: feature provided by the primary device to the companion devices

GEM application: application running upon a GEM middleware

GEM device: device running GEM middleware

GEM companion service: companion service offered by a GEM device

GEM middleware: middleware which agrees to GEM specification and requirements

GEM platform: GEM device

hybrid STBs: STB connectable to internet

home network: local network within the home

NOTE: Devices are connected via Wifi or Ethernet. A router/gateway device connects the home network to the Internet via an ISP.

middleware: computer software that provides services to software applications beyond those available from the operating system

notification: message from a primary device to a companion screen application that is sent upon the state changes in the state of the primary device

primary device: TV or STB for receiving and decoding digital television services

NOTE: It may include (personal) video recorder functions, access to additional IP based on-demand services, interactive applications, etc. It is a shared device that, in the context of the present document, could be hierarchically referenced with a companion screen (i.e. a personal portable device).

primary device state: body of information which describes the state of the primary device at a particular time

NOTE: It is composed by information related to TV service (present & following) and information about primary device settings.

primary GEM device: primary device running GEM middleware

primary screen: primary device

primary service: TV service played by the primary device

service: grouping (usually defined by a PMT) of one or more data streams which are offered as a whole to the user

subscription: mechanism for a companion screen application to register itself to primary device in order to receive notifications

transient application: interactive application running upon primary device in agreement to GEM application lifecycle

NOTE: It can be provided both from broadcast and broadband feed.

3.2 Abbreviations

For the purposes of the present document, the following abbreviations apply:

AIT	Application Information Table
AMS	Application Management Service
API	Application Programming Interface
BD-J	Blu-ray Disc Java
CII	Content Identification and Information
CSA	Companion Screen Application
CSS	Companion Screen Services
CSS-WC	CSS-Wall Clock
DCP	Device Control Protocol
DDD	Device Description Document
DVB	Digital Video Broadcasting
GEM	Globally Executable MHP
GENA	General Event Notification Architecture
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
IETF	Internet Engineering Task Force
IP	Internet Protocol
JSON	JavaScript Object Notation
MHP	Multimedia Home Platform
MUG	MHP Users Group
OCAP	OpenCable Application Platform
PC	Personal Computer
PMT	Program Map Table
QR-code	Quick Response Code
REST	REpresentational State Transfer
RFC	IETF Request for Comments
RPC	Remote Procedure Call
SI	Service Information
SOAP	Simple Object Access Protocol
SSDP	Simple Service Discovery Protocol
STB	Set Top Box
TCP	Transmission Control Protocol
TE	Trigger Event
TS	Timeline Synchronization
TV	Television
UDA	UPnP Device Architecture
UDN	Unique Device Name
UDP	User Datagram Protocol
UI	User Interface
UPnP	Universal Plug and Play
URI	Uniform Resource Identifier
WC	Wall Clock
XML	eXtensible Markup Language

4 Companion Screen Service Framework

4.1 Introduction

4.1.0 Overview

The model of interaction between companion screen applications and a GEM device is based on the concept of the *Companion Screen Service Framework*. This framework is included in the GEM middleware; it is able to expose functions of the primary GEM device to companion screens directly or via a GEM application. Those functions, conceptually equivalent to APIs, may be used to retrieve information about the status of the primary device or to control its behaviour. The availability of these functions is not related or influenced by the GEM application lifecycle when companion screens interact directly with functions provided by the *Companion Screen Service Framework*.

4.1.1 Core Framework Functions

The *Companion Screen Service Framework* provides a set of functions defined as *core functions* that can be accessed directly by companion screen applications. This is a subset of all the possible functions exposed by the primary device. In line with the GEM traditional model, specific sensitive controls or information (typically those more content-related) can only be accessed through a GEM application, therefore with a direct control by the broadcaster.

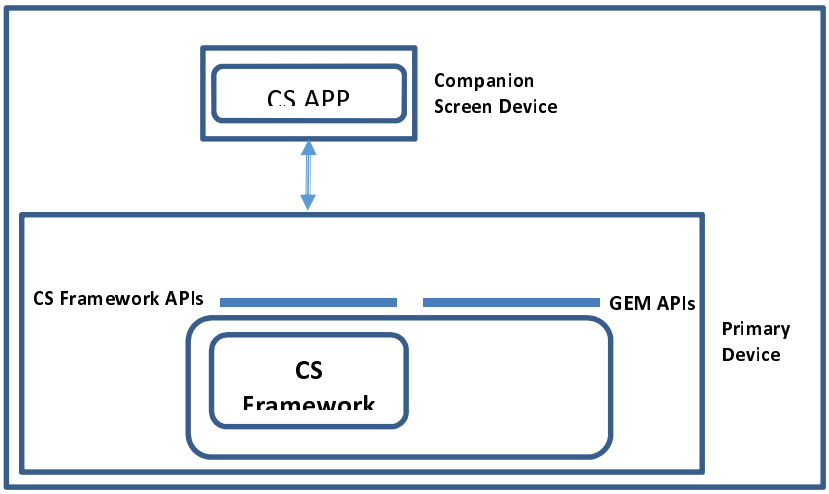

Figure 1: Core Framework Functions

4.1.2 Application-Defined Functions

A companion screen application can access a richer set of functions (including those provided by the *Companion Screen Service Framework*) when a GEM application, designed to support it, is active on the primary device. This model of interaction is directly under the control of the broadcaster.

In this case, as depicted in Figure 2, the GEM application can implement specific logic to handle the requests coming from a companion screen application, and can then access *Companion Screen Service Framework* functions via local GEM APIs. In this model, constraints imposed by the GEM application lifecycle apply.

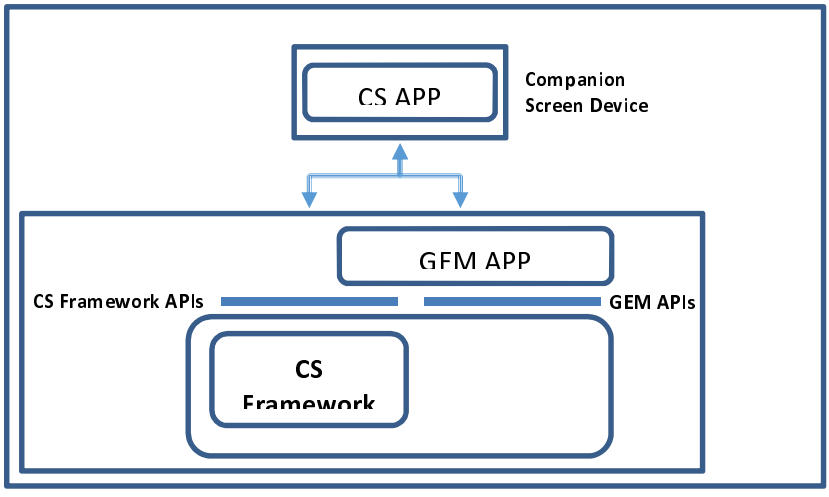

Figure 2: Application-Defined Functions

Through a GEM application, the broadcaster can implement companion screens functionalities with customized and flexible authentication and authorization mechanisms (if required), and can access a variety of sensitive functions exposed by the *Companion Screen Service Framework* (e.g.: interaction with the TV service on the main TV, redirection of content, etc.).

4.2 Framework Architecture

The *Companion Screen Service Framework* offers the common infrastructure to provide its services (*Companion Services*) to the companion screen applications as a set of services. The *Companion Services* defined by the present document address the requirements for Phase 1 and Phase 1+. They can be extended by services defined in later phases of the companion screen work.

The framework provides the necessary infrastructure to seamlessly integrate *Companion Services*, that are defined by later specifications and also supports dynamically extending the set of available *Companion Services* via transient GEM applications. *Companion Services* can be used by multiple companion devices simultaneously, where companion devices may run multiple connected applications. *Companion Services* can provide multiple protocol endpoints for using them via different protocols.

All *Companion Services* are handled by the *Service Manager*, which implements common functionalities for the discovery and management of *Companion Services*.

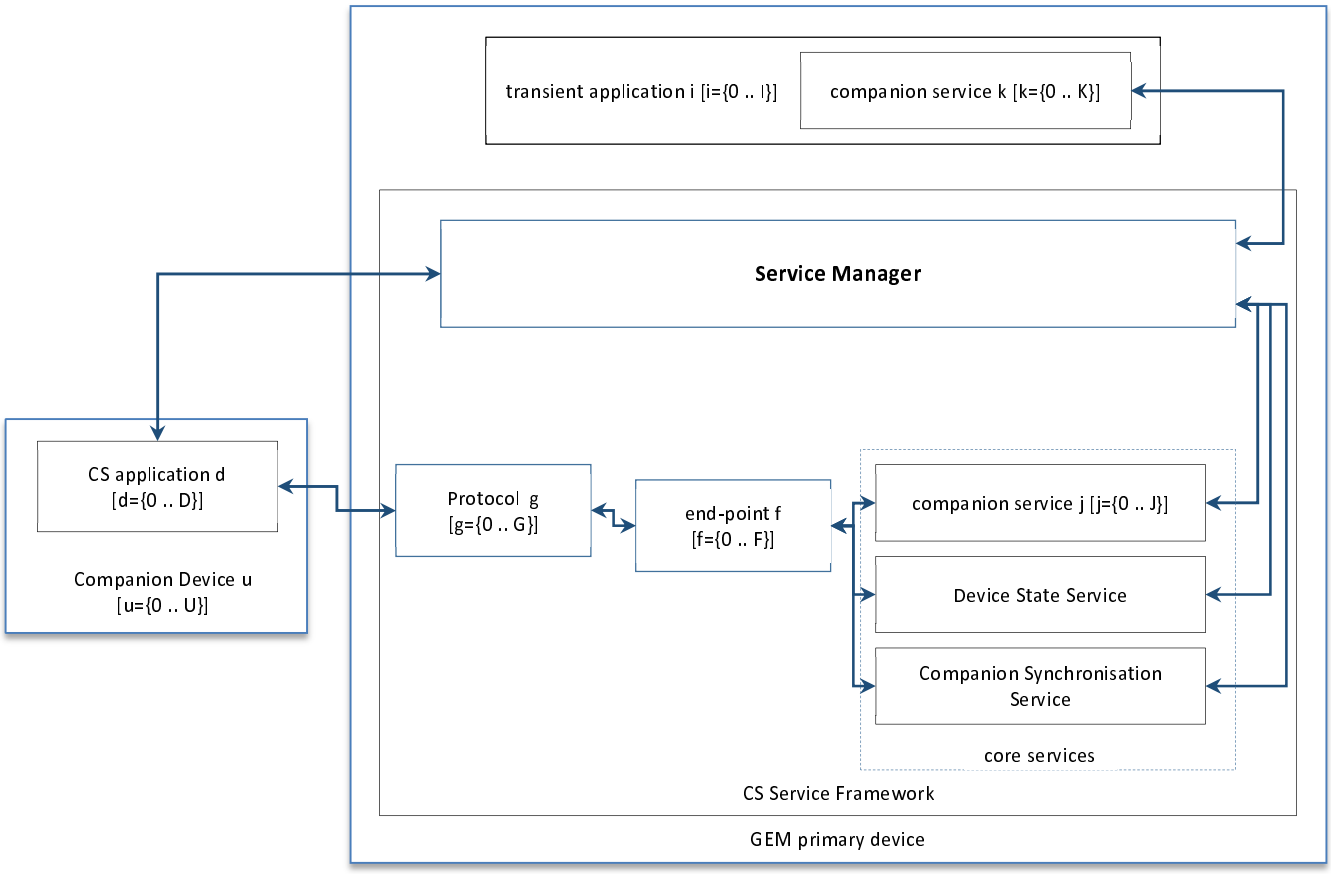

Figure 3: Framework architecture

In order to address use cases requested by the commercial requirements for Phase 1 and Phase 1+ the framework defines a couple of *core services*, that are described in clauses 7.1 and 7.2 of the present document. Such services expose to companion screen application information which can be grouped as follow:

- Device Status Information

  Status Information is handled by the *Device State Service*, which enables a companion screen application to be aware about:

  - status of the content played upon the primary device, e.g. service name;
  - status of the primary device, e.g. list of GEM applications.

- Content Synchronization Information

  Synchronization Information is handled by the *Companion Synchronization Service*, which enables a companion screen application to get information related to the timeline of content presented on the primary device.

These services are *Core Services*, which are included in the platform; they provide synchronous and asynchronous communication interfaces between primary device and companion screen applications.

5 GEM Companion Service Model

5.1 GEM Companion Service

5.1.0 Overview

A *Companion Service* is a software component that manages a set of state variables (resources). These state variables can change over the lifetime of the service and represent a model of the functionality provided by the service.

State variables are strings. They have a type and a value that can be queried or changed via operations.

Execution of a command can be triggered via a method call on the external API. The external API can be called directly from a Java application or is used by a protocol endpoint via a protocol handler. This exposes the external API to Companion Screen Applications.

A Companion Service provides one or more protocol endpoint to offer an external interface to companion screen applications via a specific protocol.

The Companion Service model supports dedicates service types that enable notifications on changes to the service state. This allows registered subscribers to be notified of a change of the state of the service, i.e. a change to one or more of the state variables.

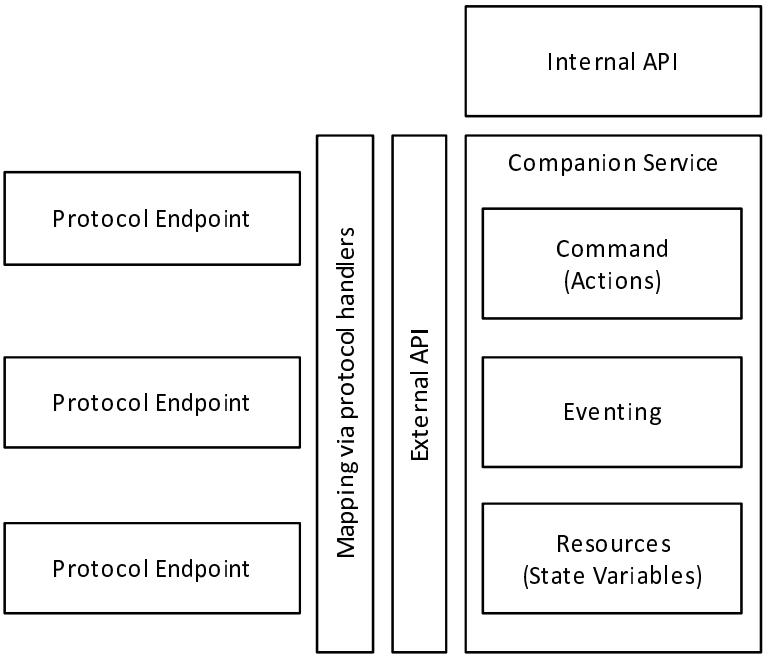

Figure 4: Companion Service Model

5.1.1 Service Types

The GEM companion screen service framework defines the following service types with appropriate protocol handlers:

- REST.
- WebSocket.
- CSS.

An application can use these service types and benefit from the underlying protocol support or it can implement proprietary protocols by defining its own protocol endpoints and protocol handlers.

5.1.2 Publishing Services

A service is available to applications and services within the platform. To make it usable for external clients, it has to be discoverable via a common discovery mechanism and it has to provide one or more protocol endpoints that are used for communication with external clients.

The discovery mechanism is defined in clause 6.

The framework does not put obligations on the protocols that can be used on a protocol endpoint to communicate with companion services but provides a common mechanism to integrate various protocols.

5.1.3 Protocol Endpoints

A *protocol endpoint* terminates an external network communication interface in the GEM platform. An external protocol is mapped by the platform to API calls of the service interface via a protocol handler. Protocol handlers are internal to the platform and provide a bridge between an external communication protocol (terminated with an IP address and a port number) and a service interface. The GEM companion screen service framework provides protocol support for REST over HTTP/HTTPS, the CSS protocols and WebSockets.

5.1.4 Resources and State variables

A Companion Service encapsulates a set of resources (state variables) and provides a set of commands to get or set the resource values.

All Companion Service resources are of type String.

NOTE: A service may choose to encode structured information in the String by using JSON IETF RFC 7159 [10] notation.

A Companion Service can provide information to the interested companion screen applications in an asynchronous way about change in the state some of its resources. This mechanism is called *notification* and is described in clause 5.1.5.

5.1.5 Notifications

The notification mechanism allows companion screen applications to receive messages about companion service's events (CS-event) in an asynchronous way. The primary device sends messages to companion device as soon as the corresponding CS-event happens, without any explicit request.

Notifications are sent using WebSocket protocol IETF RFC 6455 [5].

When a Companion Service is starting it shall create and open a WebSocket server and include its connection address in the AppToAppInfo section that is used during the discovery phase (see clause 6.2). There are no restrictions in a type of connection, i.e. the connection can be secured (wss://) or unsecured (ws://).

When a value of an evented resource changes then a Companion Screen sends a corresponding message using WebSocket channel. The payload format is defined in clause 7.1.1.

A companion screen application interested in events connects to a WebSocket address acquired during discovery phase and listen for messages.

When the Companion Service is deregistered, WebSocket channel is closed and events are no longer available.

5.1.6 Naming of Companion Service

All Companion Services carry a unique name. Companion Services shall be named according to the definitions in ISO/IEC 29341 [3] and clause 6.2 of the present document.

Names of a Companion Service and its matchingProtocolNames are given by the Companion Service creator.

As described by clause 6.2, the name of a Companion Service and its *matchingProtocolName* are used by the CSA to obtain information about the services provided by the primary device.

The `matchingProtocolName` schema is defined by:

```
matchingProtocolName := <service_name>".GEM.DVB.org_v1"
```

The names of platform-provided *Companion Services* are predefined by clauses 7.1 and 7.2 of the present document.

The names of application-provided Companion Services shall follow the same schema.

5.2 Service Manager

5.2.0 Overview

As introduced by Figure 3, companion screen applications are able to access services on the primary device through the Service Manager. It is a software module, which is started at platform startup of the primary device and it remains running while the primary device is on.

Some of the tasks of the Service Manager were already introduced, additional functions include:

1) Service Manager shall manage registration for available Companion Services provided by the GEM platform (static).

2) Service Manager shall manage registration for available Companion Services provided by a GEM application (transient).

3) Service Manager shall manage the state of the Companion Services (lifecycle).

4) Service Manager shall provide all information related to the list of started Companion Services to companion `screen` applications.

5) Service Manager shall manage "private mode", see clause 5.2.2 in the present document.

6) Service Manager shall provide user information about Companion Services available on the primary device.

The Service Manager supports discovery as defined by ISO/IEC 29341 [3] which is described by clause 6.

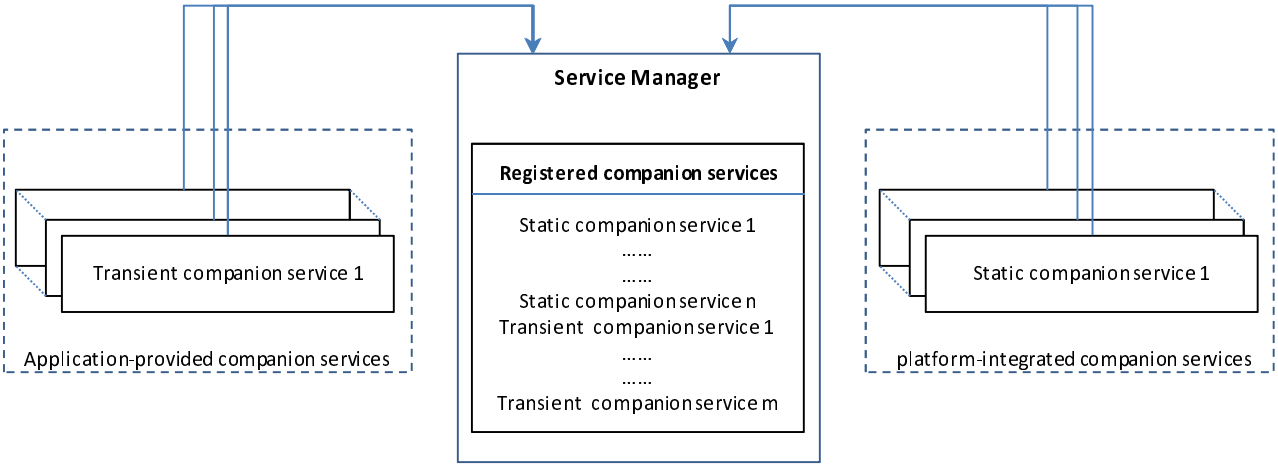

Figure 5: Companion Service registration

There is no difference between the registration and possible functionality between a platform-integrated companion service and application-provided companion service.

Application-provided companion services can add features to the platform. It is not possible that an application-provided companion service replaces a platform-provided companion service.

Application-provided companion services and platform-integrated companion services can be used simultaneously by the same or by multiple companion screen applications.

5.2.1 Service lifecycle

A GEM companion service follows a lifecycle: It is always in one of the following execution states.

Table 1: Service Lifecycle

State	Description
Registered	The service was registered with the service manager
Started	The service is active and accepting requests from the external interface
Paused	The service is on standby and not accepting requests from the external interface
Stopped	The service is stopped and ready to be either started again or to be unregistered
Unregistered	The service is not registered with the service manager

Service transitions are described in figure 6.

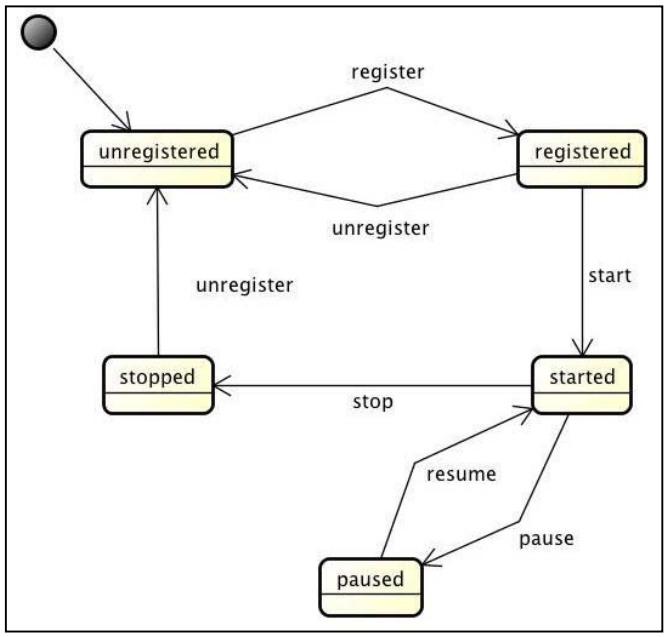

Figure 6: State model for Companion Service

State changes are caused by method calls on the Service Manager.

5.2.2 Private Mode

The Service Manager offers a way to set the Private Mode state, where all companion services are paused and do not respond to requests from companion screen applications. All incoming requests during the Private Mode to the Companion Services are ignored.

When the Private Mode is set, the Service Manager is able only to manage request from the user to exit from the Private Mode.

Companion Screen services, that are registered and started during the Private Mode, are automatically paused. Event notifications are not sent until leaving the Private Mode.

During the Private Mode, the Service Manager returns an empty list of started Companion Screen services.

Only the user can set/unset Private Mode by means of a dedicated UI provided by the GEM Primary Device; for this purpose the Companion Screen Service Framework shall provide APIs.

Such APIs act upon all registered Companion Services, i.e. the user has no access to each single registered Companion Service.

5.3 Service deployment

Platform-integrated Companion Services are registered during system initialization and are available during the entire up time of the platform. Application-provided Companion Services are transient, i.e. they are only available while their corresponding application is available, e.g. during the presentation of a TV service. They are registered by an application during its initialization and shall be deregistered by the application before the application terminates.

Transient Companion Services are packaged and deployed in complete agreement to lifecycle of the application which register them. For details refer to the application model for GEM which is defined in ETSI TS 102 728 [1], clause 9. There is no specific signalling or dedicated application type for applications that provide Companion Services.

A service is started by the application and is available as long as the application is available. When an application that provided a service is terminated, it shall stop and unregister all CS-services that were created by it.

NOTE: To protect the integrity of the companion screen service framework, a platform may clean up services that were not properly stopped or terminated by the application after the creating application has terminated.

6 Discovery and Association

6.0 Overview

The present document supports two operation variants to enable a companion screen application consuming services on the primary device.

The first mode assumes a preconfigured system, i.e. the linkage between a companion screen application and its companion services happens without discovery. This mode is further described in clause 6.1 in the present document.

The second mode uses UPnP Application Management Service to dynamically discover and announce companion services. This mode is further described in clause 6.2 in the present document.

It is out of scope of the preset document to define a mechanism for companion screen application authentication and/or authorization; this is left to the implementation of application-provided companion services.

6.1 Service use without discovery (Phase 1)

To be able to use a service without discovery it is assumed that the devices have already been associated, i.e. they have a way to communicate with each other over the home network. This means that each companion screen application has been provided with sufficient information to use the services and corresponding protocol endpoints of the primary device without discovery.

This could be achieved by manual configuration of the companion screen application or by out of band discovery (e.g. communication via QR-codes).

6.2 Service Discovery on the local network (Phase 1+)

6.2.0 Dynamic service discovery

The use of services as described in the previous clause has certain drawbacks and limitations. This clause describes the dynamic discovery and association of GEM companion services over the home network using the UPnP Application Management Service.

Clause 6.2.1 and subclauses contain an informative description on how the UPnP application management service is integrated into the GEM companion service model.

The normative requirements on the GEM UPnP application management are described in clause 6.2.2.

NOTE: In the context of clause 6.2 the term application and Application Management refers to UPnP applications.

6.2.1 Introduction (informative)

6.2.1.0 Outline

Universal Plug and Play (UPnP) is a well known and established protocol to find and identify UPnP-enabled devices on the network. UPnP is used to convey the information to enable the CSA to discover and associate with the GEM Device as depicted in Figure 7, and thereby establish communication, See also ETSI TR 103 286-1 [i.2].

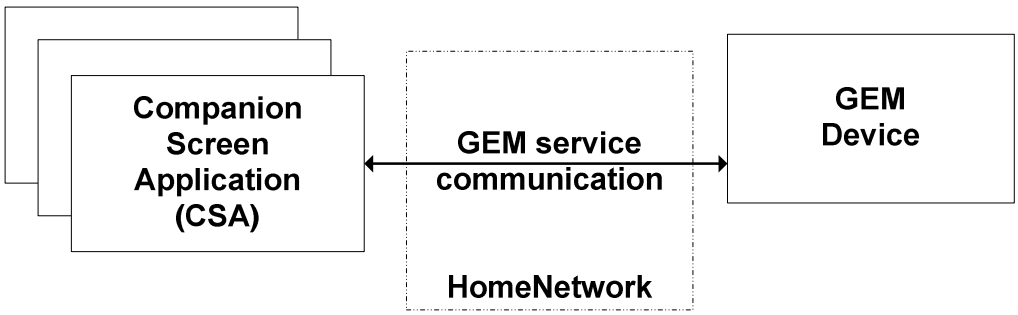

Figure 7: CSA and GEM Device connected to home network

The following clauses provide an informative descriptive overview of the discovery functionality in the light of the device requirements defined in clause 5.

Clause 6.2.1.1 provides a brief overview of the main elements of the UPnP Device Architecture.

Clause 6.2.1.2 describes UPnP Application Management that the present document profiles the implementation of by the GEM Device.

Clause 6.2.1.3 explains how the location of the endpoint for the GEM application communication is exposed through UPnP Application Management.

Clause 6.2.1.4 illustrates the sequence of operations that a CSA can use to discover and associate with the GEM Device and thereby determine the location of the GEM companion service communication endpoint.

6.2.1.1 UPnP Device Architecture

UPnP implements a client server model. In UPnP terminology clients are called Control Points and servers are called UPnP Devices. The functionality of UPnP Devices and Control Points is abstracted from the transport mechanism. The functionality is described in Device Control Protocols (DCPs) that define a collection of Actions (RPC methods) and Events. The transport mechanism is described in the UPnP Device Architecture (UDA) [1] and the components that make up the UDA are illustrated in Figure 8. The UDA describes:

- how Control Points can detect UPnP Devices on the network without prior knowledge of the device IP address using the Simple Service Discovery Protocol (SSDP) that takes place via UDP;
- which capabilities the detected UPnP Devices implement (by means of downloading an XML Device Description Document and XML Service Control Protocol Description document by means of HTTP);
- how Control Points interact with the exposed functionality by invoking actions using SOAP requests via HTTP; and
- how Control Points receive events using GENA via TCP.

The domain specific functions are described per domain in DCPs. The DCPs are specified as UPnP Services in a UPnP Device.

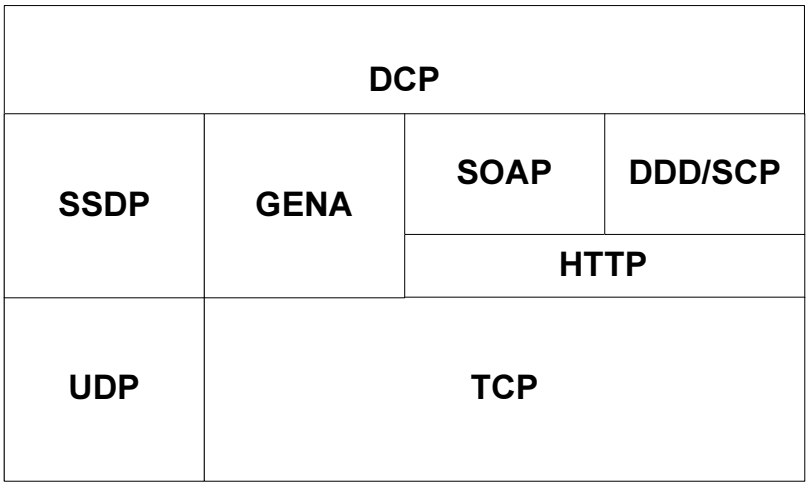

Figure 8: Schematic overview of a UPnP Stack

6.2.1.2 UPnP Application Management

The UPnP Application Management service DCP (ETSI TS 103 286-2 [2]) is used to establish the communication between the CSA and the GEM Device. The GEM Device implements a UPnP Device that incorporates the Application Management service and the CSA implements an Application Management Control Point, this is illustrated in Figure 9.

NOTE: Application Management is a UPnP term and refers to a service for managing abstract entities known as applications. An application is defined by individual profiles and implementations of the Application Management service DCP. Therefore it should not be confused with the management of applications that run on a companion device.

What follows is a brief overview of the Application Management service. How this service is used to expose the location of GEM companion service communication endpoint is described in clause 6.2.1.4.

The UPnP Application Management service specification describes functionality to establish App-to-App communication. The Application Management service has mechanisms to:

- determine which applications are supported by the implementation;
- determine which applications are currently running;
- retrieve detailed information about an application, including which kind of app-to-app protocols are supported by each application;
- retrieve the information needed to establish the Out-of-Bound communication for a given app-to-app protocol.

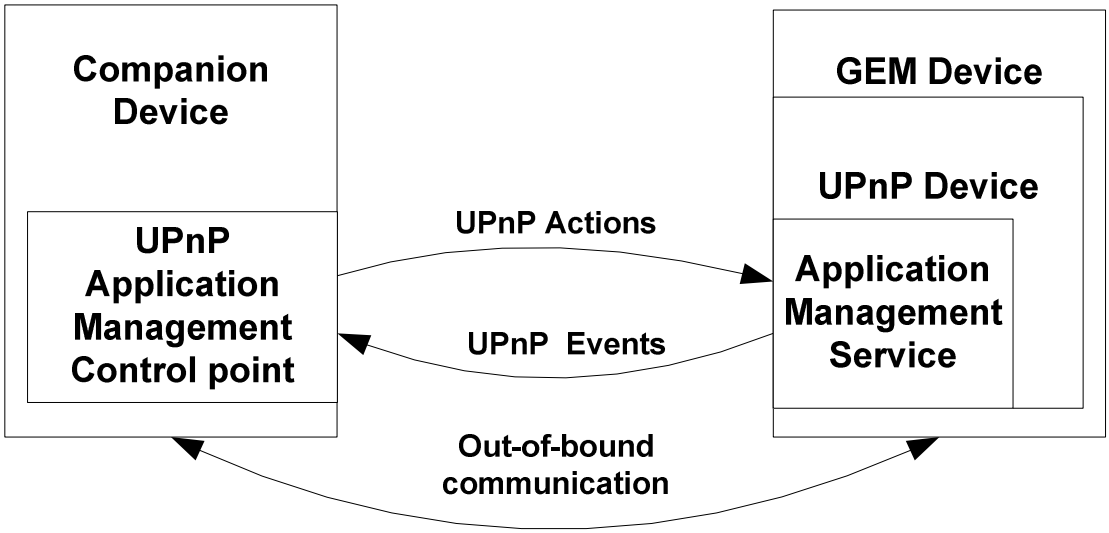

Figure 9: Typical implementation of an Application Management Control Point in a Companion Device and the UPnP Application Management service in a GEM Device

6.2.1.3 GEM companion service mapping on Application Management.

The CSA acts as an Application Management Control Point. This will allow the CSA to detect the GEM Device that will implement the UPnP Device that implements the Application Management service DCP. This is depicted in Figure 10.

The Application Management service in the UPnP Device within the physical GEM Device will lists a GEM companion services applications. These applications will be identified by means of the matchingProtocolName.

The GEM companion service communication interface is discovered by finding the application with the correct matchingProtocolName in any Application Management service. The GEM companion service communication interface can be REST and/or WebSocket based.

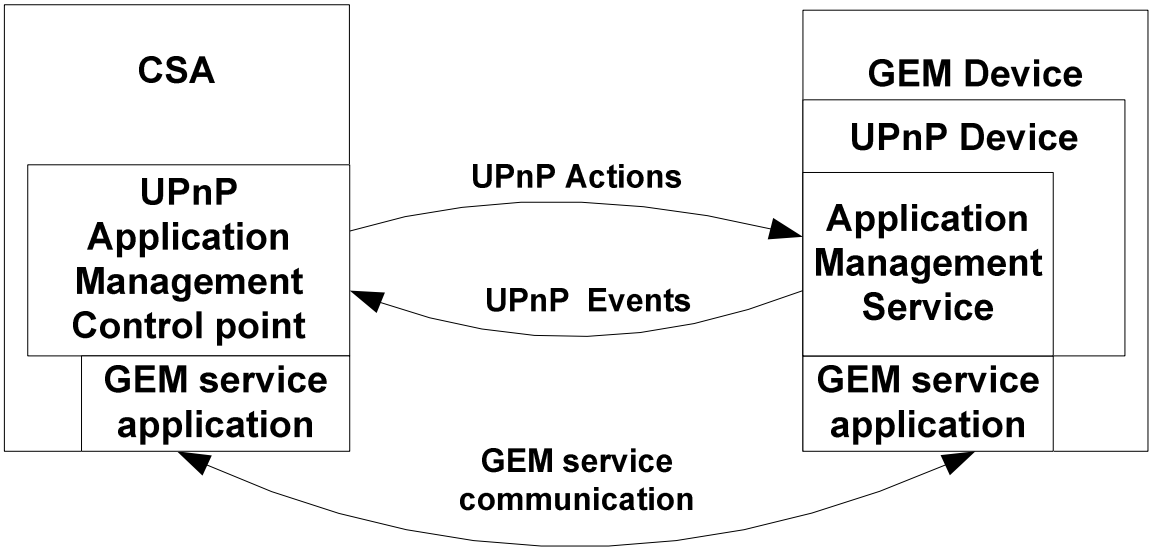

Figure 10: CSA and GEM Device implementing UPnP Application Management and GEM companion service application communication

6.2.1.4 CSA behaviour

The determination of the GEM companion service end point address from a CSA towards a GEM Device is achieved by the following sequence of steps as depicted in Figure 11:

1) Finding the GEM Device on the home network, by issuing an M-Search.

2) Selecting a GEM Device by filtering the responses of the M-Search.

3) Downloading the DDD of the selected GEM Device.

4) Issuing Actions to determine:

   a) If an application exist with matchingProtocolName by invoking Action GetAppIDList().

   b) Retrieve the application information that contains the endpoint address and the runningState by invoking action GetAppInfoByIDs().

5) Establish the GEM companion service application to application connection.

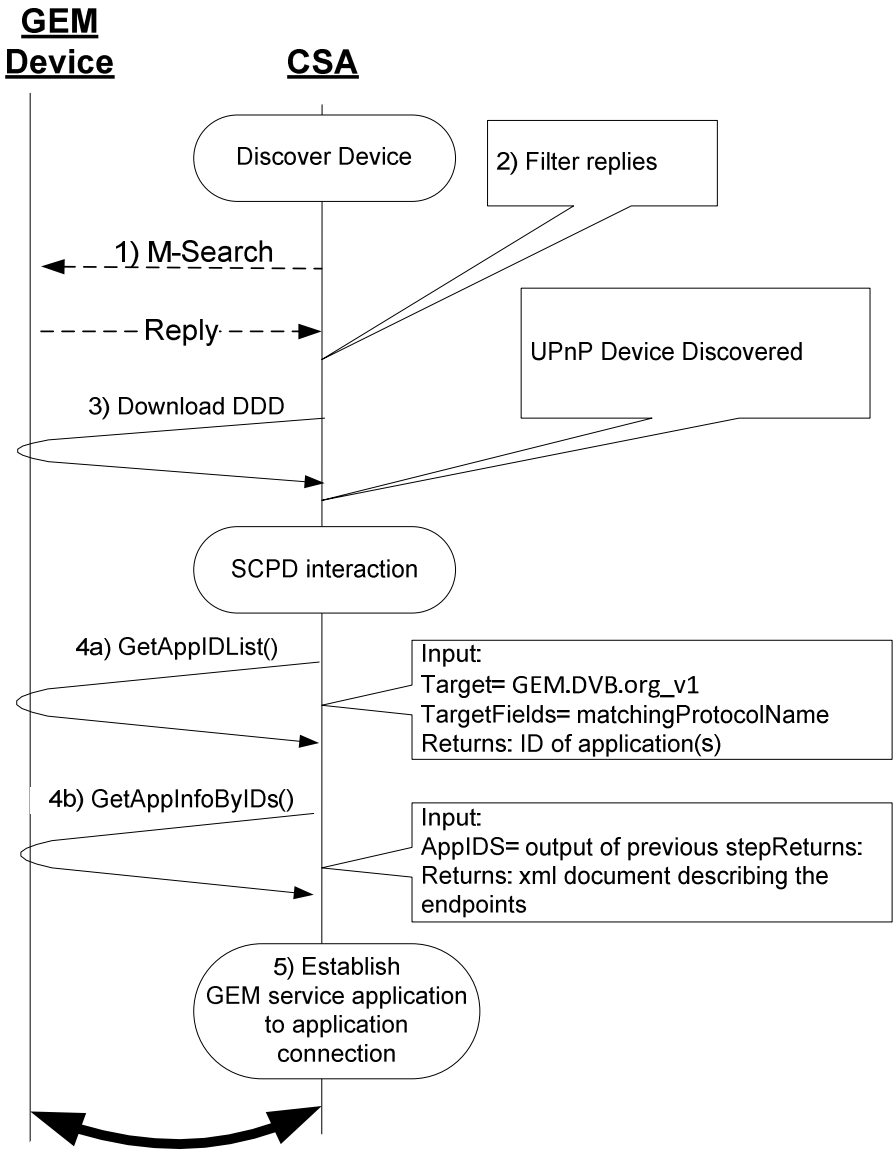

Figure 11: Sequence of commands between CSA and GEM Device to establish a GEM companion service application to application connection

It is not known whether the CSA or the GEM Device started first on the network. Therefore, typically, the CSA implements two mechanisms to detect the GEM Device:

1) The CSA can monitor ssdp:alive messages for the Application Management service. This allows the GEM Device to be detected by the CSA when the GEM Device is entering the network and the CSA has already entered the network.

2) The CSA can search for the Application Management service when the CSA enters the network and find already started GEM Devices.

The Application Management service implemented by any UPnP Device type can be found by issuing an M-Search with one of the search targets defined as one of the following:

- "ssdp-all", this will include also the replies of each implemented service;
- "urn:schemas-upnp-org:service:ApplicationManagement:1", this will result in only in replies of UPnP Devices that implement this particular service. Note that higher versions of the service will be found as well.

The Control Point can filter the SSDP responses on the Application Management service identifier to find the UPnP Device that contains the Application Management service.

The Control Point can find a previously found GEM Device on the network by issuing an M-Search with the UDN as search target. The UDN is conveyed in the DDD and remains the same over reboots of a UPnP Device. To associate a previous selected TV Device the UDN of the GEM Device needs to be stored persistently by the CSA. This will enable a CSA to provide its user with the facility to automatically reconnect to, for example, the GEM Device they selected the previous time they used the CSA.

The Control Point uses the value of the location header of the SSDP reply or alive message to download the DDD. The DDD contains the UPnP controlURL for each implemented service. The Control Point may use the UPnP controlURL of the Application Management service to invoke actions on the Application Management service implemented in the UPnP Device.

The Control Point uses the Actions of the Application Management service to find the applications identified by the substring matchingProtocolName "GEM.DVB.org_v1" which identifies all GEM companion service based applications and verifies that the found applications has the runningState "Running". These preconditions need to be met before establishing the GEM companion service application connection for the GEM companion services communication interfaces as described in clause 6 of ETSI TS 103 286-2 [2] by using the end point address supplied by the application's connectionAddress.

In the following example the CSA Control Point uses the GetAppIDList() action to determine if the Application Management service has implemented version 1 of a GEM companion service application. The implementation hosts 2 applications as with "GEM.DVB.org_v1" as substring of matchingProtocolInfo and has application ids "110E4EC1-6CC4-4D12-9995-7F996B709727" and "110E4EC1-6CC4-4D12-9995-7F996B709728".

Request:

```
GetAppIDList(Target="GEM.DVB.org_v1", TargetFields="matchingProtocolName")
```

Response:

```
GetAppIDList(AppIDs = "110E4EC1-6CC4-4D12-9995-7F996B709727,110E4EC1-6CC4-4D12-9995-7F996B709728")
```

The returned AppIDs are used as input for the GetAppInfoByIDs() action to retrieve the XML description of each returned application id. In this case 2 application ids are returned.

Request:

```
GetAppInfoByIDs(Target="110E4EC1-6CC4-4D12-9995-7F996B709727,110E4EC1-6CC4-4D12-9995-7F996B709728")
```

Response:

```
GetAppInfoByIDs(AppInfo =
   "<?xml version="1.0" encoding="UTF-8"?>
   <AppInfoList
    xmlns="urn:schemas-upnp-org:ms:ams" xsi:schemaLocation="
     urn:schemas-upnp-org:ms:ams">
    <application id="110E4EC1-6CC4-4D12-9995-7F996B709727">
        <friendlyName>GEM Companion Service Application</friendlyName>
        <runningStatus>Running</runningStatus>
        <apptoAppInfo>
            <matchingProtocolName>myserviceapp1.GEM.DVB.org_v1</matchingProtocolName>
            <protocol required="1">REST</protocol>
            <connectionAddress>https://192.168.0.1:8081/MyDVBGEMServiceApp</connectionAddress>
        </apptoAppInfo>
    </application>
   <application id="110E4EC1-6CC4-4D12-9995-7F996B709728">
        <friendlyName>GEM Companion Service Application 2</friendlyName>
        <runningStatus>Running</runningStatus>
        <apptoAppInfo>
            <matchingProtocolName>myserviceapp2.GEM.DVB.org_v1</matchingProtocolName>
            <protocol required="1">REST</protocol>
            <connectionAddress>https://192.168.0.1:8082/MyDVBGEMServiceApp2</connectionAddress>
        </apptoAppInfo>
    </application>

   </AppInfoList>")
```

The returned XML document contains 2 found GEM companion service applications in the running state. The first application is identified with id "*110E4EC1-6CC4-4D12-9995-7F996B709727" and has a matchingProtocolName defined as* "myserviceapp1.GEM.DVB.org_v1" with an REST endpoint at "*https://192.168.0.1:8081/MyDVBGEMServiceApp*". The second application is identified with id "*110E4EC1-6CC4-4D12-9995-7F996B709728" and has a matchingProtocolName defined as* "myserviceapp2.GEM.DVB.org_v1" with an REST endpoint at "*https://192.168.0.1:8082/MyDVBGEMServiceApp2*".

In the following example the CSA Control Point uses the GetAppIDList() action to determine if the Application Management service has implemented version 1 of any DVB application as defined as: "DVB.org_v1".

Request:

```
GetAppIDList(Target="DVB.org_v1",TargetFields="matchingProtocolName")
```

Response:

```
GetAppIDList(AppIDs = "110E4EC1-6CC4-4D12-9995-7F996B709726,110E4EC1-6CC4-4D12-9995-
7F996B709727,110E4EC1-6CC4-4D12-9995-7F996B709728")
```

The returned AppIDs are used as input for the GetAppInfoByIDs() action to retrieve the XML description of each returned application id. In this case 3 applications are returned.

Request:

```
GetAppInfoByIDs(Target="110E4EC1-6CC4-4D12-9995-7F996B709726,110E4EC1-6CC4-4D12-9995-
7F996B709727,110E4EC1-6CC4-4D12-9995-7F996B709728")
```

Response:

```
GetAppInfoByIDs(AppInfo =
   "<?xml version="1.0" encoding="UTF-8"?>
   <AppInfoList
    xmlns="urn:schemas-upnp-org:ms:ams" xsi:schemaLocation="
     urn:schemas-upnp-org:ms:ams">
    <application id="110E4EC1-6CC4-4D12-9995-7F996B709726">
        <friendlyName>TV Device CSS-CII DVB Application</friendlyName>
        <runningStatus>Running</runningStatus>
        <apptoAppInfo>
            <matchingProtocolName>CSS-CII.TVDevice.CSS.DVB.org_v1</matchingProtocolName>
            <protocol required="1">Websocket</protocol>
            <connectionAddress>ws://192.168.0.1:80/MyCSSCIIDVBApp</connectionAddress>
        </apptoAppInfo>
    </application>
   <application id="110E4EC1-6CC4-4D12-9995-7F996B709727">
        <friendlyName>GEM Companion Service Application</friendlyName>
        <runningStatus>Running</runningStatus>
```

```
        <apptoAppInfo>
            <matchingProtocolName>myserviceapp1.GEM.DVB.org_v1</matchingProtocolName>
            <protocol required="1">REST</protocol>
            <connectionAddress>https://192.168.0.1:8081/MyDVBGEMServiceApp</connectionAddress>
        </apptoAppInfo>
    </application>
    <application id="110E4EC1-6CC4-4D12-9995-7F996B709728">
        <friendlyName>GEM Companion Service Application 2</friendlyName>
        <runningStatus>Running</runningStatus>
        <apptoAppInfo>
            <matchingProtocolName>myserviceapp2.GEM.DVB.org_v1</matchingProtocolName>
            <protocol required="1">REST</protocol>
            <connectionAddress>https://192.168.0.1:8082/MyDVBGEMServiceApp2</connectionAddress>
        </apptoAppInfo>
    </application>

    </AppInfoList>")
```

The returned XML document contains a CSS-CII application in the running state that contains a web socket address ws*://192.168.0.1:80/MyCSSCIIDVBApp and the 2 previously found GEM companion service applications.*

NOTE: For readability the XML in the supplied examples above is not XML escaped.

6.2.2 UPnP Device requirements for the GEM Device

The GEM Device shall implement a UPnP Device that implements the Application Management service DCP. The implemented UPnP Device shall comply with UPnP Device Architecture ISO/IEC 29341-1:2011 [8] or a later version of the UDA.

NOTE 1: The Application Management service can reside in any UPnP Device Type. It is up to the vendor of the UPnP Device to choose an appropriate UPnP Device type. The implemented UPnP Device is compliant with UPnP Device Architecture ISO/IEC 29341-1:2011 [8] or a later version of the UDA.

The Application Management service shall comply with ISO/IEC 29341 [3] or a later version of this service. This means that the mandatory actions and state variables in the Application Management service shall be implemented.

The GEM companion service with endpoints shall be presented as a (UPnP) application in the Application Management service.

The following requirements shall apply for GEM companion service applications:

- The application shall provide an <apptoAppInfo> element within XML documents describing the application where:
  - the matchingProtocolName shall be defined as "<ApplicationIdentification>.GEM.DVB.org_v1"
    - ApplicationIdentification is free text to be supplied by the application vendor
  - the protocol shall be defined either as "REST" or as "Websocket"
  - the value of the protocol required attribute shall be defined as "1"
  - for protocol value defined as "Websocket" the connectionAddress shall be a valid WebSocket address
  - for protocol value defined as "REST" the connectionAddress shall be a valid HTTP or HTTPS address
- The connectionAddress shall point at the protocol end point as defined in clause 6.3 of ETSI TS 103 286-2 [2] and shall convey the GEM companion service protocol as defined in clause 6.2 of ETSI TS 103 286-2 [2].
- The application shall always have the runningState of "Running".
- When the implementation implements the action StopApp(), the application shall not be stopped when the action StopApp() is invoked. This function shall return error code 710.

NOTE 2: It is not required to implement StopApp().

- The XML document describing the application as result of invoking the action GetAppInfoByIDs() shall always include the properties:
  - runningStatus
  - apptoAppInfo

7 External Service Interface

7.0 Services

As stated in clause 5.1.1, the GEM Companion Screen Service framework defines the following Companion Services:

- REST.
- WebSocket.
- CSS.

This clause describes how these protocols are used with the Companion Screen Service Framework.

7.1 REST Companion Service

7.1.0 Overview

A companion service, which can be used by REST over HTTP or HTTPS, offers a REST-ful external interface; refer to IETF RFC 2616 [4], IETF RFC 2818 [9] and Principled Design of the Modern Web Architecture [i.1].

The REST-ful companion service interface defines four methods:

- GET.
- PUT.
- POST.
- DELETE.

They cover all the actions a companion screen application can perform over the corresponding companion service.

7.1.1 Notifications

The notification mechanism allows companion screen applications to receive messages about events of a companion service in an asynchronous way.

The companion framework sends a notification message to the companion device as soon as the corresponding event happens.

The message shall be in the form of:

```
"{"<ResourceName>":"<Value>"}"
```

A REST companion service provides notifications using the WebSocket protocol IETF RFC 6455 [5] as described in clause 5.1.5 in the present document.

7.1.2 JSON Data Format

The Companion Screen Applications and REST Companion Services are exchanging data in the JSON data interchange format. All payloads in request/response messages are JSON-encoded.

In addition notifications are using JSON for the payload.

The GEM Platform includes a JSON API with the API semantics defined by JSON in Java [11].

This API can be used by platform-integrated and application-provided companion services for other communication protocols.

7.1.3 REST Companion Service API

The GEM Companion Service Framework includes a simple REST API that can be used over HTTP and HTTPS following the REST concepts as described in Principled Design of the Modern Web Architecture [i.1].

The web APIs for REST Companion Services in the present document are described via the following template:

```
request = method resourceURI action [payload]
```

and

```
response = return-code [payload]
```

`action`, `key` and `payload` are Companion Service dependent.

`Payload` shall be a JSON document.

The interpretation of the payload is performed by each companion service, where platform APIs such as a JSON decoder may be used to parse it. Similarly the response payload is created by each companion service by means a JSON encoder. It means both Response and Request are JSON string.

To get available resource URIs, a Companion Screen Application has to send a GET request to a Companion Screen Service with the root resourceURI, i.e. URI defined in the connectionAddress inside the AppToAppInfo section retrieved from the UPnP Application Management System. The payload of the response carries a comma-separated list of resource-names.

Here is the list of permitted methods, together to their meaning in the scope of the present document.

method	Meaning
GET	the GET method means retrieve information from the Primary Device within the specified REST Companion Service
PUT	the PUT method requests that the parameters be stored by the specified REST Companion Service
POST	the POST method is used to request the Primary Device to accept parameters enclosed in the request as data to be processed within the specified REST Companion Service
DELETE	the DELETE method requests that specified REST Companion Service delete the resource identified by request parameters

Here below the list of admitted return-code, together to their meaning in the scope of the present document.

return-code	Meaning
200	success - the request has succeeded, and an entity corresponding to the requested resource is sent in the response
400	bad request - the request has a malformed syntax
404	not found - the REST Companion Service which should execute the request is not present
500	internal server error - the server encountered an unexpected condition which prevented it from fulfilling the request
503	Service Unavailable: the REST Companion Service which should execute the request is not able to complete it
505	HTTP version not supported

7.2 Web sockets

The GEM companion screen framework contains a WebSocket API and defines a WebSocket service, which can be used to create services that interact over the WebSocket protocol IETF RFC 6455 [5].

7.3 Core Services

7.3.1 Device State Service

7.3.1.0 Overview

The present document introduces the platform-provided REST service described in this clause. As described in clause 6.2 such a service shall be announced during discovery phase, and all data required by companion applications to establish a correct communication shall be provided within such a phase.

The formal name for this service is:

```
matchingProtocolName := “DeviceStateService.GEM.DVB.org_v1”
```

`DeviceStateService` is a REST service with both query and notification mechanisms, refer to clause 7.1.1 within the present document, and provide information about the state of the primary device state.

Figure 12 illustrates the integration of Device State Service within Companion Screen Service Framework.

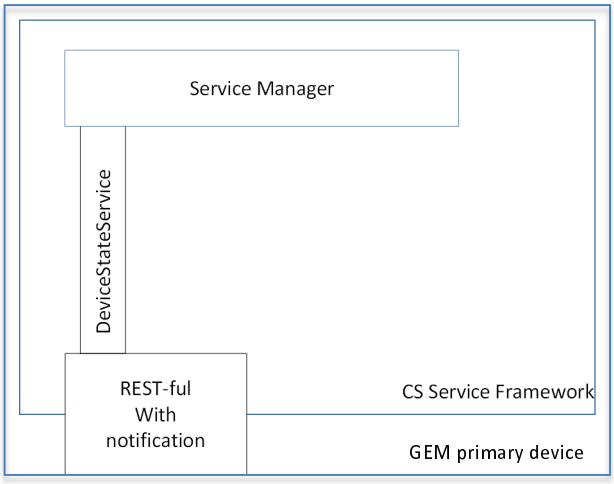

Figure 12: Device State Service

7.3.1.1 APIs for DeviceStateService

Query for information

In order to get information in synchronous way `DeviceStateService` provide following API:

```
method = GET
Action = getDeviceStateInfo
```

Registration to notification mechanism

`DeviceStateService` provide notification mechanism, in agreement to clause 7.1.1 within the present document. It can be started by a companion screen application by means following API:

```
method = POST
Action = subscribeDeviceStateInfo
```

Deregistration to notification mechanism

In order to stop notification a companion screen application shall use following API:

```
method = POST
Action = unsubscribeDeviceStateInfo
```

Information

The `DeviceStateService` provides the following information both as a response to a query and as a notification:

```
payload ="{"
    "video-media-component-type" ":" video_type ","
    "audio-media-component-type" ":" audio_type ","
    "audio-media-component-lan" ":" ISO_639_language_code ","
    "txt-media-component" ":" ISO_639_language_code ","
    "sub-media-component" ":" ISO_639_language_code ","
    *("application-name" ":" application_name ","
    "application-state" ":" application_state ",")
"}"
```

`video_type` = hexadecimal representation of the elementary stream type in the form "0xSSCC", in agreement to the coding defined by "stream content" (SS) and "component type" (CC) in table 26 of ETSI EN 300 468 [6].

`audio_type` = hexadecimal representation of the elementary stream type in the form "0xSSCC", in agreement to the coding defined by "stream content" (SS) and "component type" (CC) in table 26 of ETSI EN 300 468 [6].

`ISO_639_language_code` = refer to ETSI EN 300 468 [6].

`application name` = string of char fields specifying the name of the GEM application, as defined in the AIT in ETSI TS 102 809 [7].

```
application-state = loaded |
                    paused |
                    started |
                    destroyed
```

7.3.2 Companion Synchronization Service (CSS)

The Companion Screen Service Framework provides a platform integrated companion service in order to perform synchronization between a GEM primary device and companion screen applications.

Such a companion service is called Companion Synchronization Service.

As for all Companion Services, the Companion Synchronization Service can be paused and resumed by the User in the private mode. See clause 5.2.2 in the present document.

The Companion Service for Synchronization provides the following functionalities:

- Content Identification: let the companion `screen` application to be able to get information about the content which is currently played upon the primary device;
- Synchronization: let a content played upon a companion `screen` application to be synchronized with to the related one upon the primary device;

The Companion Service for Synchronization is fully compliant to ETSI TS 103 286-2 [2].

The CSS protocol is integrated into GEM as follows:

Figure 13 illustrates the integration of the Companion Synchronization Service in the Companion Screen Service Framework.

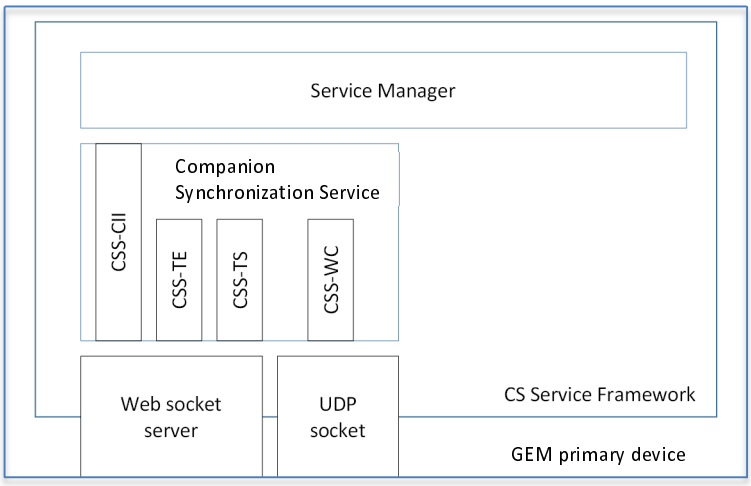

Figure 13: Companion Synchronization Service

The Companion Service for Synchronization registers itself to the Service Manager only for the CSS-CII protocol end point; CSS-CII is the master end point to which the other services belong.

The Companion Synchronization Service shall have the following formal name:

```
matchingProtocolName :=
           "CSS-CII.TVDevice.GEM.DVB.org_v1"
```

The GEM primary device operates only as master with respect to Companion Synchronization Service.

In order to fulfil commercial requirements, the whole set of endpoints introduced by ETSI TS 103 286-2 [2] shall be implemented, with the following clarifications and requirements:

- Endpoint for Content Identification and Information (CSS-CII) shall be implemented as it is the root service for Companion Service for Synchronization.
- Endpoint for Timeline-Synchronization (CSS-TS) shall be implemented at least in agreement to "Simplified scenario" introduced in ETSI TS 103 286-2 [2], clause 4.2.4 and defined in ETSI TS 103 286-2 [2], clause C.5.
- Endpoint for Trigger-Events (CSS-TE) shall be implemented because of requirements for stream-events tracking.
- Endpoint for Wall Clock (CSS-WC) shall be implemented because it is required by (CSS-TS).

  The CSS-WC Server shall listen for Wall Clock Synchronization protocol request messages from CSAs on the IP interface and port number corresponding to the service endpoint.

  The WC Server shall send any response message and any follow-up message back to the same IP address and port number from which the original request was received.

The Companion Synchronization Service is based, with the exception of the endpoint for Wall Clock, on the Web Socket protocol.

8 Framework API

The GEM companion screen service framework is defined in the package `org.dvb.gem.websocket` and `org.dvb.gem.companion` and its sub-packages. The class diagram is shown in Figure 14. The detailed API semantics are defined in the API JavaDoc, which is provided together with the present document.

The JavaDoc defines the normative companion API.

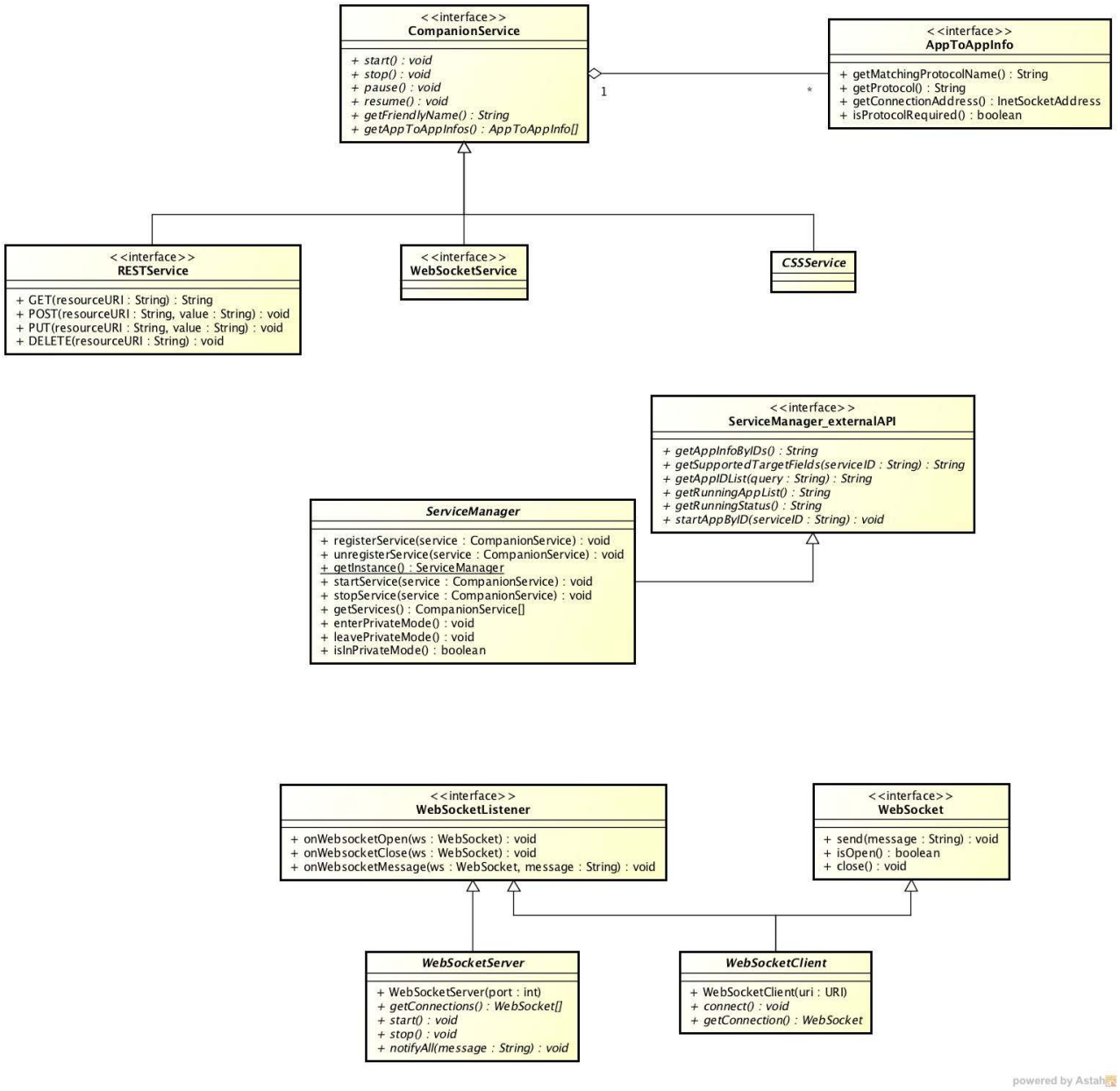

Figure 14: GEM Companion Services Framework

9 API Sequence Diagrams (informative)

9.0 Overview

The sequence diagrams in this clause illustrate usage examples of the GEM companion screen service framework APIs. They are intended to show the conceptual usage of the API and are not intended to illustrate all details.

This clause contains the following usage scenarios:

- 9.1 REST service provided by an application.
- 9.2 CSS Service.
- 9.3 Service with subscription.

9.1 REST service provided by an application

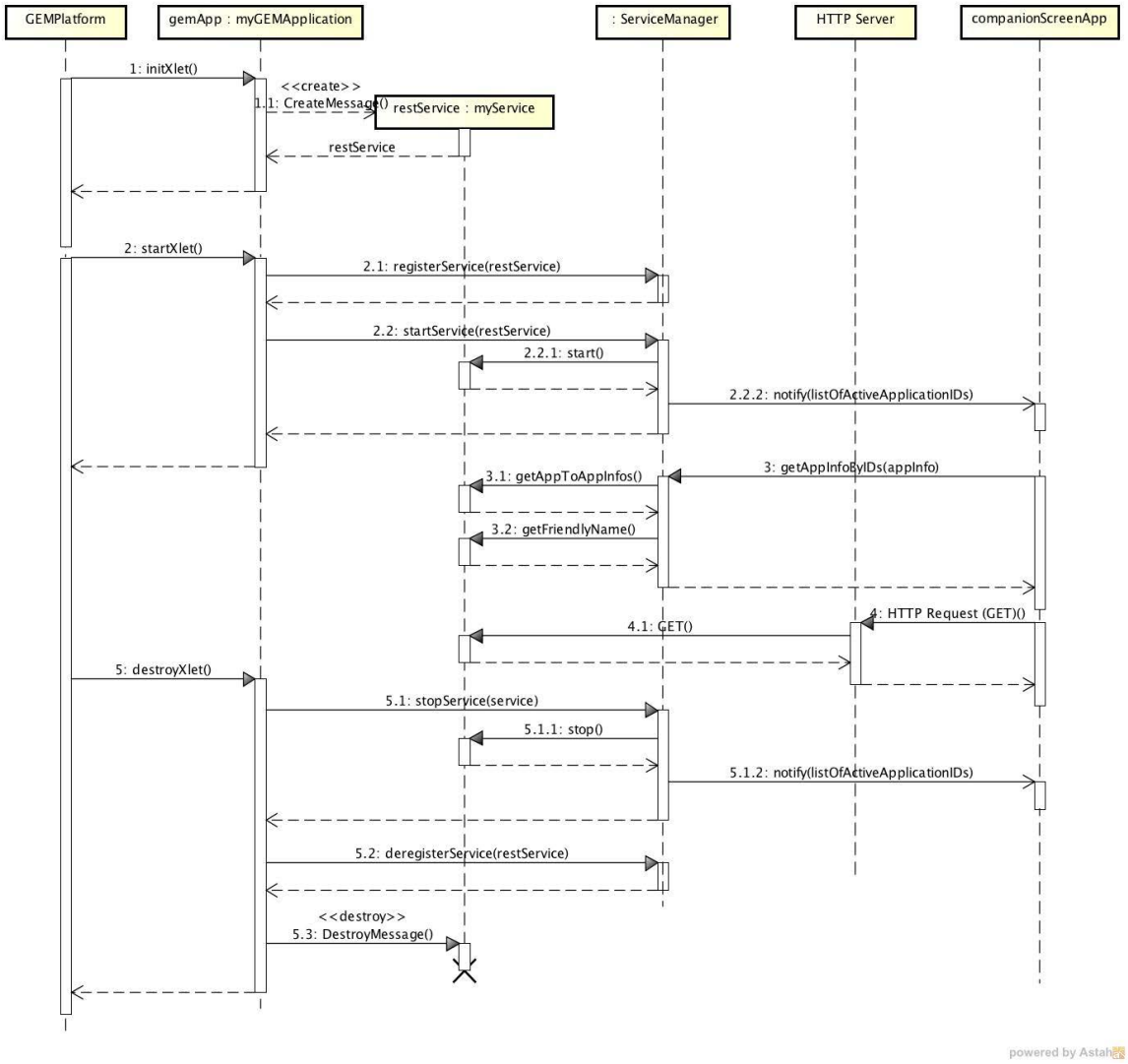

Figure 15: REST Service - start-up and shutdown

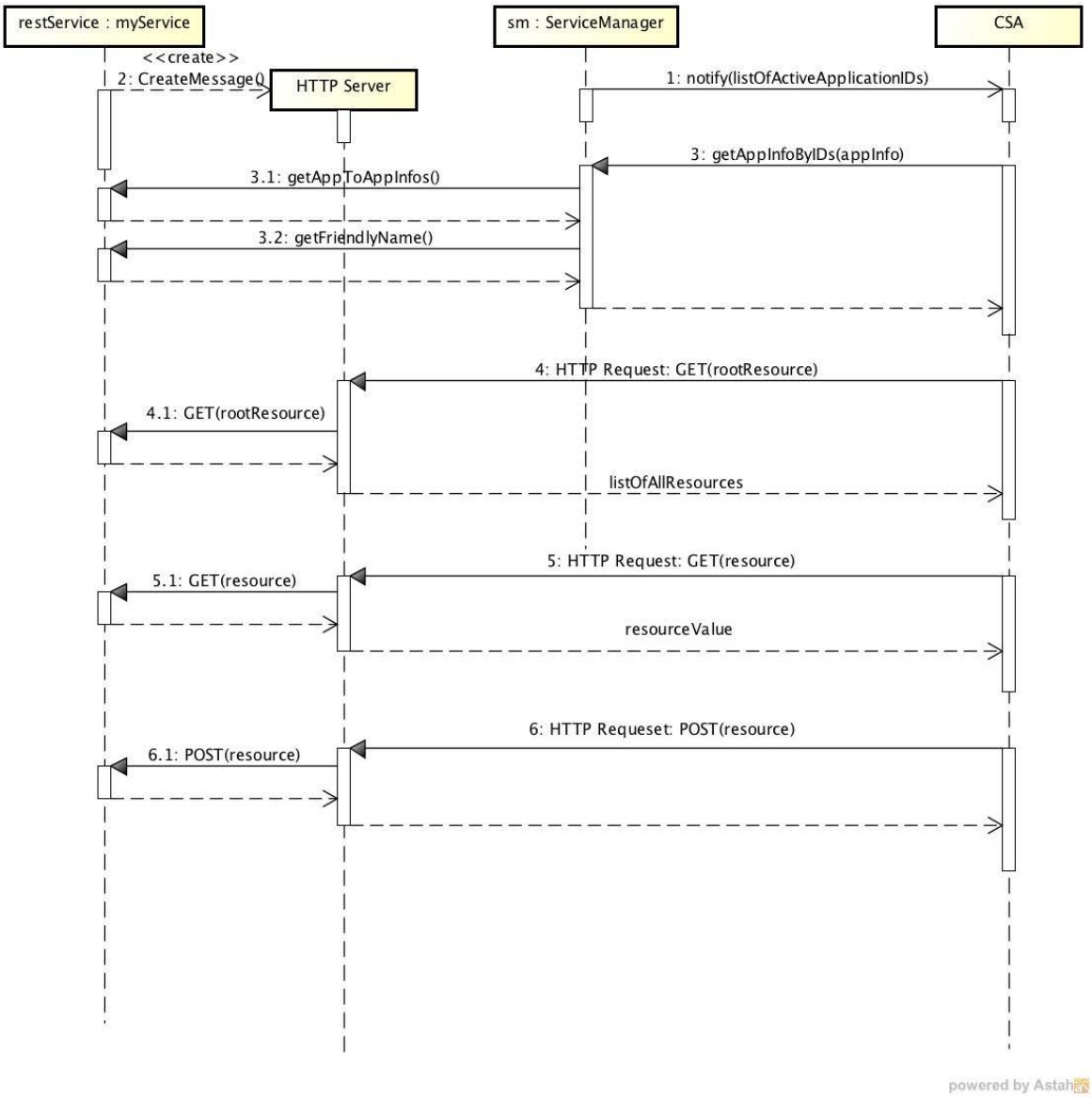

Figure 16: REST Service - usage

9.2 CSS CII Service

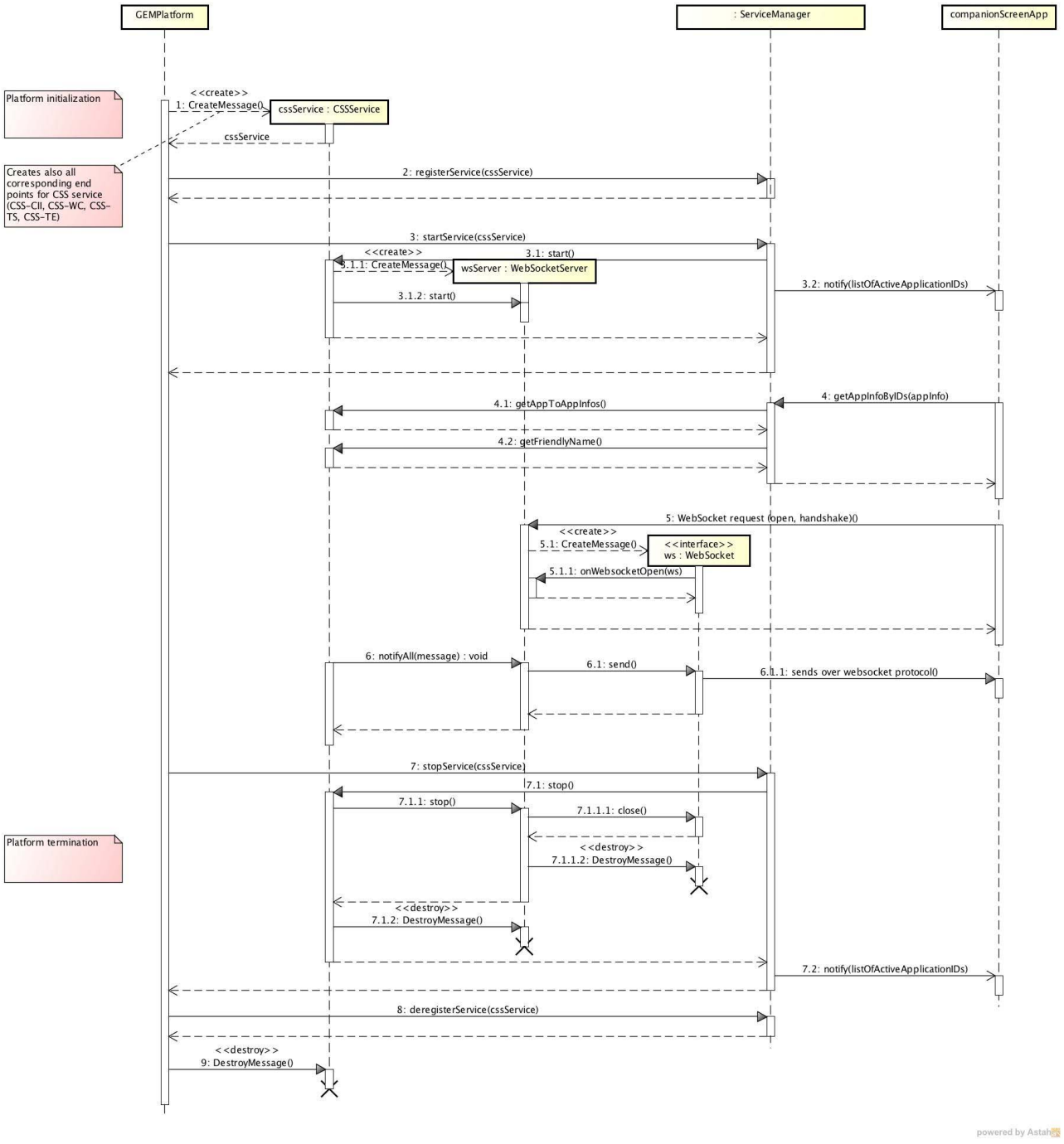

Figure 17: CSS-CII service start-up, usage and shutdown

9.3 Service with subscription

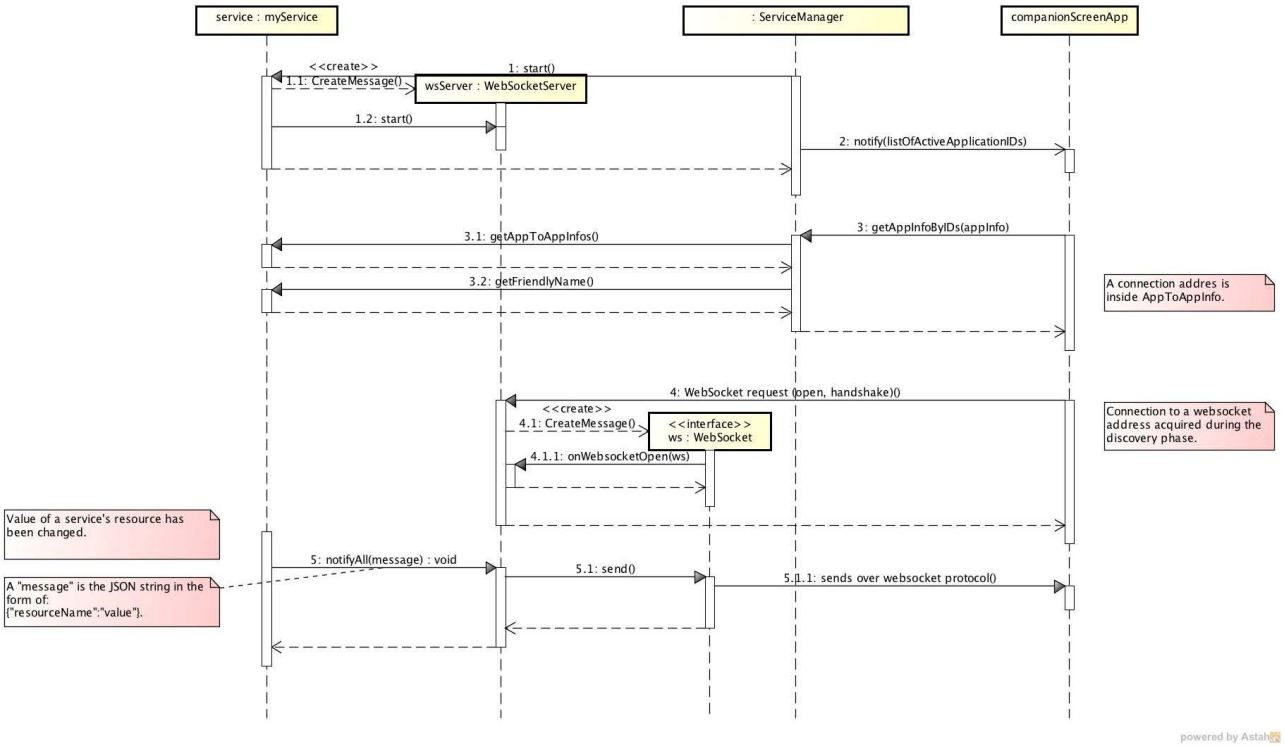

Figure 18: Service with subscription

Annex A (informative): Bibliography

- ISO/IEC 13818-1: "Information technology - Generic coding of moving pictures and associated audio information - Part 1: Systems".

History

Document history		
V1.1.1	May 2015	Publication