I n t e r n a t i o n a l   T e l e c o m m u n i c a t i o n   U n i o n

# ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

**J.1202**
(07/2019)

SERIES J: CABLE NETWORKS AND TRANSMISSION OF TELEVISION, SOUND PROGRAMME AND OTHER MULTIMEDIA SIGNALS

Smart TV operating system

---

## The architecture of a smart TV operating system

Recommendation ITU-T J.1202

ITU-T

## Recommendation ITU-T J.1202

# The architecture of a smart TV operating system

**Summary**

Recommendation ITU-T J.1202 defines the architecture of a smart television operating system (TVOS) to enable integrated broadcast and broadband (IBB)-capable cable set-top box (STB) and TV to apply to broadcasting services and IP-based interactive services provided by cable television operators and third-party providers. By running the smart TV operating system, the IBB capable STB and TV will be able to provide subscribers with advanced and personalized services by downloading and installing advanced and personalized apps from cable operators' platforms and third-party platforms, which are interconnected with the related cable operators' platforms.

**History**

| Edition | Recommendation | Approval | Study Group | Unique ID[*] |
|---|---|---|---|---|
| 1.0 | ITU-T J.1202 | 2019-07-29 | 9 | 11.1002/1000/13975 |

**Keywords**

DCAS, DRM, Smart TV operating system, smart TVOS.

[*] To access the Recommendation, type the URL http://handle.itu.int/ in the address field of your web browser, followed by the Recommendation's unique ID. For example, http://handle.itu.int/11.1002/1000/11830-en.

## Table of Contents

# Recommendation ITU-T J.1202

# The architecture of a smart TV operating system

## 1 Scope

This Recommendation specifies the architecture of a smart television operating system (TVOS). The smart TV operating system is intended to enable integrated broadcast and broadband (IBB)-capable cable set-top box (STB) and TV to apply to broadcasting services and IP-based interactive services provided by cable television operators and third-party providers. By running the smart TV operating system, the IBB capable STB and TV will be able to intelligently provide subscribers with advanced and personalized services by downloading and installing advanced and personalized apps from cable operators' platforms and third-party platforms which are interconnected with the related cable operators' platforms.

In this Recommendation, software components, frameworks, overall software architecture and security capabilities are also required by the smart TV operating system.

The smart TV operating system is not intended to support functionalities of the head-end systems of cable television operators.

## 2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

None.

## 3 Definitions

### 3.1 Terms defined elsewhere

This Recommendation uses the following terms defined elsewhere:

**3.1.1 integrated broadcast and broadband (IBB) DTV service** [b-ITU-T J.205]: A service that simultaneously provides an integrated experience of broadcasting and interactivity relating to media content, data and applications from multiple sources, where the interactivity is sometimes associated with broadcasting programmes.

**3.1.2 television operating system (TVOS)** [b-ITU-T J.1201]: A system software running on IBB capable cable STB and TV which is capable of managing hardware, software and data resources of the IBB-capable cable STB, supporting and controlling the application software execution.

## 3.2 Terms defined in this Recommendation

This Recommendation defines the following terms:

**3.2.1 rich execution environment (REE)**: A hugely extensible and versatile operating environment which brings flexibility and capability.

**3.2.2 trusted execution environment (TEE)**: A secure area of the main processor in an IBB-capable cable STB and TV to ensure that sensitive data is stored, processed and protected in an isolated and trusted environment. It offers isolated safe execution of authorized security software providing end-to-end security by enforcement of protected execution of authenticated code, confidentiality, authenticity, privacy, system integrity and data access rights.

**3.2.3 secure OS**: An operating system running in a trusted execution environment (TEE) which is used to trigger secure execution of applications within the TEE.

**3.2.4 TVOS execution environment (runtime)**: A software module in a smart television operating system (TVOS) that evaluates and executes applications consisting of computer language instructions, associated data and media content. An execution environment is implemented right above the component layer in TVOS, and may be implemented along with computer language interpreters and/or language compilers, which an application may use to present audiovisual content, interact with a user, or execute other tasks that are not evident to the user. A common example of an execution environment is the Java software environment, using the Java programming language byte code interpreter, and a Java virtual machine for program execution.

**3.2.5 TVOS application framework**: The software module of a smart television operating system (TVOS) which consists of application programming interface units for constructing application programming interfaces with the computer programming language used by the respective applications and works along with the corresponding execution environment for the execution of respective applications.

**3.2.6 device tree source**: The device tree source is a textual representation of a device tree in a form that can be processed and compiled into a binary device tree in the form expected by the operating system. A device tree is a tree data structure with nodes that describe the physical devices in a hardware system. The tree data structure allows a runtime operating system to run on top of the respective hardware system without hard-coding hardware information of the related devices included in the hardware system.

## 4 Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

| | |
|---|---|
| API | Application Programming Interface |
| App | Application |
| AV | Audio Video |
| DCAS | Downloadable Conditional Access System |
| DRM | Digital Rights Management |
| DTS | Device Tree Source |
| DTV | Digital Television |
| ECM | Entitlement Control Message |
| EMM | Entitlement Management Message |
| EPG | Electronic Programme Guide |

| | |
|---|---|
| HAL | Hardware Abstraction Layer |
| HCI | Human-Computer Interaction |
| IBB | Integrated Broadcast and Broadband |
| IPC | Inter-Process Communication |
| OS | Operating System |
| REE | Rich Execution Environment |
| TApp | Trusted Application |
| TEE | Trusted Execution Environment |
| TVOS | Television Operating System |
| VFS | Virtual File System |

## 5 Conventions

In this Recommendation:

The phrase "**is required to**" indicates a requirement that must be strictly followed and from which no deviation is permitted if conformity with this document is to be claimed.

The phrase "**is recommended**" indicates a requirement that is recommended but which is not absolutely required. Thus this requirement needs not be present to claim conformity.

The phrase "**is prohibited from**" indicates a requirement that must be strictly followed and from which no deviation is permitted if conformity with this document is to be claimed.

The phrase "**can optionally**" indicates an optional requirement that is permissible, without implying any sense of being recommended. This term is not intended to imply that the vendor's implementation must provide the option and the feature can be optionally enabled by the network operator/service provider. Rather, it means the vendor may optionally provide the feature and still claim conformity with this Recommendation.

In this Recommendation the words *shall*, *shall not*, *should*, and *may* sometimes appear, in which case they are to be interpreted, respectively, as *is required to*, *is prohibited from*, *is recommended*, and *can optionally*. The appearance of such phrases or keywords in an appendix or in material explicitly marked as *informative* are to be interpreted as having no normative intent.

## 6 Functional architecture of the software

The television operating system (TVOS) consists of the rich execution environment (REE) and trusted execution environment (TEE).

The TVOS REE employs a hierarchical and modular software architecture and consists of five functional software layers, kernel, hardware abstraction layer (HAL), functional component, execution environment and application framework in a loose-coupling mode. Each functional software layer consists of multiple software modules in a loose-coupling mode.

The TVOS TEE consists of the secure OS, TEE HAL, and trusted application (TApp). Figure 1 shows the functional architecture of the TVOS software.

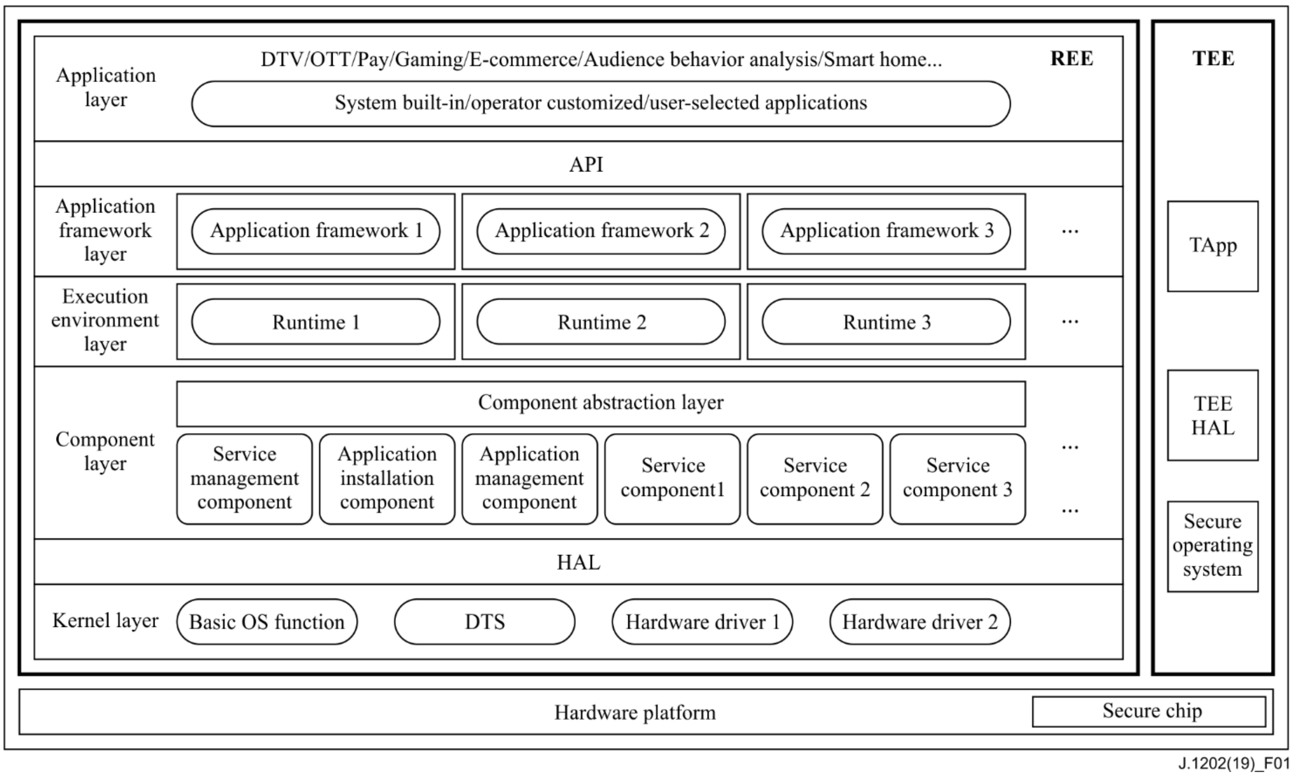


**Figure 1 – Functional architecture of the TVOS software**

The TVOS kernel layer provides basic OS functions, including abstraction, management, and allocation of system resources such as process scheduling, memory management, virtual file system (VFS), network protocol stack, inter-process communication (IPC), security policies and hardware drivers. The kernel layer provides basic OS services for the upper-layer software.

The TVOS HAL provides abstraction and encapsulation of the TVOS hardware platform capability, uses a unified abstraction and encapsulation model for hardware devices of the same type and offers unified invocation interfaces for the upper-layer software to access and control the hardware platform capability.

The TVOS functional component layer provides core functions of the TVOS and offers common service capability support for various applications. This layer contains shared functional component modules such as media processing, digital television (DTV), digital rights management (DRM), downloadable conditional access system (DCAS), secure payment, smart home, human-computer interaction (HCI), terminal control, application management and window management. Each shared functional component module is implemented in client-server mode and is independent from each other. The server and client run in different process space and use the same IPC mechanism to implement IPC. The server implements component functions and uses the HAL to invoke the kernel-layer software modules and lower-layer hardware. The shared functional component modules can be added and tailored according to system requirements and can support multiple execution environments.

The TVOS execution environment layer implements the interpretative execution environment of the application software and application adaptation software and support the loading and running multiple types of applications such as Java applications and web applications. TVOS execution environment layer supports can be added and tailored according to system requirements. Each TVOS execution environment is independent from each other and has its own application framework. For example, the execution environment includes web runtime, Java runtime or python runtime.

The TVOS application framework layer provides APIs to different types of applications. Each TVOS application framework is independent from each other and corresponds to one execution environment.

## 7 Kernel layer

The TVOS kernel layer contains the kernel, device tree source (DTS) and hardware driver modules such as Linux.

The kernel implements basic OS functions such as process scheduling, memory management, VFS, network protocol stack, I/O management, IPC, and security protection. The kernel collaborates with the secure chip to support and implement the security trust chain verification mechanism based on the hardware security trust root.

The DTS software module should decouple the kernel from hardware drivers based on the DTS mechanism.

The hardware driver software module should contain various universal hardware drivers, DTV-related hardware drivers, and security-related hardware drivers and support the TVOS kernel to communicate with, access, and manage various hardware devices.

## 8 Hardware abstraction layer

The TVOS hardware abstraction layer (HAL) consists of multiple hardware abstraction functional interface modules. These modules implement abstraction and encapsulation of different hardware capabilities, and provide the upper-layer software with interfaces used to invoke the corresponding hardware capabilities.

The hardware abstraction functional interface modules are implemented by using the Stub hardware abstraction model. The Stub hardware abstraction model takes a hardware module, hardware devices, and their operation methods as a Stub operation function. By mapping the hardware module ID to the corresponding pointer of the Stub operation function, the Stub hardware abstraction model provides the upper-layer software with the invocation method of the hardware capability, implementing operation and control of hardware capabilities.

Figure 2 shows the principle of the Stub hardware abstraction model of the TVOS HAL.

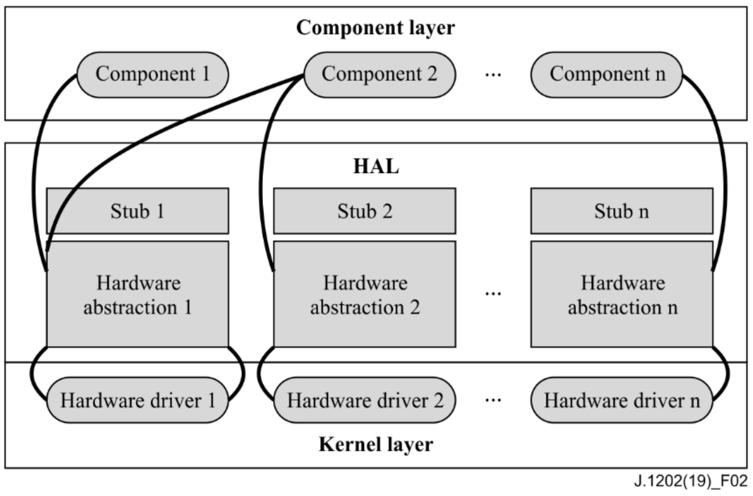


**Figure 2 – Principle of the Stub hardware abstraction model of the TVOS HAL**

The TVOS HAL functional interface modules should include the HAL functional interface modules for the hardware dedicated to media processing and those for the generic hardware.

The HAL functional interfaces for the hardware dedicated to media processing belong to hardware abstraction interfaces.

## 9 Component layer

### 9.1 Component model

The TVOS component consists of the server and client, which run in different process space and use the Binder mechanism to implement IPC. The server is responsible for implementing component functions and uses the HAL to invoke the kernel-layer software modules and lower-layer hardware. The component server is a system resident running instance and contains software modules including functional implementation of a related service and a service Stub which is an abstract implementation of the corresponding service and works as the IPC interface for exporting the service to the client. A component server running instance provides services for multiple different component client running instances. The component client contains software modules including client implementation, client APIs and service proxy. A service proxy sends the requests being executed in the remote service process to the server and gets the corresponding reply to the client. The server and client of the shared functional component modules should be implemented by using an appropriate programming language to work with various execution environments as illustrated in Figure 3.

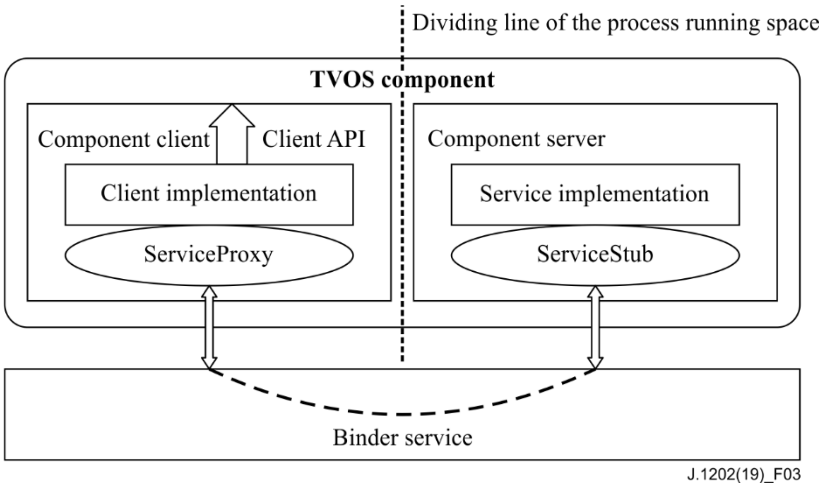


**Figure 3 – Component model diagram**

The collaboration between the component server and client requires support from the component service manager.

The component server registers the server information with the component service manager. The component client queries information about the corresponding component server from the component service manager to invoke the component server.

The component service manager parses the component names, maintains the mapping between the component names and component instances, and checks and controls the access rights of the component server. The principle of collaboration between the component and the component service manager is shown in Figure 4.

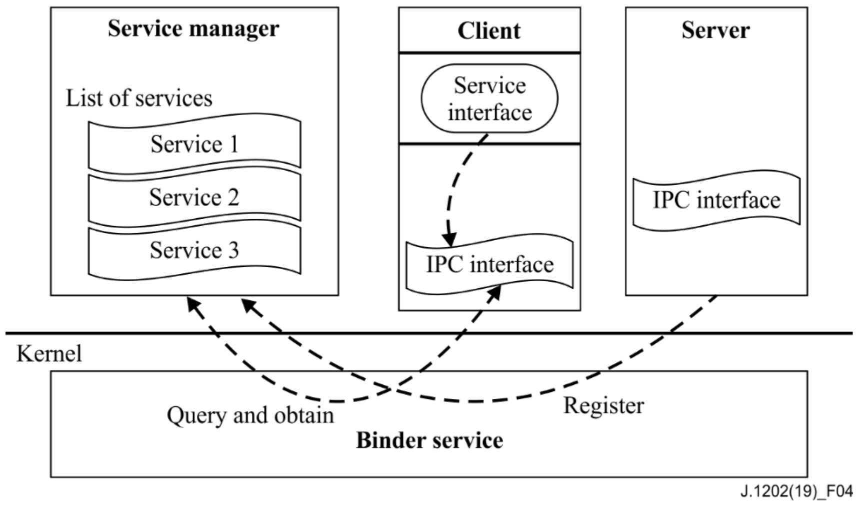


**Figure 4 – Principle of collaboration between the component and component service manager**

## 9.2 Component service manager

### 9.2.1 Functions

The component service manager should implement the following functions:

– Manage all the components in the system in a centralized manner and provide the function of registering TVOS components.

– Register a component with checking whether a component is effectively registered, including confirming whether the component is legal, whether the component is repeatedly registered, and whether the component can be allocated with sufficient resources.

– Provide other software modules and applications with the function of searching for components and obtaining component clients.

The component service manager is a special component that provides the component management function and complies with the TVOS component model.

### 9.2.2 Component implementation and invocation mode

The TVOS component service manager should be implemented according to the component model. Figure 5 shows the component implementation and invocation mode.

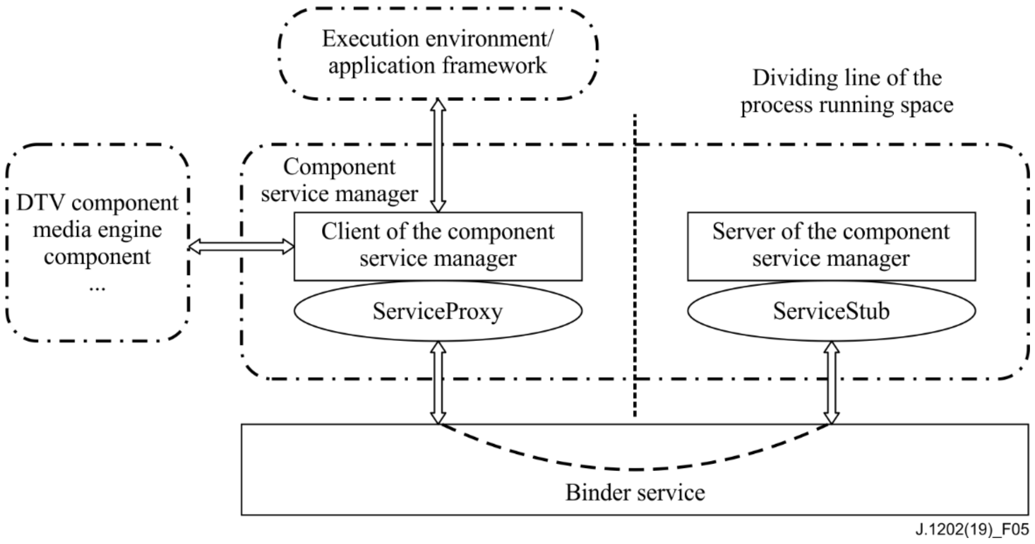

Execution environment/ application framework
Dividing line of the process running space
DTV component media engine component ...
Component service manager
Client of the component service manager
Server of the component service manager
ServiceProxy
ServiceStub
Binder service
J.1202(19)_F05

**Figure 5 – Implementation and invocation mode of the component service manager**

### 9.2.3 Functional architecture and modules

The component service manager consists of the component registration, component validity check, and component query modules, which correspond to the itemised bullets in clause 9.2.1. Figure 6 shows the architecture of the component service manager.

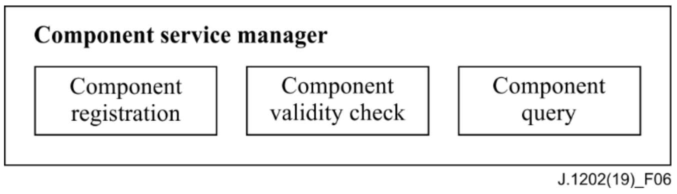


**Figure 6 – Functions architecture and modules of the component service manager**

The component registration module responds to the requests of registering other components with the component service manager.

The component validity check module checks whether the component permissions are legal, whether a component is repeatedly registered and whether resources are sufficient for implementing registration.

The component query module provides the function of querying components and obtaining component clients.

### 9.2.4 Interfaces

The component service manager should provide other software modules with component registration interfaces and provide other software modules and applications with component query and component client obtaining interfaces by using the client. The component service manager provides four functional interfaces as adding service, checking service, getting service, and listing service.

### 9.2.5 Role of component service manager

The component service manager is a special component to provide the registration and query functions for all other components and applications. The component service manager relies on the system lower-layer Binder service to manage other components.

## 9.3 Application installation component

### 9.3.1 Functions

The application installation component should implement the following functions:

– Parse TVOS application packets.

– Authenticate the security of TVOS application packets, including integrity and validity authentication and permission check.

– Install, uninstall and update TVOS applications.

– Query the installation information of TVOS applications.

– Support the application management component to manage installed applications.

### 9.3.2 Component implementation and invocation mode

The TVOS application installation component should be implemented according to the component model. Figure 7 shows the component implementation and invocation mode.

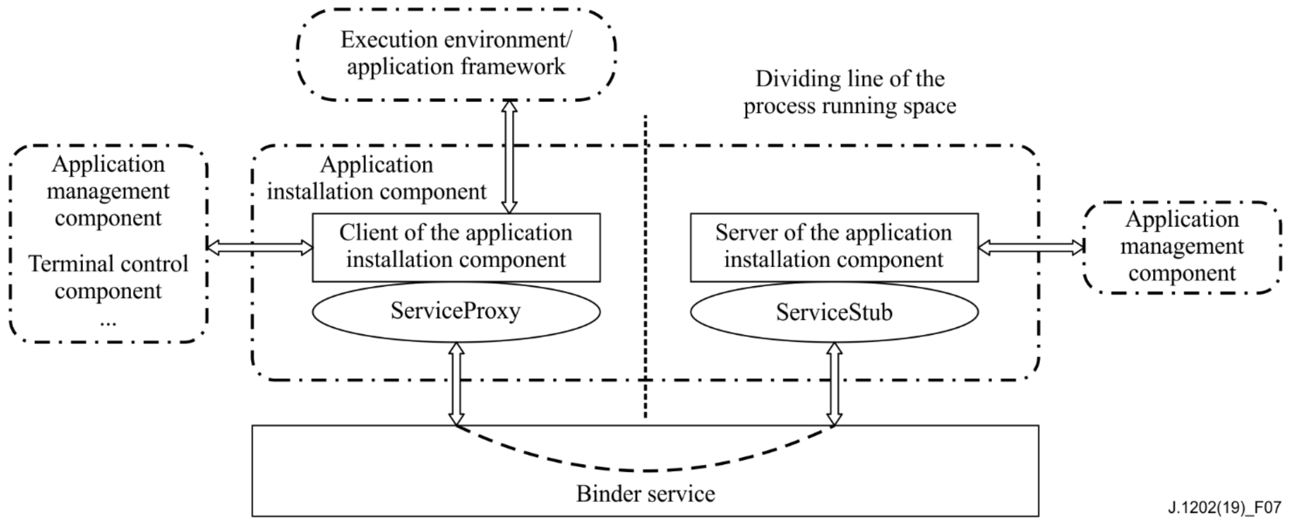

Execution environment/ application framework
Dividing line of the process running space
Application management component
Terminal control component
...
Application installation component
Client of the application installation component
ServiceProxy
Server of the application installation component
ServiceStub
Application management component
Binder service
J.1202(19)_F07

**Figure 7 – Implementation and invocation mode of the TVOS application installation component**

### 9.3.3 Functional architecture and modules

The application installation component consists of the application package parsing, application installation, application security and application query modules. Figure 8 shows the architecture of the application installation component.

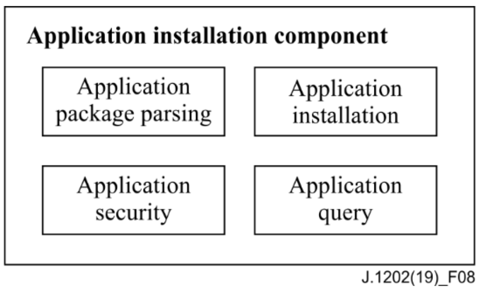


**Figure 8 – Architecture and modules of the application installation component**

The application package parsing module parses the application package to be installed and checks the installation space required for corresponding applications.

The application security module verifies the integrity and validity of the application to be installed and checks the permission requests of the application.

The application installation module installs, uninstalls and updates applications.

The application query module allows other software modules and applications to query information of the installed applications.

#### 9.3.4 Interfaces

The application installation component should provide other software modules with invocation interfaces used to install, uninstall, and update TVOS applications and interfaces used to query the installation information of the installed TVOS applications. These interfaces are functional component interfaces.

#### 9.3.5 Collaboration with other software modules

The application installation component and the application management component are interdependent. The application installation component collaborates with the application management component to implement related functions and also provides support for the application management component. The application installation component also provides the information query service for other software modules and applications.

### 9.4 Application management component

#### 9.4.1 Functions

The application management component should implement the following functions:

– The registration and deregistration of TVOS applications.

– Management of the life cycle of the running states (including start, stop, pause, resume, and exit) of TVOS applications.

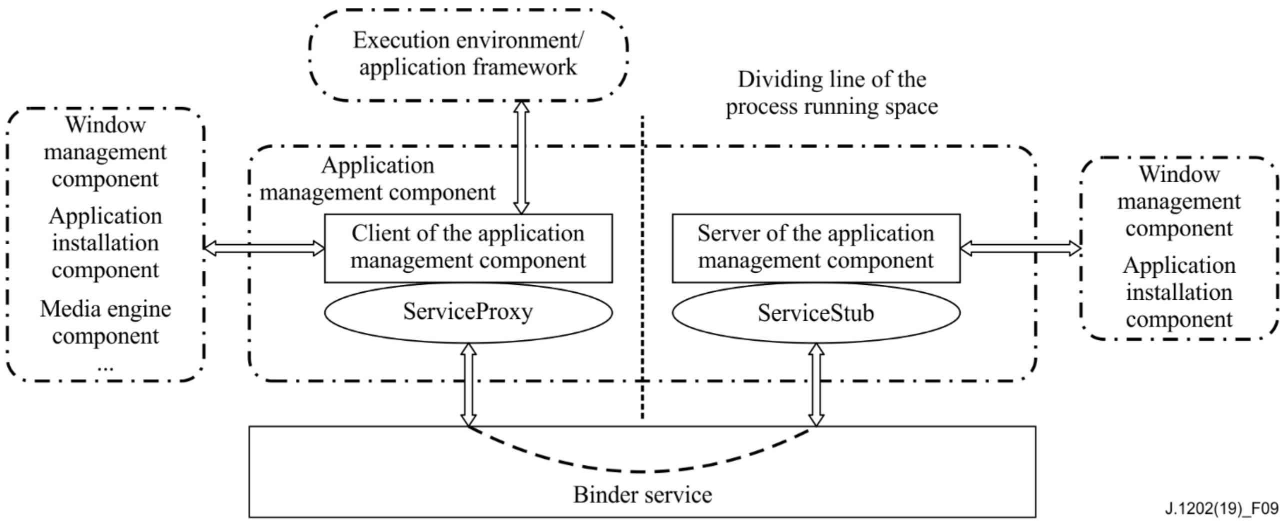
– Implementation of the communication mechanism between different software modules and applications, including the messaging mechanism for both single and multiple targets.

– Implementation of the data sharing mechanism between different software modules and applications, as well as the information on way of data sharing, and mechanism to notify data changes.

– Assistance of the application installation component for installing and updating applications.

### 9.4.2 Component implementation and invocation mode

The application management component should be implemented according to the component model. Figure 9 shows the component implementation and invocation mode.

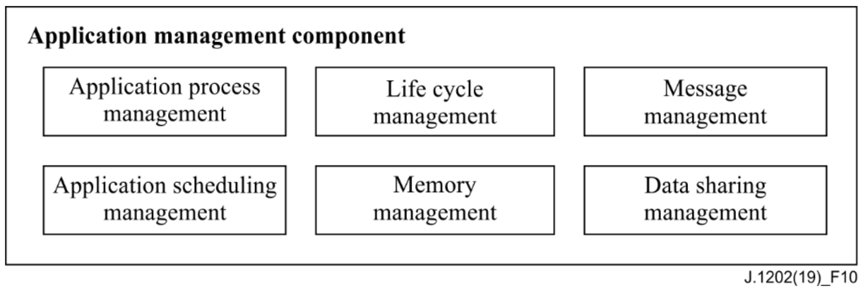


Execution environment/ application framework
Dividing line of the process running space
Window management component
Application installation component
Media engine component
...
Application management component
Client of the application management component
ServiceProxy
Server of the application management component
ServiceStub
Window management component
Application installation component
Binder service
J.1202(19)_F09

**Figure 9 – Implementation and invocation mode of the application management component**

### 9.4.3 Functional architecture and modules

The application management component contains the application memory management, process management, application scheduling management, application life cycle management, messaging management and data sharing management modules. Figure 10 shows the functional architecture and modules of the application management component.



Application management component
Application process management
Life cycle management
Message management
Application scheduling management
Memory management
Data sharing management
J.1202(19)_F10

**Figure 10 – Functional architecture and modules of the application management component**

The application process management module manages the registration and deregistration of the application, allocates a unique identifier for the application and allocates resources for the application.

The application memory management module manages the allocation of the memory for the application.

The application life cycle management module manages the application running states such as start, running, suspend, resume, and stop.

The application scheduling management module schedules and manages the state of the applications considered including activation, suspend, and switching the execution between foreground and background.

The messaging management module provides the messaging mechanism for both single and multiple targets for communication between different software modules and applications.

The data sharing management module implements the data sharing mechanism to support and manage data sharing between different software modules and applications.

#### 9.4.4 Interfaces

The application management component should provide interfaces used to control start and stop of applications as well as interfaces used to query information about running applications through the client.

#### 9.4.5 Collaboration with other software modules

Installation, uninstallation and execution of the applications are achieved by collaboration of the application management component, application installation component, and window management component. The application management component and application installation component are key components for application installation and uninstallation. The application management component and window management component are key components for application execution. For that purpose, the application management component provides support for the implementation and running of other components and applications.

### 9.5 Service components

#### 9.5.1 Functions

TVOS may implement various services by different service component. TVOS should support following components:

– DTV component for DTV related service;

– Media engine component for media service;

– DCAS component for DCAS service;

– DRM component for DRM service;

– HCI component for human-computer interface service.

#### 9.5.2 Component implementation and invocation mode

TVOS service components should be implemented according to the component model. Figure 11 shows the component implementation and invocation mode.

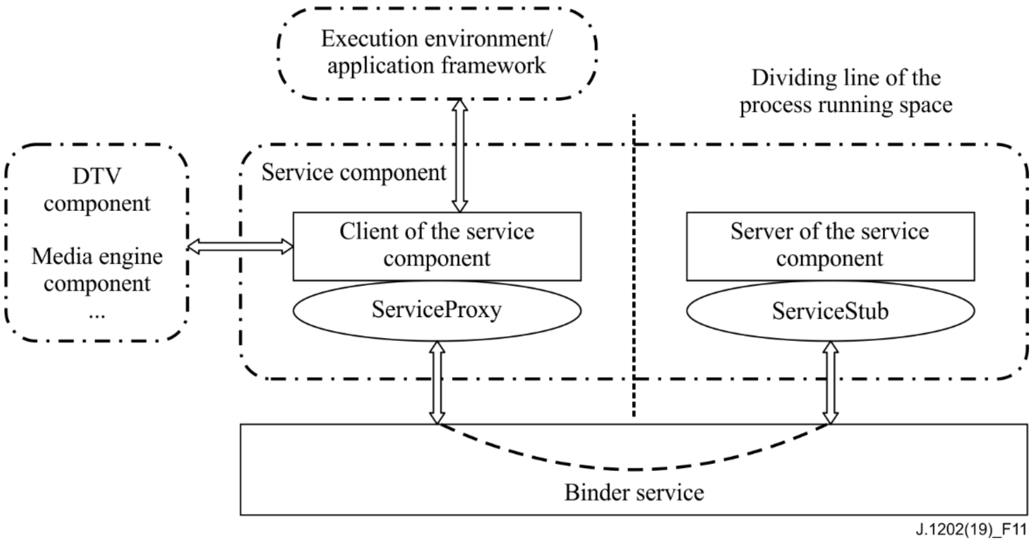


**Figure 11 – Implementation and invocation mode of the service component**

## 10 Application execution environment

### 10.1 Functions

The TVOS execution environments should implement the following functions:

– The interpretation and running environment for TVOS applications and the application framework-layer functional interface instances invoked by TVOS applications.

– Loading and running of TVOS applications.

– Process isolation for TVOS applications by using of different Runtime instance for each TVOS application.

– Cooperation with the application management component to manage the life cycle of TVOS applications, including starting, pausing, resuming, and restarting TVOS applications.

– Managing the access rights of TVOS application resources, including checking and requesting the rights of TVOS applications.

### 10.2 Architecture and implementation mechanism

The TVOS execution environments create and provide the running environment for TVOS applications and manage the running, rights and security of TVOS applications.

Figure 12 shows the architecture and implementation of TVOS execution environment.

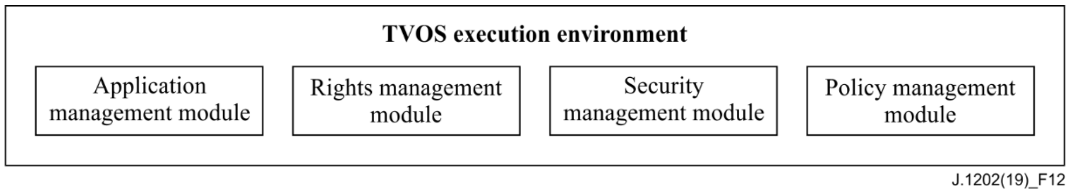


**Figure 12 – Architecture and implementation of TVOS execution environment**

TVOS execution environment consist of the application running management module, rights management module and security management module.

The application management module creates instances of TVOS execution environment, forms the basic environment of TVOS applications, loads applications to the basic environment, starts and manages the life cycle of applications.

The rights management module manages the access rights of TVOS application resources when TVOS applications are running.

The security management module implements security management including process isolation and data isolation.

The policy management module implements policy management for the running modes of TVOS applications, including the policy to exclusively occupying processes for each application or sharing the same process for multiple applications.

## 11 Application framework

### 11.1 Architecture of the TVOS application framework

The TVOS application framework consists of the TVOS application programming interface units and corresponding interpretative execution environment functional interface units.

### 11.2 TVOS application programming interface units

#### 11.2.1 Functions

The TVOS application programming interface units implement interfaces of various functional components and modules, provide application programming interfaces, and assist applications in implementing DTV services such as electronic programme guide (EPG), channel list and TV programme playing.

The TVOS application programming interface units include:

– Unidirectional broadcast network access unit;

– Broadcast protocol processing unit;

– Bidirectional broadband access unit;

– HCI unit;

– AV setting unit;

– Media processing unit;

– Message management unit;

– Broadcast information service unit.

#### 11.2.2 Main functional interface units

Main functional interface units are as follows:

a) Unidirectional broadcast functional interface unit:

This unit cooperates with the DTV component to access the unidirectional cable DTV broadcast network, including controlling the frequency, modulation mode, and symbol rate as well as obtaining information such as the signal strength and quality. This unit also cooperates with the DTV component to implement basic DTV functions such as programme search, programme guide, and information search, supporting the running of DTV-related applications.

b) Bidirectional broadband network functional interface unit:

   This unit cooperates with the network service to implement bidirectional broadband network access, including connection management.

c) HCI functional interface unit:

   This unit cooperates with the HCI component to implement the user interface. The user instructions can be obtained by input devices such as the remote control, mouse, keyboard, and front-panel key as a form of key messages. The information to the user is shown to the front panel or display screen.

d) Configuration functional interface unit:

   This unit cooperates with the AV configuration component to some applications to configure audio and video output parameters, including global volume, and volume status of the audio output and brightness, contrast, saturation of the video output.

e) Media playing functional interface unit:

   This unit cooperates with the media component to implement media playing functions, including playing, control, language selection, event processing, and exception processing.

f) Message management functional interface unit:

   This unit is the TVOS message repository for the applications dealing with message event, message event listener, and message manager. This unit manages and distributes messages and assists applications in message acquisition.

g) DCAS functional interface unit:

   This unit cooperates with the DCAS component to implement DCAS application management, ECM/EMM data processing, and DCAS data interaction, assisting DTV applications in playing encrypted DCAS programmes. Some interfaces may have limitation for access by the application.

h) Terminal control data collection unit:

   This unit cooperates with the terminal control component and other components for some applications to interact with the terminal control and data collection components.

i) Broadcast message service unit:

   This unit provides the monitoring, reception, and processing functional interfaces related to broadcast information services.

### 11.2.3 Relationship of the interfaces and functional components

The framework-layer TVOS application programming interface depends on function support from components at the component layer. Figure 13 shows the interface relationship.

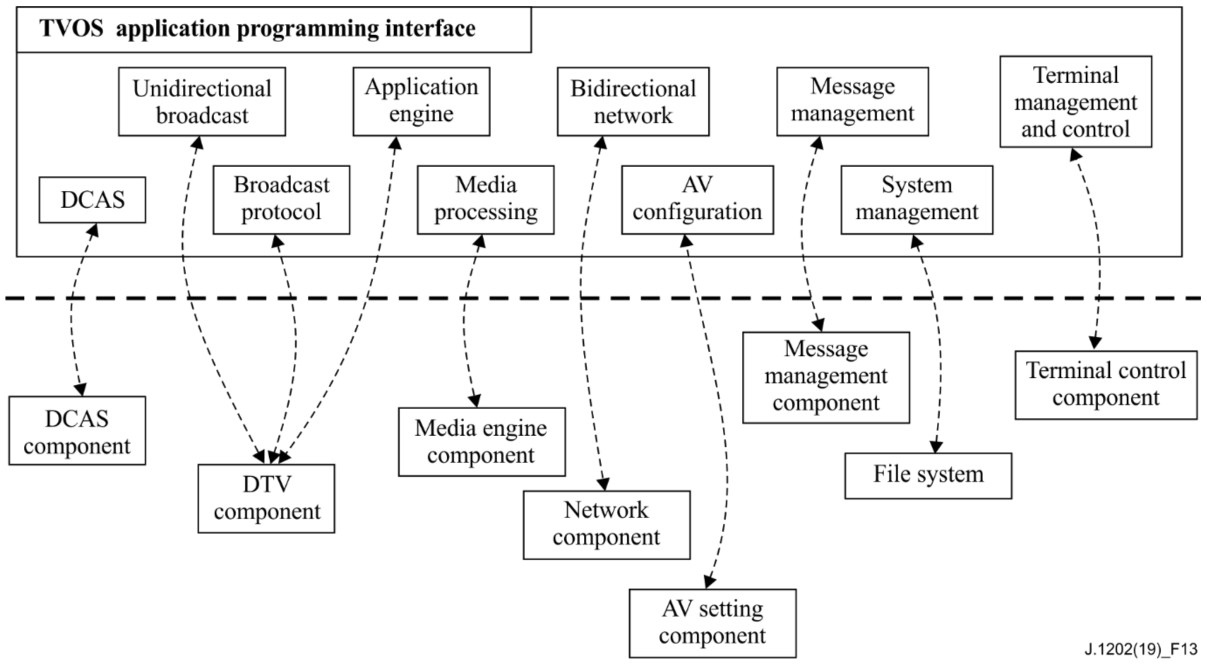


**Figure 13 – TVOS application programming interface of the framework layer**

## 11.3 Interpretative execution environment functional interface units

The interpretative execution environment functional interface units support the standard application programming interfaces of each interpretative programming language.

## Bibliography

| | |
|---|---|
| [b-ITU-T J.205] | Recommendation ITU-T J.205 (2012), *Requirements for an application control framework using integrated broadcast and broadband digital television*. |
| [b-ITU-T J.1201] | Recommendation ITU-T J.1201 (2019), *Functional requirements of a smart TV operating system*. |
| [b-ETSI EN 301 192] | ETSI EN 301 192 V1.5.1 (2009), *Digital Video Broadcasting (DVB), DVB specification for data broadcasting*. |
| [b-ETSI TS 102 809] | ETSI TS 102 809 V1.3.1 (2017), *Digital Video Broadcasting (DVB), Signaling and carriage of interactive applications and services in Hybrid broadcast/broadband environments.* |
| [b-ETSI TS 102 851] | ETSI TS 102 851 V1.2.1 (2011), *Digital Video Broadcasting (DVB), Uniform Resource Identifiers (URI) for DVB Systems*. |

Printed in Switzerland
Geneva, 2019