



# **Service Interface Specification rev 1.0**

## **The Basebox Environment**

**April 2007**

**Acreo**



# 1 Background

To enable Open IPTV where the service providers for TV based services can reach the endusers on non discriminating conditions a, well defined interface between the services and the TV platform is necessary. An important part of the Basebox environment as being the tool to establish Open IPTV is therefore to create and standardise the Service Interface

## 1.1 Scope

This document describes the Service Interface, i.e. the API interface between the Basebox environment middleware and the supported services. It describes types of services to be integrated as well as defines the guidelines to ensure system performance, stability and security.

This document will during the 3:rd quarter 2007 be followed by a document where the very API will be published in detail.

## 1.2 Services

One of the initial requirements defined by the Basebox environment is that services from different providers need to be easy to integrate. Every service shall only need to be integrated once.

Examples of services are:

1. TV
2. VoD (Video on Demand)
3. Games
4. Audio books
5. Educational services
6. Advanced EPGs (Electronic Program Guides)
7. e-Stores
8. Images, for example pause-TV and photo albums
9. Internet radio
10. News



## 2 Definitions

### 2.1 Middleware

Wikipedia<sup>1</sup> defines *middleware* as “...a piece of software that connects two or more software applications so that they can exchange data.” In the Basebox environment, the middleware's role is to provide the glue between services, set-top boxes, CA (Conditional Access), billing systems, provisioning, and other third party applications.

### 2.2 Services

A service is a bundle of content that end users have access to.

#### Integrated Services

Integrated services are services developed following the Basebox environment guidelines according to this document. This means they will function on all Basebox verified set-top boxes and meet the performance and stability requirements issued by the Basebox environment in their Functional Requirements.

#### Non-Integrated Services

Non-integrated services as the non-integrated plug-ins described below can be run in the Basebox environment but do not necessarily function on all supported set-top boxes and do not automatically meet the performance and stability requirements of the environment

#### 2.2.1 Service Components

A service contains one or more components. One service, for instance, may contain a VoD component with movies, a TV channel package and a customised EPG implemented as a plug-in.

---

<sup>1</sup> See <http://en.wikipedia.org/wiki/Middleware>



## **2.3 Content**

Content is the value of a service, the contained items. In a VoD service, for example, the movies are the content. In a game portal, the games are the content.

Content can be presented to the end user through the middleware content engine or through a third party developed plug-in.

### **2.3.1 Native Content**

Native content is content the middleware can manage itself through the content engine without a third party application. Many services can be distributed using native content only. This means the service provider does not need to worry about building their own application.

Services distributed through the middleware content engine are always considered integrated as the middleware guarantees they can be run on all supported set-top boxes and that they meet other system requirements.

### **2.3.2 Plug-In**

A plug-in is a third party application run from the middleware. Example of plug-ins are games, portals and external EPGs. A plug-in can either be integrated or non-integrated

#### **Integrated Plug-In**

Plug-ins developed following the base-box guidelines including using the supported HAL, i.e. functions on all verified set-top boxes are considered integrated, as defined above.

#### **Non-Integrated Plug-In**

Other plug-ins can also be run from the middleware but do not necessarily meet the Basebox environments requirements as defined. They might not function on all Basebox verified set-top boxes and performance, stability and security may be an issue. These plug-ins are considered non-integrated



## 3 Functionality of the Service API

### 3.1 General Requirements

Initial requirements outlined by the Basebox environment that needs to be considered in the middleware and the service API.

<i>ID</i>	<i>Item</i>	<i>Description</i>
	Set-top box independence	Services need to be set-top box independent. When the service has been added it should work on all supported set-top boxes. This is not applicable for non-integrated services/plugin-ins.
	Service provider access	Service providers need to have direct access to the middleware through a well-defined service API, allowing them to manage their services and content.
	Types of services	The middleware needs to support both native content and plug-ins as defined above.
	Extendable	The middleware needs to be extendable to include service types and functionality added to the base-box environment in the future. Future extensions should not affect existing third party suppliers and services.
	Web services	The service API should be implemented using well proven standardised technologies familiar to the majority of developers, Therefore web services has been chosen for the moment as the implementation strategy, using standard technologies such as HTTP, XML and SOAP.



## 3.2 Content Engine

The middleware shall provide a content engine used by service providers to present their content without having to implement their own application.

<i>ID</i>	<i>Item</i>	<i>Description</i>
	Native data	The content engine needs to support the following native data types <ul style="list-style-type: none"> <li>■ Streaming video (e.g. TV channels, movies)</li> <li>■ Streaming audio (e.g. music, speech books)</li> <li>■ Images (e.g. photos)</li> </ul>
	Customised properties	Service properties can be customised on a per component basis. The service provider defines what properties should be used for a specific component.
	Branding	Services can be branded to match the providers' graphical profile. This includes changing <ul style="list-style-type: none"> <li>■ Screen background</li> <li>■ Colors</li> <li>■ Graphics</li> </ul>
	Screen Areas	The service view is divided into customisable areas. Configurable properties include <ul style="list-style-type: none"> <li>■ Size</li> <li>■ Position</li> <li>■ Update Conditions</li> </ul> There are different types of areas. Initially, the following are defined (subject to additions) <ul style="list-style-type: none"> <li>■ Static areas</li> <li>■ Preview areas</li> <li>■ Popup dialogs</li> <li>■ Item views</li> </ul>
	Startup Times	Start times for services using the content engine needs to be very short (a maximum of 2-3 seconds is desirable).



### 3.3 Service Properties

A service has properties that define how the service is displayed, how end users subscribe to it and under what terms and conditions etc.

<i>ID</i>	<i>Item</i>	<i>Description</i>
	Authentication	End users can be authenticated in different ways (subject to additions) <ul style="list-style-type: none"> <li>■ PIN code</li> <li>■ MAC address</li> <li>■ IP address</li> </ul>
	Subscription conditions	Subscription conditions are configured on a per service basis, from a set of predefined options. Additional options may be added in the future, and the middleware needs to implement these once supported by the base-box environment.
	Billing	Services can be configured in different ways to handle billing. Some examples are <ul style="list-style-type: none"> <li>■ Operator managed invoice</li> <li>■ Content provider managed invoice</li> <li>■ Credit card</li> <li>■ Internet payment, e.g. paypal</li> <li>■ SMS payment, using a cellular phone</li> </ul>
	Price models	The following price models should be supported by the middleware <ul style="list-style-type: none"> <li>■ <i>Free</i> – the service is available free of charge</li> <li>■ <i>Plain</i> – the end user pays the same amount for each item purchased</li> <li>■ <i>Ladder</i> – the end user pays less after a specified number of items purchased</li> <li>■ <i>Subscription</i> – the end user pays one amount to use the service for a specified time</li> <li>■ <i>Time</i> – the end user pays only for the time using the service, for instance a specified amount per hour</li> </ul>
	Component wizards	The service API itself should be generic, and make as little assumptions as possible about the services. However, predefined wizards for common service components need to be available to facilitate service integration. These include (subject to additions): <ul style="list-style-type: none"> <li>■ TV Channel Packages</li> <li>■ Video on Demand</li> <li>■ Plug-Ins</li> </ul> The wizards create service components with a predefined set of properties.



### 3.4 Branding and Skins

Service providers need to be able to apply their own branding and profiling.

<i>ID</i>	<i>Item</i>	<i>Description</i>
	Background	The background of a service component can be customised.
	Item list view	Content items can be customized. At least, the following aspects of an item can be customised (subject to additions): <ul style="list-style-type: none"> <li>■ Size</li> <li>■ Textual content</li> <li>■ Images</li> <li>■ Cursor</li> </ul>
	Information view	When an item is selected for additional information, a dialog or equivalent is displayed with detailed information. At least, the following can be customised in the information view: <ul style="list-style-type: none"> <li>■ Size</li> <li>■ Background</li> <li>■ Textual content</li> <li>■ Images</li> </ul>

### 3.5 Plug-Ins

Running plug-ins from within the middleware should be as easy as possible, and one service or service provider should not affect other services or other parts of the middleware.

<i>ID</i>	<i>Item</i>	<i>Description</i>
	Starting plug-ins	Plug-ins need to be started from within the middleware.
	Startup times	Start times for plug-ins need to be as short as possible (a maximum of 2-3 seconds is desirable). Note! No guarantee for non integrated plug-ins.
	Exiting plug-ins	There needs to be a coherent way to exit services and return to the middleware. This can be achieved by, for example, locking the functionality of one or more keys on the remote.
	Error Management	If an error occurs when trying to start a plug-in. The end user needs to be notified and taken back to the portal.

### 3.6 Integration with External Systems

One purpose of the middleware is to enable all actors, for instance network operators and service providers, to work towards one coherent interface. This means the middleware needs to be able to communicate and exchange information with other third party systems.



<i>ID</i>	<i>Item</i>	<i>Description</i>
	CA	When content is distributed through the middleware service engine, the CA system needs to be informed of the content added, and appropriate actions taken. CA accounts should be created when customers are added to the middleware.
	Billing	Information about transactions need to be communicated with the billing system. Plug-ins communicate transactions using the service API.
	Provisioning	Information about available services need to be communicated with third party provisioning systems. End users need to be able to choose distributors (for example ISPs and TV channel distributors) themselves.

### **3.7 Logging and Auditing**

Logging of events are of uttermost importance when a lot of different actors interact with the middleware. The information described here is accessible using the service API.

<i>ID</i>	<i>Item</i>	<i>Description</i>
	Auditing	Service usage and statistics need to be accessible by both the network operator and service providers.
	Logging	Detailed logging to help detect errors and security issues need to be available to both the network operator and service providers. Logging information include <ul style="list-style-type: none"> <li>■ Severity level               <ul style="list-style-type: none"> <li>■ Debug</li> <li>■ Information</li> <li>■ Warning</li> <li>■ Error</li> <li>■ Security</li> </ul> </li> <li>■ Timestamp</li> <li>■ Message</li> </ul>
	Transactions	Purchases made by end users are tracked as transactions with the following information <ul style="list-style-type: none"> <li>■ Sequential number</li> <li>■ Timestamp</li> <li>■ Customer reference</li> <li>■ Purchase information               <ul style="list-style-type: none"> <li>■ Price</li> <li>■ Terms</li> </ul> </li> </ul>