secure ▫ enable ▫ interact

Unity

Unity Flash EPG

Architecture

14 March 2011
SGO-DSN-1103 0.03

Confidential
NDS Employees Only

secure ▫ enable ▫ interact

NDS

Total pages:	30

Doc. Title:	Unity Unity Flash EPG Architecture

Doc. No.:	SGO-DSN-1103	**Classification:**	Confidential
Revision:	0.03	**Restriction:**	NDS Employees Only
Date:	14 March 2011	**Customer:**	Product

Owner:	Suresh Gopathy
Author:	Suresh Gopathy

Reviewers/ Approvers:	Andre Surcouf Arunkumar Kantharaj David Dinsdale Julien Marichez Madhusudhan Peta Mahesh Agarwal Muhammad Khan Prashanth Nanjundappa Steve Taylor Stewart Towler Sushma Rai Swetha VG Vladimir(NDS-India) Ruppo

Contents

List of Tables

List of Figures

1 Preface

1.1 Abstract

This document defines the STB architecture for Flash EPG in high level and defines the Unity Flash EPG component architecture in detail.

The document focuses to provide technical information and does not discuss commercials.

1.2 Applicability

This document is applicable for NDS Platforms running Flash Player supporting

- Flash Lite 4 specification
- Action Script 3.0

1.3 Terminology

Table 1 provides a short glossary of any terms crucial to the understanding of this document, and lists the acronyms and abbreviations used in the document.

Table 1 Terminology

Term	Definition
AFL	Adobe Flash Lite Player
AS	Action Script
Aux App	Auxiliary Application
AVM	Action Script Virtual Machine
CA	Conditional Access
CS	Creative Suite
DFAS	Darwin Fusion API Shim
GUI	Graphical User Interface
FLA	Flash Application Project File
FUAPI	Fusion User API
MVC	Model View Controller
MW	Middleware
SWF	Shock Wave Flash Movie

2 Introduction

Unity Flash EPG is built on Flash Player which supports Action Script 3 and compatible with Flash Lite 4 Player specification.

Unity Flash EPG uses the Flash Player API for building UI and Animation and FUAPI for domain functionalities.

First development of Flash EPG was done on Flash Player 6 using Action Script 2 targeting a specific customer with non Snowflake UI – the SKY-D Flash EPG.

Unity Flash EPG project's current objective is to develop an EPG to support different customers with various flavors of NDS Snowflake UI via configuration & customization. It can be extended in future to support various other UI flavors.

3 STB Software Architecture

Unity Flash EPG is built on Adobe Flash Player which is a component in the Platform Services Layer in the Fusion Middleware.

The Flash Player hosts the Action Script Virtual Machine (AVM) and exposes APIs for building UI and Animation. It provides extension support through which NDS exposes domain API for building EPG. NDS APIs are called as FUAPI – Fusion User API.

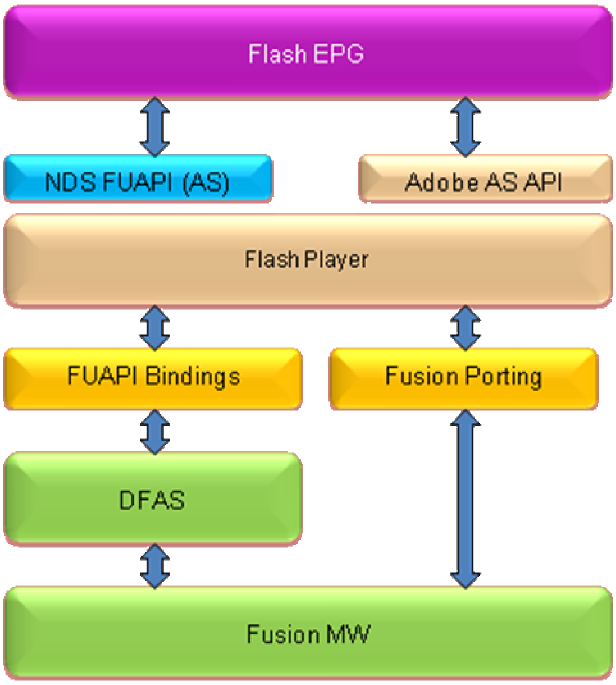

Figure 1 STB Architecture of Unity Flash EPG

Table 2 STB Components

Component	Responsibilities	Collaborates with …
Flash Player	Flash Runtime (VM) to execute AS Code and Play SWF movies.	Fusion Porting
Fusion Port & FUAPI Bindings	Manages Porting of Flash Player on Fusion. Manages FUAPI Bind between AFL and DFAS.	Flash Player, DFAS
Fusion User API	Defines the domain API that will be used by the Flash EPG	FUAPI Bind, DFAS
DFAS	Adaptation Layer of FUAPI to DMS.	DMS (Fusion MW)
Flash EPG	Handling presentation of UI, UI navigation and behavior (states)	Flash Player via Action Script and NDS FUAPI.

4 SKY-D Flash EPG

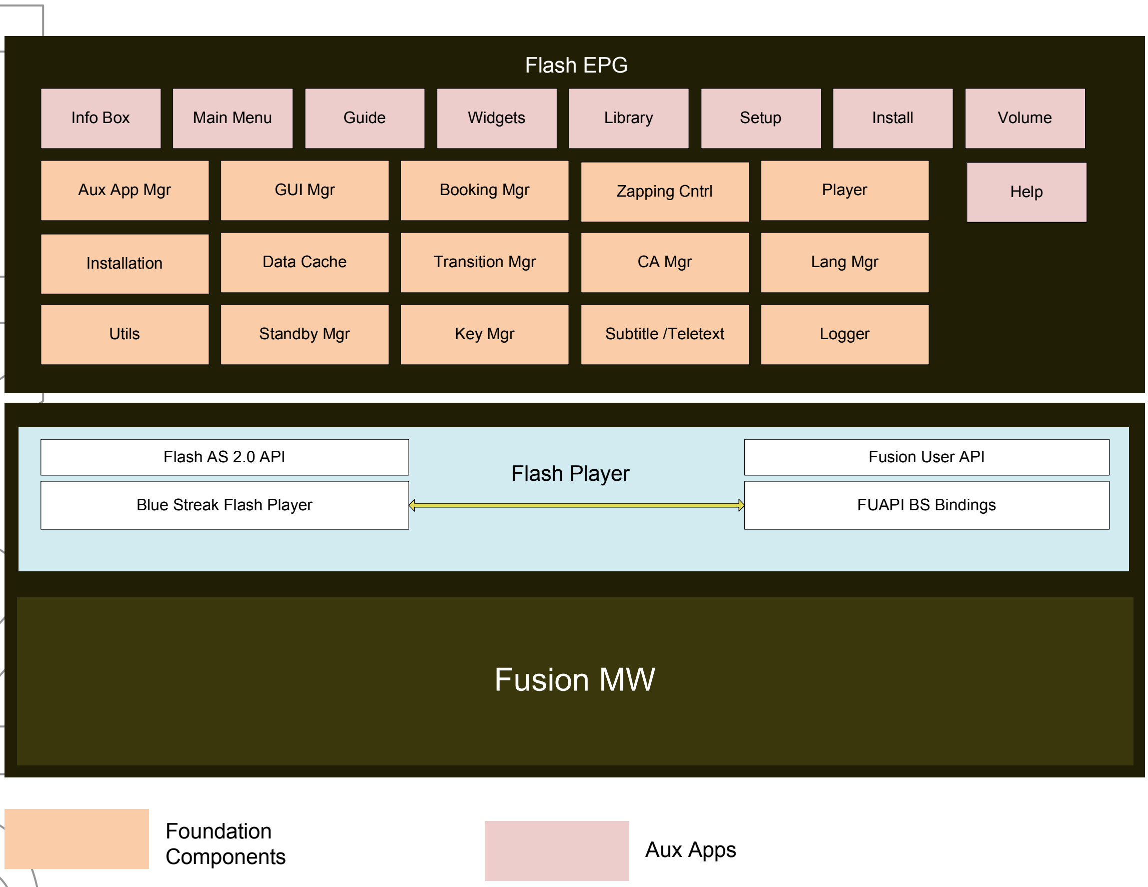

Figure 2: SKY-D Flash EPG STB Architecture

This SKY-D Flash EPG had the relevant fundamental components which formed the basis of the Unity Flash EPG.

5 Unity Flash EPG - Design Strategy

5.1 Objective

- To provide a scalable framework for developing various flavors of EPGs.
- To reduce time to market by creating reusable, modifiable and configurable components.

5.2 Key Definitions

Aux App is composed of one or more screens that are related. EPG is composed of multiple Aux Apps which can be resident or downloaded at runtime.

Screen represents the user interface layout and contains Domain & Basic Widgets and handles navigation among them.

Widgets are classified into basic and domain widgets. Basic Widgets are fundamental elements of user interface. Domain Widgets are fully implemented user interface of the domain data.

5.3 Key Design Decisions

The following are key design decisions made for Unity Flash EPG:

- **Multi layered**
  - Unity CORE Foundation
    - Foundation provides frameworks and domain services to build any flavor of Flash EPG.
    - Aux App Model is realized as domain service.
  - Unity CORE Aux Apps
    - This layer contains
      - Controller and View classes of Aux Apps
      - State Management
  - Unity EPG UI
    - This is the Presentation Layer specific to a customer.

- **Widget Driven Development**
  - Widgets are created to support various look and feel of Flash EPG. These widgets are categorized based on the look and feel which forms the Widget Catalog.

- **MVC**
  - Screens & Widgets follow MVC with Presentation separated from view.

- **Aux App Architecture**
  - EPG constitutes of multiple Auxiliary Application which can be plug & play on demand instead of one monolithic application.

- **Resource Separation**
  - The resources used in the EPG UI such as Images, Animation, Strings, Fonts, etc are specific to a customer and hence decoupled from the Unity EPG UI.
  - Helps to achieve static and dynamic skins for EPG.

- **3D – 2D**
  - Animation toolkits to support 3D and 2D animations with common code base.

- **Tailorable EPG**
  - Customization of functionality, navigation and UI variations are addressed via
    - Feature Configuration
    - GUI Configuration
    - Feature Profiling
    - Navigation Configuration

- **Across NDS Middleware**
  - Unity Flash EPG can run on any NDS middleware supporting NDS Fusion User API.

5.4 Multilayered Architecture

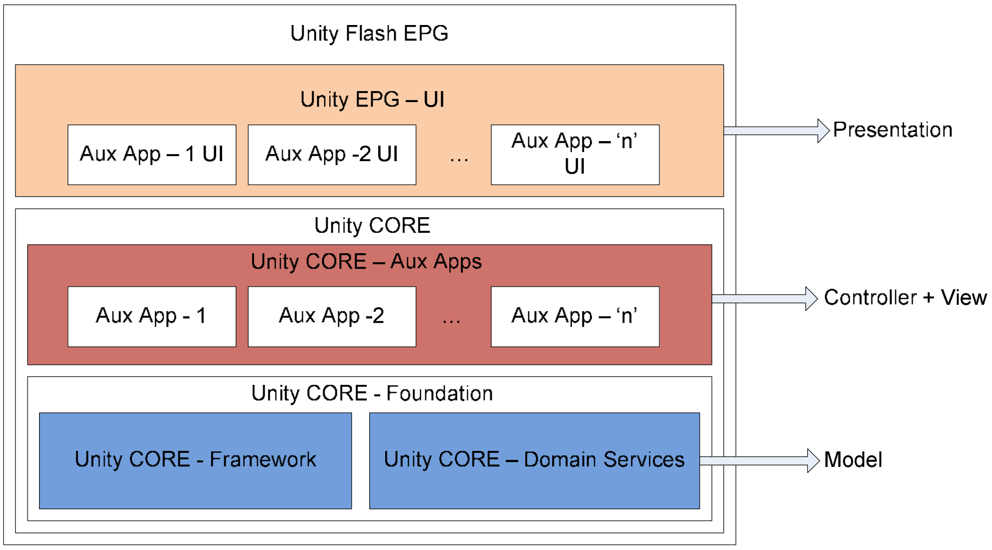

Figure 3 Multilayered Architecture

Unity CORE layer is organized as Unity Core Foundation and Unity Core Aux App.

Unity CORE Foundation layer is organized as Framework and Domain Services which are reused across all EPG applications developed on Flash technology.

Unity CORE Framework provides the fundamental building blocks to build any EPG Applications e.g., Aux App Mgmt, GUI Mgmt, State Transitions, etc.

Unity CORE Domain Service provides domain related services that are required to build any EPG application e.g., Booking, Player, Tuner, etc.

Unity CORE Aux Apps layer has featured based apps designed to support various flavor of UI Architecture and navigation paradigm e.g., Infobox, Main Menu, Guide, Settings, Library, Help, etc. Views are defined in this layer.

Unity EPG UI layer provides the UI (Screen Layout, resources, animation, etc) to the various Aux Apps in the Unity CORE Aux App layer. This decouples the UI from behavior.

5.5 Widgets

Unity Flash EPG will provide a

- Framework to build customizable widgets
- Catalog of widgets

Widgets will be plugged into the screens as per the demands of the customer or based on look of the EPG and Widgets will follow the MVC-Presentation model.

The widgets are classified into two categories:

- Basic Widgets
  - Are fundamental building blocks used in a screen or domain widget e.g., List, Table, etc.
- Domain Widgets
  - Are made up of one or more basic widgets to provide a specific UI representation of a business data e.g. Guide, Channel Bar, Search Result, etc.
  - Have the domain model associated with them.

> **Note** Both the Basic and Domain Widgets can be configured and customized. By this way, we can decouple the customer specific / Snowflake flavor specific views from the screens and navigation.

5.6 MVC Pattern

The Unity EPG uses the MVC pattern for Screens and Widgets.

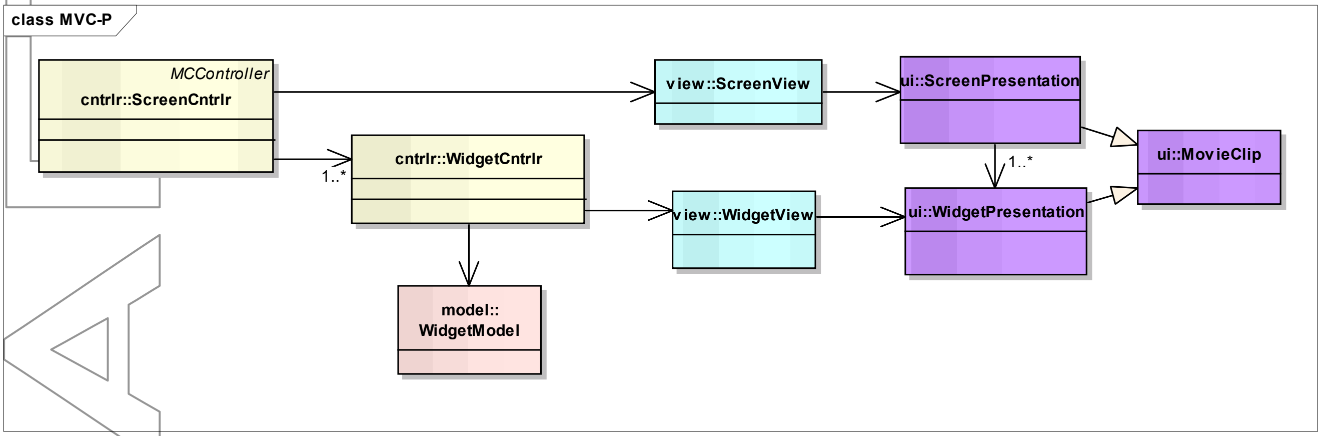

Figure 4 MVC - Screens & Widgets

Screen:

Screen follows the MVC pattern and is a container of widgets

- All Screen Controllers are derived from the base class “ScreenCntrlr” which provides the necessary interfaces for the states to communicate.
- Screen Views are interfacing classes with navigation logic and loosely coupled with presentation.
- Screen Presentations are the classes that are attached to the Movie Clip defined in .FLA for presenting the view of the screen. Layout, animation, colors are defined here.

Widget:

Widget follows the MVC structure internally.

- All Widget Controllers are derived from the base class “WidgetCntrlr” which provides the necessary interfaces for the screens to communicate with the widgets.
- Widget Model - Each widget has its own model. Basic widgets provide the base model class which can be derived for specific model implementation to support domain widgets. E.g. ChannelListModel to support ChannelListWidget.
- Widget Views are interfacing classes with navigation logic and loosely coupled with presentation.

- Widget Presentations are the classes that are attached to the Movie Clip defined in .FLA for presenting the view of the widget. Layout, animation, fonts, colors are defined here.

The realization of MVC-P in widgets is depicted below:

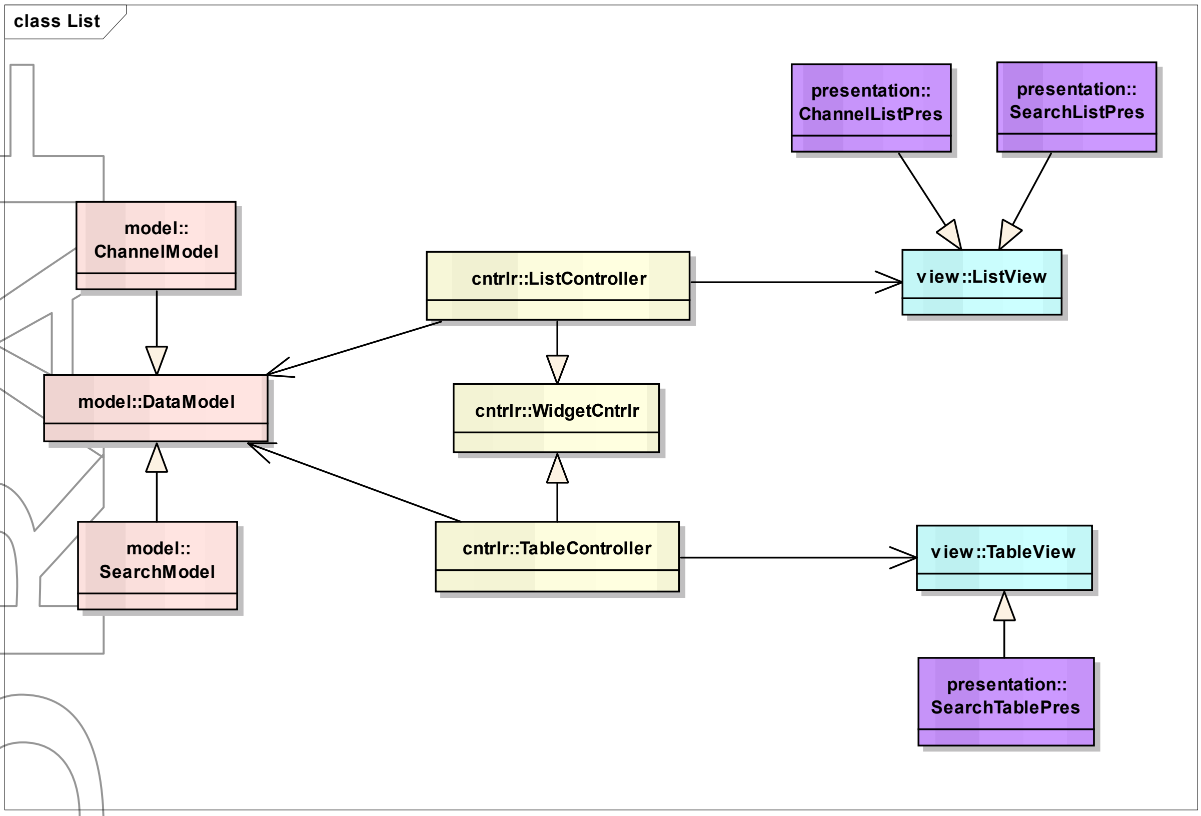

Figure 5 Widget MVC-P Realization

5.7 Aux App Architecture

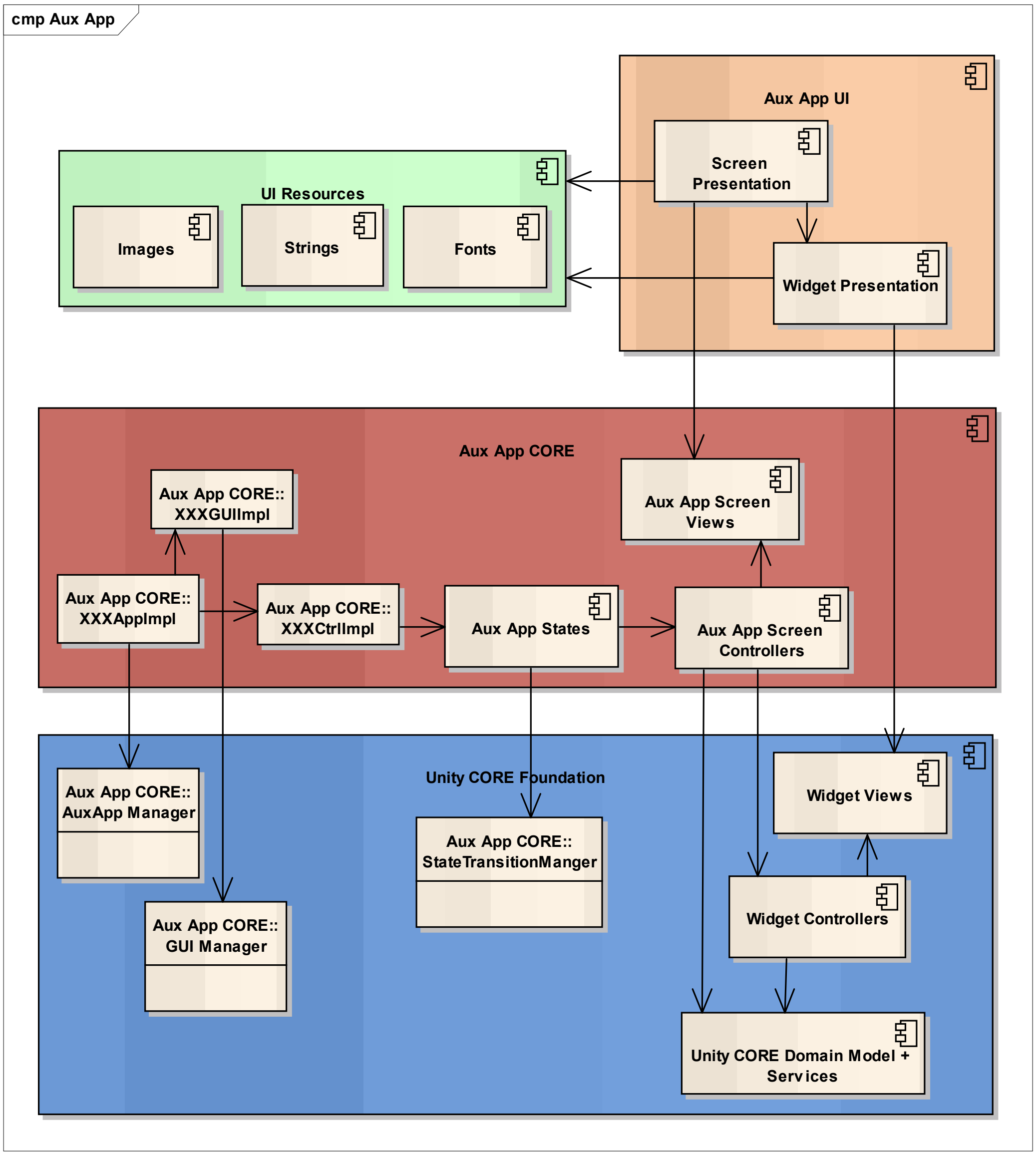
cmp Aux App
Aux App UI
Screen Presentation
Widget Presentation
UI Resources
Images
Strings
Fonts
Aux App CORE
Aux App CORE:: XXXGUIImpl
Aux App CORE:: XXXAppImpl
Aux App CORE:: XXXCtrlImpl
Aux App States
Aux App Screen Views
Aux App Screen Controllers
Unity CORE Foundation
Aux App CORE:: AuxApp Manager
Aux App CORE:: GUI Manager
Aux App CORE:: StateTransitionManger
Widget Views
Widget Controllers
Unity CORE Domain Model + Services

Figure 6 Aux App Architecture

The Aux App is logically organized into two layers:

- Aux App – CORE
- Aux App – UI

Aux App - CORE

The Aux App CORE is a common code supporting UI design and navigation paradigm which is tailored via configuration to support different customers.

Each Aux App

- Implements the interfaces for App, Ctrl and GUI which takes care of App transition, GUI transitions and Key dispatch.
- Have its own states that are managed by an instance of the State Transition.
- Have Screen controllers that use specific and reusable widget controllers to handle UI and widget navigation.
- Have / use Business Models to support the different features supported by the Aux App.
- Have Separate configuration files to support tailoring needs.

Aux App - UI

The Aux App UI is specific to a look (customer). Handles

- UI Layout and Definition of UI elements in FLA
- Assets (Images) in FLA
- Presentation of Screen & Widget UI
- Animation definition in .DAE for 3D or in Code or in Timeline

Example of Aux App includes Guide, Library, Store, etc.

5.8 Resource Separation

EPG UI is composed of UI Layout and UI elements such as Images, Strings, Fonts, Colors and Animations.

The UI Layout, color & font definitions are define in flash design file – the FLA.

The element “fonts” in form of TTF is packages separately and not included in the design file.

The element “strings” in form of XML is packaged in the EPG resource folder and is accessed by Framework component – Dictionary.

The element “images” are packaged in separate design file and compiled into a .SWF which will be loaded at runtime by the EPG main application.

These UI resources are decoupled from the EPG code facilitating us to get

- Easy customization of look and feel for various customers
- Easy customization to get different look and feels – skins and themes

5.9 Tailoring

The customization of Unity Flash EPG to support multiple customers with unified code is achieved by the concept of tailoring. The tailoring is classified as follows:

- Feature Configuration

  Every feature in EPG will have the support for:

  - Enabling & Disabling Features and Sub-Features
  - Configuring Data required for Features and Sub-Features

- GUI Configuration

  Includes configuration (add, remove & modify) of UI associated elements:

  - Images
  - Strings Tables
  - Fonts
  - Animations

- Feature Profiling

  Statically includes or omits features (code) when building Unity Flash EPG for a customer.

- Navigational flow changes

  - Across screens
  - Within a screen

Table 3 Tailoring Realization Table

Tailoring Parameters	Realized via
Feature Configuration	Configuration Files
GUI Configuration	Resource Separation & Definition in .FLA
Feature Profiling	Preprocessor Directives & Build Macros
Navigation Flow Configuration	State Transition Table

5.10 3D / 2D Animations

Unity Flash EPG needs to be supported in 3D and 2D mode on STBs with and without OPEN GLES support.

In case of Open GLES support, EPG can have both 3D and 2D animations. In this case, 3D animation will be defined in Cinema 4D and executed by the 3D Engine and 2D animation where ever applicable will be defined in action script or timeline and executed by the Flash Player.

In case of STBs without OPEN GLES support, EPG can have only 2D animation which is defined in action script or in timeline and executed by the Flash Player.

6 Unity Flash EPG Architecture

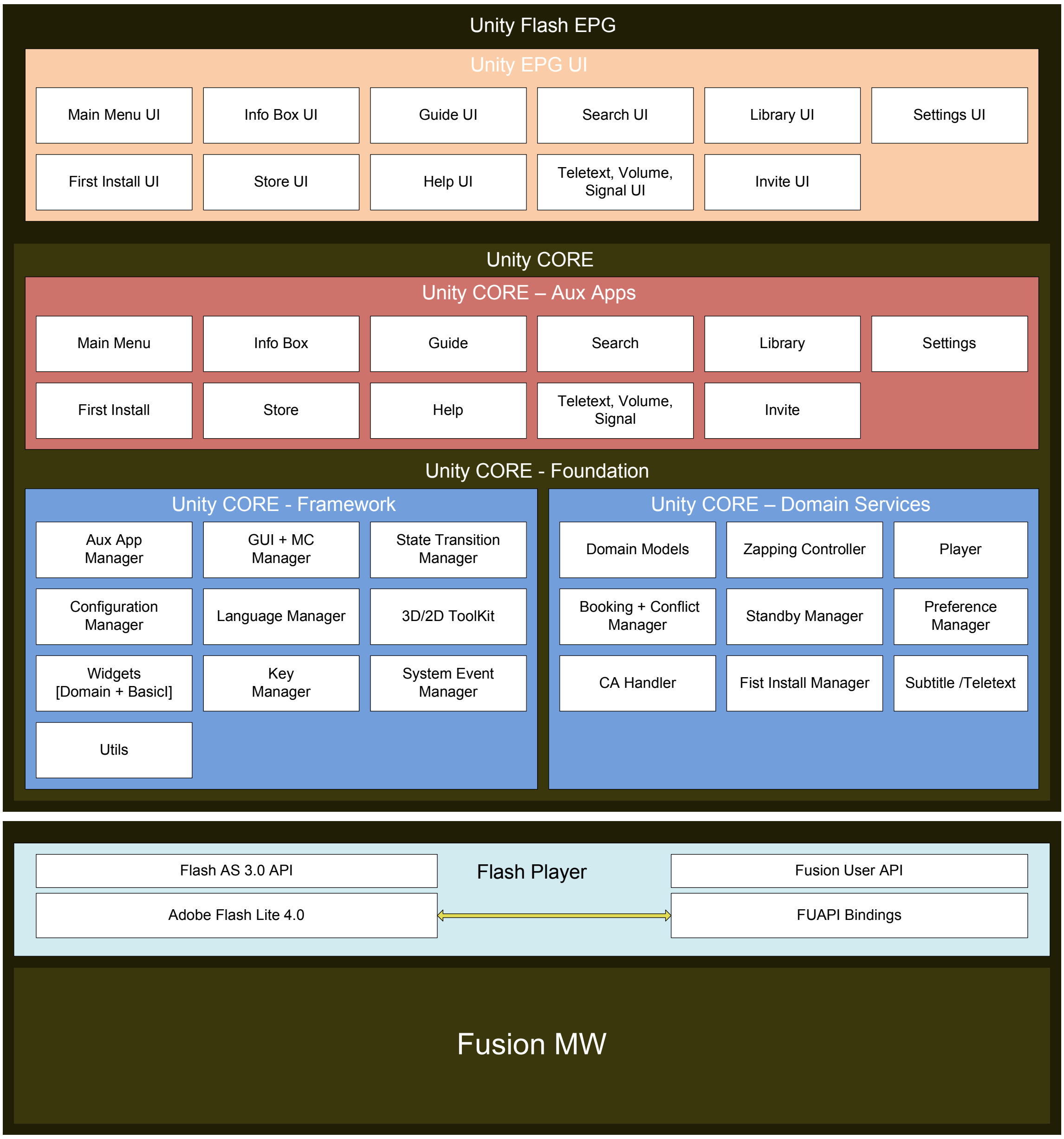

Figure 7 Unity Flash EPG STB Architecture

6.1 Foundation Components

The following table presents the components in the Unity CORE Framework:

Table 4 Unity CORE Framework Components

Components	Description
Aux App Manager	Manages the life cycle of different Aux Apps. Support resident and downloadable apps
GUI Manager	Manages the life cycle of the user interface element – Screen Movie Clips
Aux App State Transition	Manages the State Transition in an Aux App as well as across Aux Apps
Configuration Manager	Manages configuration elements needed for tailoring the EPG
Language Manager	Supports runtime language change of various EPG languages
Toolkit (3D-2D)	Abstracts 3D / 2D animations from EPG code
Widgets	Catalog of Basic and Domain Widgets
Key Manager	Does KEY mgmt and delivery of keys to respective Aux App controllers
System Event Manager	Manages system event delivery to various aux app and controllers
Utils	Has different utilities to support Logging, Timer, etc.

6.2 Domain Services

The following table presents the components in the Unity CORE Domain Service:

Table 5 Unity CORE Domain Service Components

Components	Description
Domain Models	Holds all the domain models that are required for EPG
Zapping Controller	Handles all functionality related to zapping
Player	Handles all functionalities related to playing of content from lib, store, etc.
Booking & Conflict Manager	Handles booking of event and conflict management
Standby Manager	Handles all standby management activities needs to be done by EPG
Preference Manager	Handles get and set of any user preference data to & from the preference store
CA Handler	Handles all CA events from MW and takes appropriate actions
First Install Manager	Handles all configuration and settings related to first installation
Subtitle & Teletext	Handles event related to subtitle and tele text

6.3 Aux Apps

The Aux Apps that are supported in the Unity Flash EPG project and its description are as follows:

Table 6 Aux Apps

Aux App	Description
Main Menu	Presentation of Main Menu and navigates to other Aux Apps
Info box	Handles TV transitions, error message, etc
Guide	Presents EPG Guide with various filters and UI representations
Search	Presents the options to search the contents from Live TV, VOD, library, etc and presents the results to user and handles further action on it
Library	Presents the list of contents that was recorded and stored in the users STB
Settings	Handles configuration of the STB by taking inputs from the end user
First Install	Presents options to do the first installation of the STB
Store	Presents users with options to search and buy content from the network - VOD
Help	Presents context sensitive help to the end user
Volume + Tele text	Controls & presents Volume, teletext
Invite	Handles interactive or 3rd party application launch notification and interaction

6.4 Unity Flash EPG Component Dependency Diagram

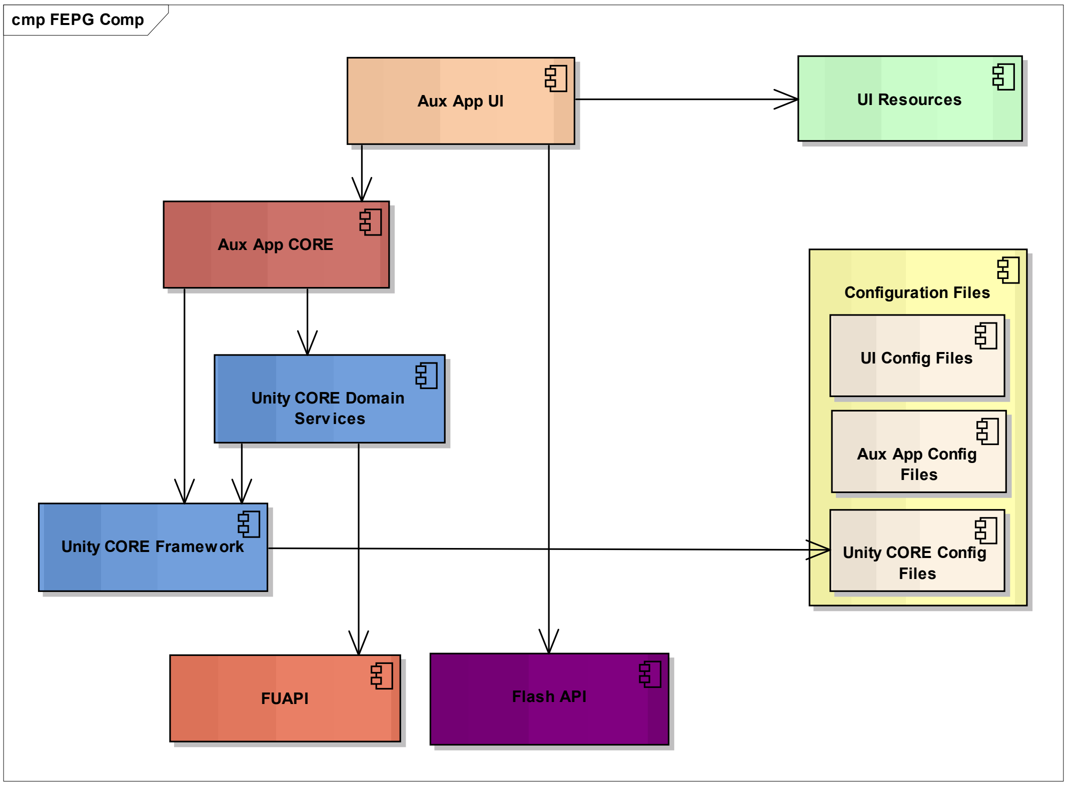

Figure 8 Unity Flash EPG Component Dependency diagram

7 Animation Workflow

7.1 3D Animation Definition Flow

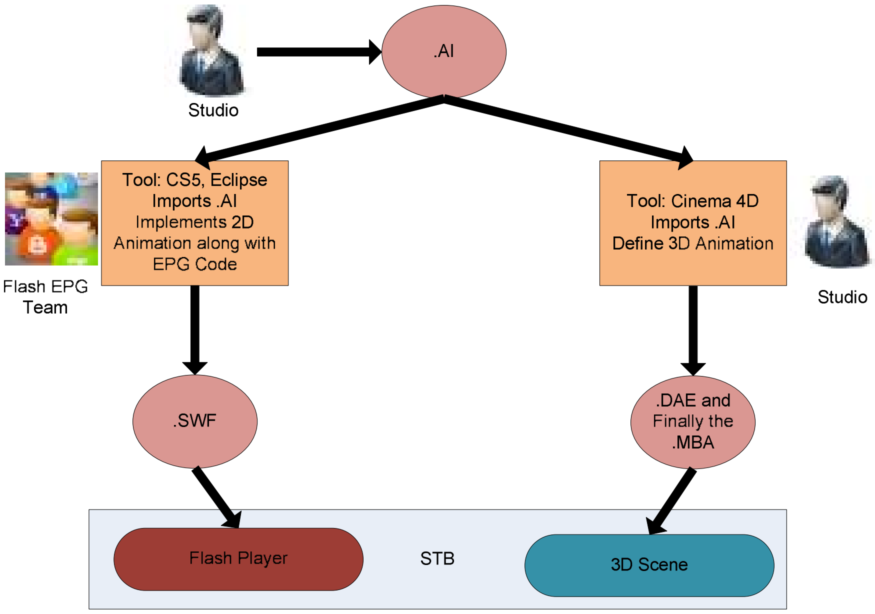

Figure 9 3D Animation Flow

7.2 2D Animation Definition Flow

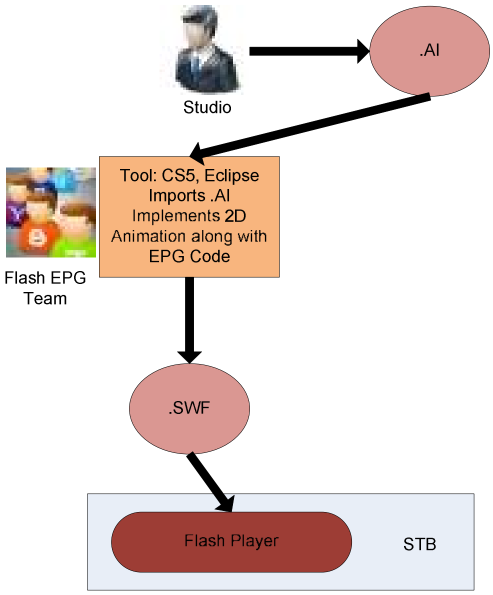

Figure 10 2D Animation Flow

8 Deployment

The Unity Flash EPG modules along with feature configurations are built together into one executable (FEPG.SWF).

The following resources required for the executable are placed in the resource folder:

- String Tables as XML
- Font Configurations as XML
- Font files as TTF
- Images as a separate SWF
- 3D Animations files as .DAE & XML

Note: 2D animations and its related configuration will be built along with the main executable (FEPG.SWF).

8.1 Deployment Platform

The Unity Flash EPG's recommended chipsets and software platform and defined in this section.

Table 7 Deployment Platform

	Software	Recommended Chipsets (Min)
Basic Platform	CDI and NDS Fusion Middleware Adobe Flash Lite 4.0 Player	BCM 7335 ST 7104
3D UI	ARM Colloda Engine Open GLES 2.0	Chipset supporting Open GLES 2.0 specification such as Intel Groveland, ST Shiner, etc.

9 Development Tools

9.1 Design Tools

- Adobe Illustrator

  Studio Design Team uses Illustrator to define the images to be used in the Flash EPG.

- Adobe Cinema 4D

  Studio Design Team uses Cinema 4D to define the 3D animations to be used in the Flash EPG.

9.2 Development Tools

- Adobe CS5

  EPG Team uses CS5 to define the UI layout, Screen and Widget UI definitions, 2D animation, etc.

- Adobe Flash Builder / FDT

  EPG Team uses Flash Builder to develop the EPG in action script.

10 Test Framework

Note This section will be updated in future release of this document.

Change History

Date	Revision	Author	Change
09.Feb.11	0.1	Suresh Gopathy	Created the first draft of the Unity Flash EPG Architecture
18.Feb.11	0.2	Suresh Gopathy	Review and Update from Prashanth Nanjundappa, Madhusudhan Peta
10.Mar.11	0.3	Suresh Gopathy	Review and Update from Arunkumar Kantharaj, Sushma Rai, Mahesh Agarwal