DragonFly HTML Player and Flash to HTML conversation

# High-Level Technical Design

Version 1.0

1st July 2016

## Introduction

This document specifies technical design to develop a HTML Player (Player). Player is responsible to load and run the html content which includes html, javascript and CSS. Player also provides global navigation to jump through different type of HTML content/templates.

### Technical Architecture

Player is developed using HTML 5 technologies i.e. JavaScript and CSS. Below is the list of core components of the player

* DEFApplication
* CourseController
* NavController
* SubNavController
* AudioManager
* AbstractPage
* SwiffyController
* QuizController
* TutorialNumeralsController
* DeviationController
* ApplicationController

## Player initialization process

NavController

Create UI

UpdateUI

Boards

Search

Tutorial

Lecture

**Accordion**

ChapData.xml

Next

Back

Sub Nav

Tutorial

Quiz

Numerals

Page

Load Controller

Load Assets

CourseController

DEFApplication

**Core Classes and Controller:** These are the core classes and controllers of the framework. Responsibilities of each of these classes are detailed below:

**DFEApplication:**

DFEApplication JS is the entry point to the entire Player system. DFEApplication initializes CourseContoller and NavCotroller. These two controllers are responsible for initializing the player.

**CourseController:**

CourseController is responsible for gathering required resource information to be loaded and rendered by a page. It initializes the AbstractPage object and provides it with the required resource information. It also provides API to access page information and load/unload page. CourseController has Singleton implementation.

**NavController:**

*NavController* is responsible for creating navigation UI and provide API to application to jump between pages. *NavController* has dependency on an xml file (i.e. *ChapData.xml*) to create navigation UI. *ChapData.xml* has required information regarding different types of content and the location of assets. *NavController* has Singleton implementation.

**AudioManager:**

As part of Flash to HTML process all animations are converted to SVG format. Audio in the animation is exported as mp3 format. New templates are created to load and run these animations and play mp3 audio files. *AudioManager* is a Singleton and is responsible to loading as well as playing audio files. It also provides API’s to play/pause, stop and restart the audio, it dispatches various events which help in audio and animation synchronization.

**AbstractPage:** This class is responsible for loading the XML data and rendering the HTML view. It iterates the XML data, initializes the required components / controllers, and provides them with the required data. When all components / controllers finish loading, it notifies the *CourseControlller* to display the page.

**Template Controllers:** Below is the list of page level controllers developed to control and manage different component and page types.

**QuizController:** This Controller is responsible for controlling quiz questions.

**TutorialNumeralController:** This controller is responsible for handling tutorial and numerals behavior.

**SwiffyController:** This controller is responsible for registering and linking the *SwiffyWidget* with the *AudioPanel* component.

**DerivationController:** This controller extends the *SwiffyController* and is responsible for loading sub content of Derivation pages. Apart from registering and linking the *SwiffyWidget* with the *AudioPanel* component, it registers and handles the events of *DerivationPanel* view component for loading the Derivation sub-pages.

**ApplicationController:** This controller extends the *SwiffyController* and is responsible for loading sub content of Application pages. Apart from registering and linking the *SwiffyWidget* with the *AudioPanel* component, it registers and handles the events of *ApplicationPanel* view component for loading the Application sub-pages.