Player studio

Design document

Itay Kinnrot, Michal Radwantzor, Ran Yefet | 15/9/2013

# Main goal

The new player studio should replace our current studio to work with the new html5 player,plugins.

If old player will opened using the studio – the user will see a message that we’re upgrading his player and some feature might not work.

The new studio will be protected using permission that will be able us to monitor who can use it – for the beta phase.

# Architecture

## General:

The framework we chose to work with is angularJS which is a testable single page application framework.

The application will be single page based where only data and templates and JavaScript will go over the wire.

The application will be iframed in the KMC in order to allow users to embed it on other application (KAF).

Parameters we’ll get in order to open the studio are KS & partner id.

Each screen will be available using REST (/player/list, player/new, player/edit/:id …)

We’ll have a service to get / save /create & list the uiconfs.

We’ll download all the Uiconfs object to the client and cache them.

## Core Changes (Backend TODO).

### Resources:

Each player configuration can hold resources like css,js,fonts and images. We need a way to connect uiconf to this folder and an API to manage this resource folder. (Backend TODO – Security- need to load from cookieless domain + file version).

### User tags:

In order to help users search their players and us to control of the feature enabled for each player we need 2 tags fields – now we have one.

1. user tags (visible to user)
2. pluginsTags (not visible, will be used only for search) (Phase II)

### Field filter:

API – add filter to fields – we need specific fields from the uiconf API.

### Permissions:

1. Permission to show new studio
2. Permission to show both studios? (TBD)
3. Permission to show features in "beta"

### Save uiconf versions

for easy rollbacks

## Which player studio to open?

KMC will open old / new player studio according to a new user permission.

*TBD: maybe have the option to open both studios?*

## New Player Studio

Each new studio version will be associated with a target HTML5 lib version. It will probably always be the latest released one. HTML5 version should be configurable. (how?)

Once the first "edit player" window is opened the studio will load the featuresConfig.json file that will be part of its HTML5 lib.

### Manifest - featuresConfig.json

* All the player configuration options will be pulled from featuresConfig.
* Each plugin/component in our player have parameters the user can change and we save it with the plugin code.
* Each plugin/component has visibility level, defines which users can see the feature: All / Beta.

Studio will display the plugins according to this field and the user permission.

* Each parameter will be from some type: number/color/text/url/selection. And we’ll create the menu dynamically from the JSON.
* The manifest structure will be hierarchic and will contain default values.

For example, this featuresConfig contains two plugins - nielsen and comscore, that will be under "Analytics" tab:

{"properties": [

{

"label": "Analytics",

"type": "menu",

"model": "basicDisplay",

"children": [

{

"label": "Nielsen",

"type": "checkbox",

"default": true,

"model": "info"

},

{

"label": "Comscore",

"type": "checkbox",

"default": true,

"model": "share"

}

}

}

}

### Glossary

|  |  |  |
| --- | --- | --- |
| Name | Description | Used by |
| featuresConfig.json | JSON file that contains all available features. Will be Included in HTML5 lib. | Create new player, Edit player |
| defaultConfig.json | Json file that represents default player config. | Create new player |
| tempConfig.json | Json file that represents the current preview player config. Being updated upon feature change. | Edit player |
| tempFeatures.json | Json file that represents the current available features, merged with the values from the current config | Edit player |

## Scenarios

### Open studio

list uiconfs, load HTML5lib (to retrieve featuresConfig.json), show table

### Create New Player

* load template selection screen: (Phase II)
  + API call: uiconf --> list with "template" tag filter (Not sure: if a partner can create his own templates we should add current partnerId to the filter)
* same flow as "open edit player" only temp.json will be copied from defaultConfig.json

### Open Edit Player

When user opens the "edit player" page the following actions will occur:

1. API call: uiconf-->get(uiconf ID)
2. take the returned uiconf's JSON and copy it into temp.JSON. temp.json is the config that will be used by the embedded preview player:

kWidget.embed({

'targetId': 'myVideoTarget',

'partnerId': [pid],

'wid': \_[pid],

'uiconf\_id' : [uiconfId],

'entry\_id' : [entryId],

'flashvars':{

'confFilePath': 'temp.json'

}

})

1. Merge featuresConfig and uiconf JSON into tempFeatures.json. tempFeatures.json is actually the json that will be used to display features menu, and it will contain default values from the uiconf JSON so all existing configuration will be reflected by the menu.

### Player update

*Each plugin will implement configurationUpdate function and will make sure to update the relevant data/UI.*

*Each time property will change we’ll use kdp.setAttribute in order to update the plugin and in addition we’ll update the configuration json.*

*Some changes will required reload the player – we’ll have an attribute in the manifest that will indicate that reload is required.*

*(TBD - maybe we will always reload the player, then the above logic is unnecessary)*

Every change we perform will affect on temp.json. Only upon save we will call uiconf-->update with temp.json as the new json config.

### Save Player

1. (Phase II) go over all plugins in temp.JSON add create the matching pluginsTags string. For example, player with playlist and vast plugins will have the following pluginsTags: playlist,vast
2. API call: uiconf-->update with temp.json and pluginsTags

### Advance edit (css,js)

In this section we’ll let the user advance edit his player.

Add or replace css, add javascript images and fonts.

We’ll use the new API in order to upload files to the player.

All the files reference will be saved in the player configuration.

We’ll include them in the player load in the preview and in production we’ll add them using our resource manager. (All in one request).

In order to edit the files the user can upload his own ready files or edit them online. (<http://ace.c9.io/#nav=about>)

TBD – Product need to define this screen

## Unit Tests:

* Test merge between featuresConfig and JSonConfig.
* Test pluginTags creation

Test method that will affect temp.json (this method will be used when someone edit the features)

## Open issues (for phase I):

* player thumbnail: shall we enabled user to change it? how?

# Development phases

## Phase 1:

* Player list.
* Player upgrade.
* Basic search.
* Breadcrumbs
* Edit player
  + Plugin configuration.
  + Basic color changes.
* Basic Preview screen

## Phase 2:

* Tags
* Advance search.
* Advance preview.
* Templates screen
* Advance edit (css,js).

# QA Guidelines

## Browsers: (Michael TBD)

* IE8 +
* iPad

## Performance:

* Make sure the studio is responsive with minimal waiting time.
* Player load time in preview.

## Data integrity:

* Verify any data change in the player – being saved and viewable.