NDL Layout Tool

Product Physical Design

June 2016

By: Florin Iacob

V1.0

Revision Table

| Ver. | Date | Updated by | Updated Sections | Comments |
| --- | --- | --- | --- | --- |
| **1.0** | **14.06.2016** | **Florin Iacob** |  | **New Document** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Table of Contents

[1. Technology, Frameworks & patterns 4](#_Toc453931117)

[2. Features 4](#_Toc453931118)

[3. Loading a navigation plan 6](#_Toc453931119)

[3.1. NavigationPlanGateway reading process 7](#_Toc453931120)

[3.2. NavigationPlanParser 8](#_Toc453931121)

[3.2.1.](#_Toc453931122) **[XMLNavigationPlanParser](#_Toc453931122)** [8](#_Toc453931122)

[4. Apply the navigation plan changes 9](#_Toc453931123)

[4.1. SkinDefinition Converter 10](#_Toc453931124)

[4.1.1.](#_Toc453931125) **[XmlSkinDefinitionConverter](#_Toc453931125)** [10](#_Toc453931125)

[4.2. Apply changes for ViewMockup 11](#_Toc453931126)

[4.3. Apply changes for saving 12](#_Toc453931127)

[5. Layout designer 13](#_Toc453931128)

1. Technology, Frameworks & patterns

Layout Tool application is developed using the following:

* Microsoft .Net Framework 4.5
* C# language
* WPF (Windows Presentation Foundation)
* PRISM for WPF with Unity
* MVVM design pattern
* NUnitFramework unit testing framework
* NSubstitute isolation framework
* Click-Once deployment

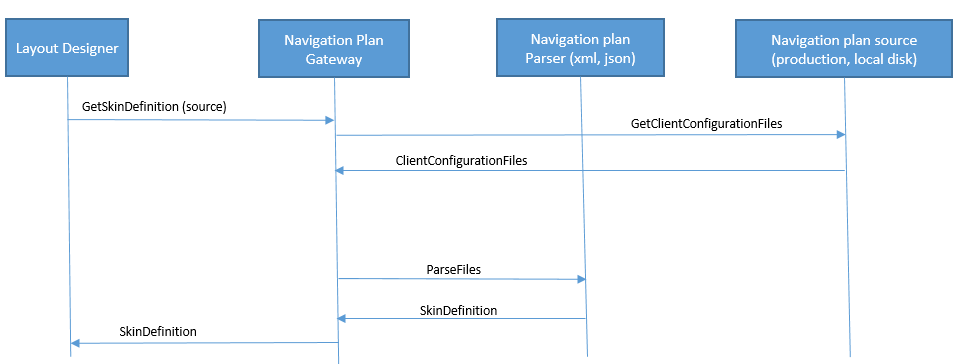
1. Features
   1. Lobby features

* Reorder arenas
* Replace arena (with one of the client supported arenas or with a game)
* Add, remove, reorder Top Games
* Add, remove, reorder VIP Games
* Reorder, remove MyAccount Lobby items
* Reorder, remove MyAccount History items
* Default Favorite games – will be postponed.
  1. Arena Features
* Enable/Disable the jackpot display
* Add, remove and reorder games
* Add, remove and reorder filters
* Add, remove and reorder Also Playing games
* Add, remove, reorder and RESIZE grid view columns.
  1. Game level features
* Mark a game as being new
* Mark a game VIP only
* If a game is placed in the NewGames arena automatically mark it with New.
* IsNewGame status is synchronized between arenas
* IsVIPGame (userMode) is synchronized between arenas.
  1. Validations
* Selected games are supported by the target brand (client).
* newGame detection in games properties.
* Don’t let the user add games to the Top/VIP games if the game is not in any arena.
* A game in the NewGames arena cannot be marked as NOT being new.
* Also playing games should contain only games from that owning arena
  1. Dynamic Layout

Has the same features as a static layout (reorder arenas, add, remove, reorder games/filters/also playing etc.) but also has the **Dynamic Layout Parameters** feature.

* 1. View Mockup
* Shows the changes in the real client using the MainProxyMock
* Allows to view the mockup for a different skin than the one started from.
* Allows viewing the mockup for dynamic layout by allowing the user to supply values for the dynamic layout parameters.
  1. Save
* Allow the user to save the changes for a different skin than the one selected for the initial layout
* Provides validations that apply to the target skin
* Save the layout in the Layout Tool proprietary format
* Save the layout in the real client navigation plan format
* Errors and Warnings are written in the LayoutTool proprietary format file.

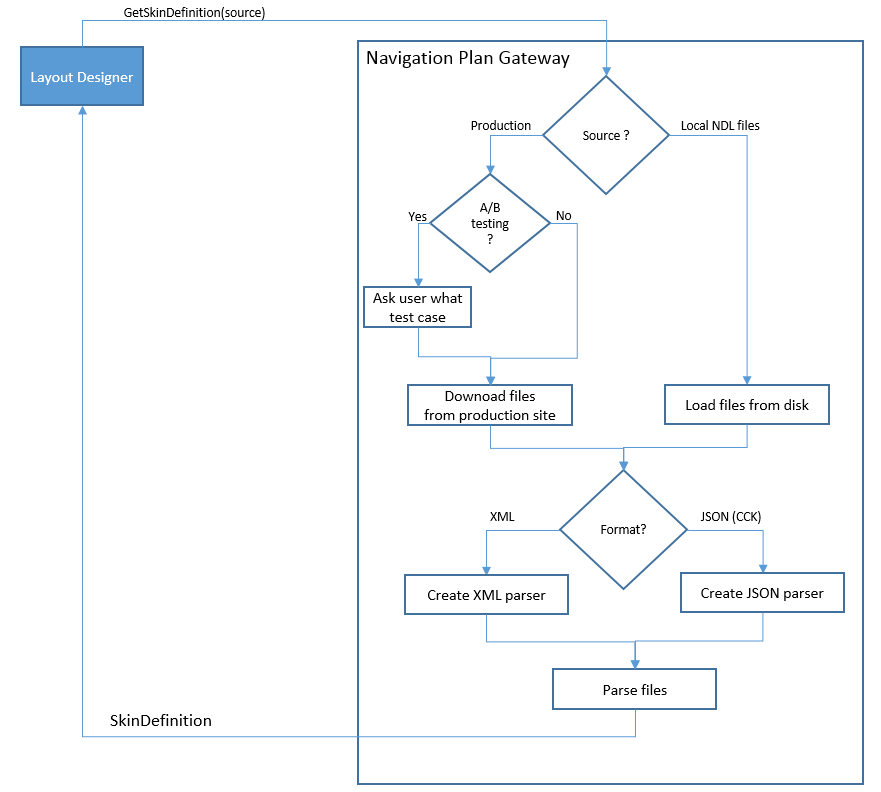
1. Loading a navigation plan



The Layout Designer will allow the user to choose the source from where to load a navigation plan. Initial implementation will have two options: Production and Local disk. Once the user selects the type of the navigation plan source it will call a component named NavigationPlanGateway in order to retrieve an in memory representation of the navigation plan. In other words a navigation plan object model which we will name it **SkinDefinition**. This way the Layout Designer will be decoupled from the actual location & storage format of the client configuration files.

According with the source provided by the Layout Designer, the NavigationPlanGateway will load the **relevant client configuration files** and according with the files format invokes the right NavigationPlanParser in order to construct the SkinDefinition. Once the SkinDefinition is built it is returned to the LayoutDesigner which can manipulate it by adding/remove/reorder games in different layout sections.

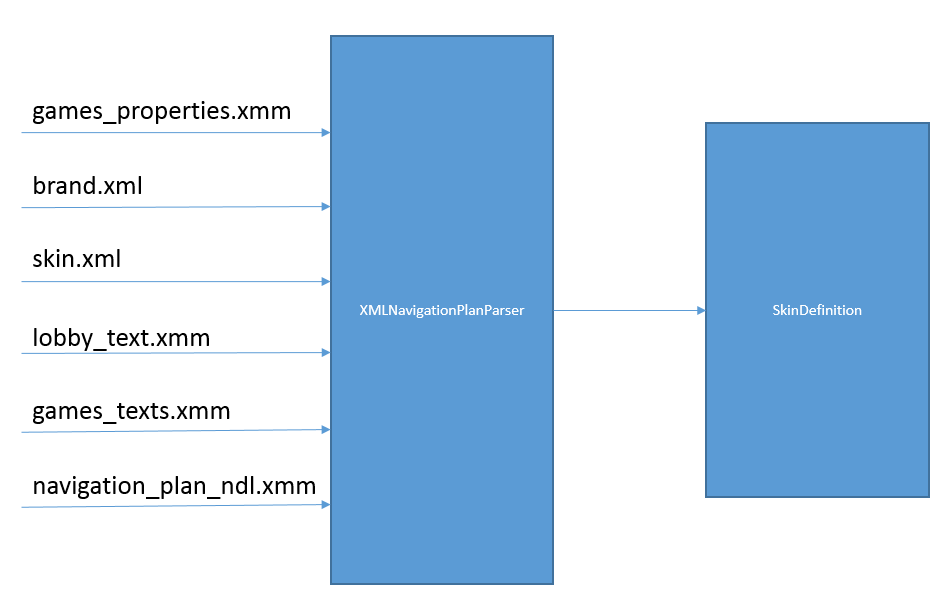
* 1. NavigationPlanGateway reading process

  
When the Layout Designer ask for a SkinDefinition from the NavigationPlanGateway it will provide the source (production or local NDL) and the correct parameters (Brand + Skin for production and file path for a local NDL navigation plan). In case the source is Production then the NavigationPlanGateway must check if there is A/B testing enabled for the requested Brand & Skin. If there is a test case then the user must specify the desired A/B test case. Once the NavigationPlanGateway will load the files either from production or from local disk according with the files format it will create the corresponding parser and pass the file content to it in order to build the SkinDefinition object model. The skin definition built by the parser is returned to the Layout Designer.

* 1. NavigationPlanParser

NavigationPlanParser will have more implementations according with different files format supported by the NDL client. Initial implementation will support XML files format parsing. When the CCK project will be released then we will add a second NavigationPlanParser implementation for the JSON format.

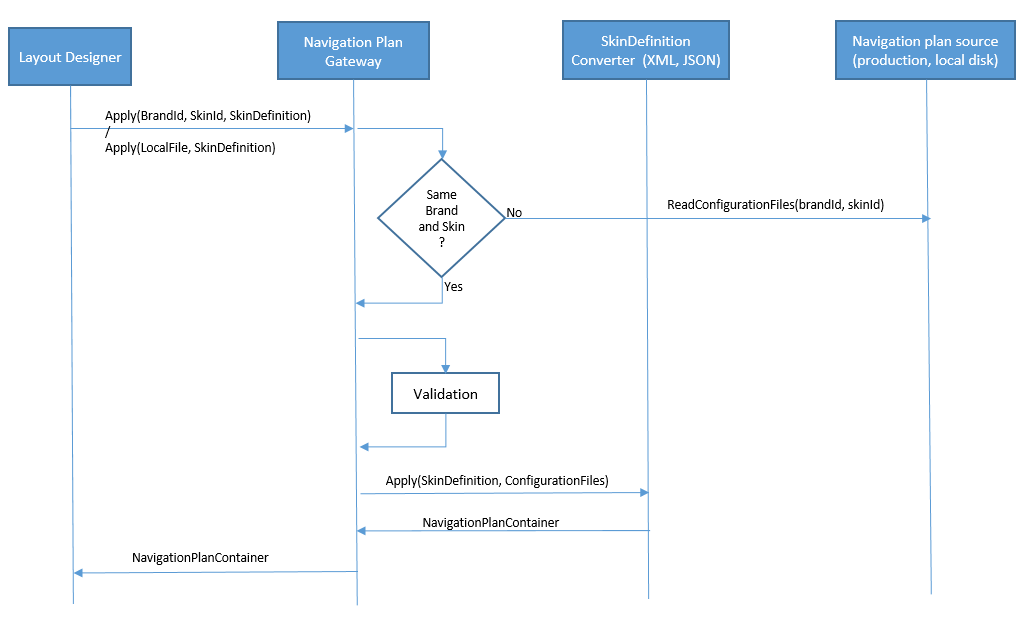
* + 1. **XMLNavigationPlanParser**

XMLNavigationPlanParser takes as input the content for the following client configuration files: games\_properties.xmm, brand.xml, skin.xml, lobby\_texts.xmm, games\_texts.xmm, navigation\_plan\_ndl.xmm.   
Each of these files provide information necessary for the LayoutTool UI.   


The table below lists the files consumed by the XMLNavigationPlanParser and the information that each file provides for the layout tool.

|  |  |
| --- | --- |
| **File name** | **Information provided** |
| games\_properties.xmm | The list of possible games and arenas to choose from. |
| brand.xml | The list of default favorites games |
| skin.xml | The dynamic layout triggers |
| navigation\_plan.xmm | The Layout structure (arenas, top games, VIP games etc.) |
| lobby\_texts.xmm | The friendly name of the arenas, filters, my account menu items. |
| games\_texts.xmm | Games friendly names the way they appear in the client. |

1. Apply the navigation plan changes

  
We need to **“apply”** the navigation plan changes in order to save it or to view the mockup.   
The LayoutDesigner will use the NavigationPlanGateway to apply the changes to a target skin. This can be a skin in production or a skin locally on the disk, it can be the same skin used for reading the navigation plan or it can be a different one. Before actually applying the skin definition to the files some validation will be performed. These validation should check the following:

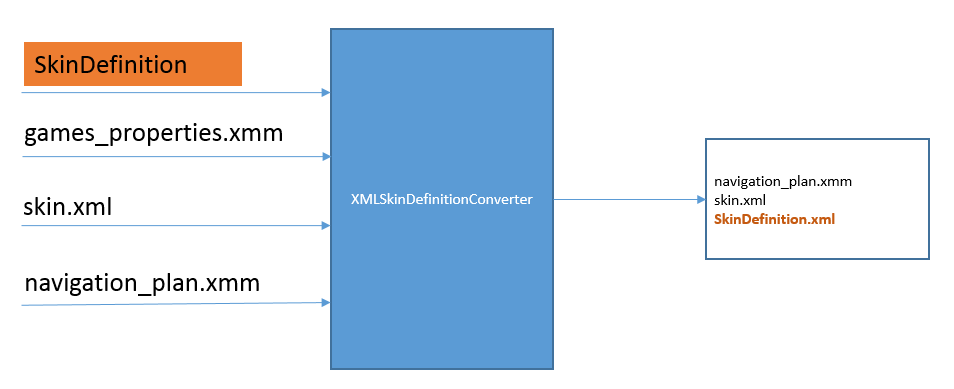
1. The games in the SkinDefinition are available in the client that supports the target skin
2. The newGame attribute set in the navigation\_plan is not overridden in the games\_properties file.

After the validation the NavigationPlanGateway will create a SkinDefinitionConverter according with the target skin files format. The SkinDefinitionConverter will apply the changes on the client configuration files and output new files and wrap them into a NavigationPlanContainer data structure. The NavigationPlanContainer is then returned to the LayoutDesigner.

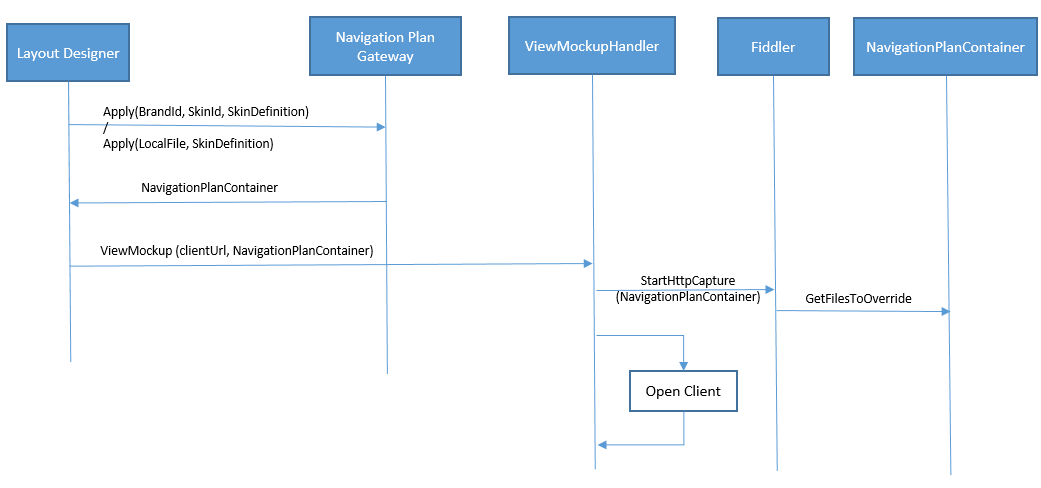
* 1. SkinDefinition Converter

Just like the NaviationPlanParser there will be one implementation for each client configuration supported format. The first implementation will be the XmlSkinDefinitionConverter and when the CCK project will be ready we will add another one.

* + 1. **XmlSkinDefinitionConverter**



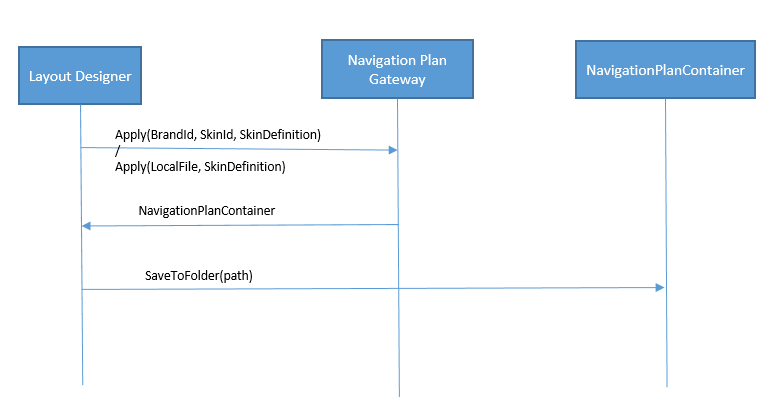
* 1. Apply changes for ViewMockup



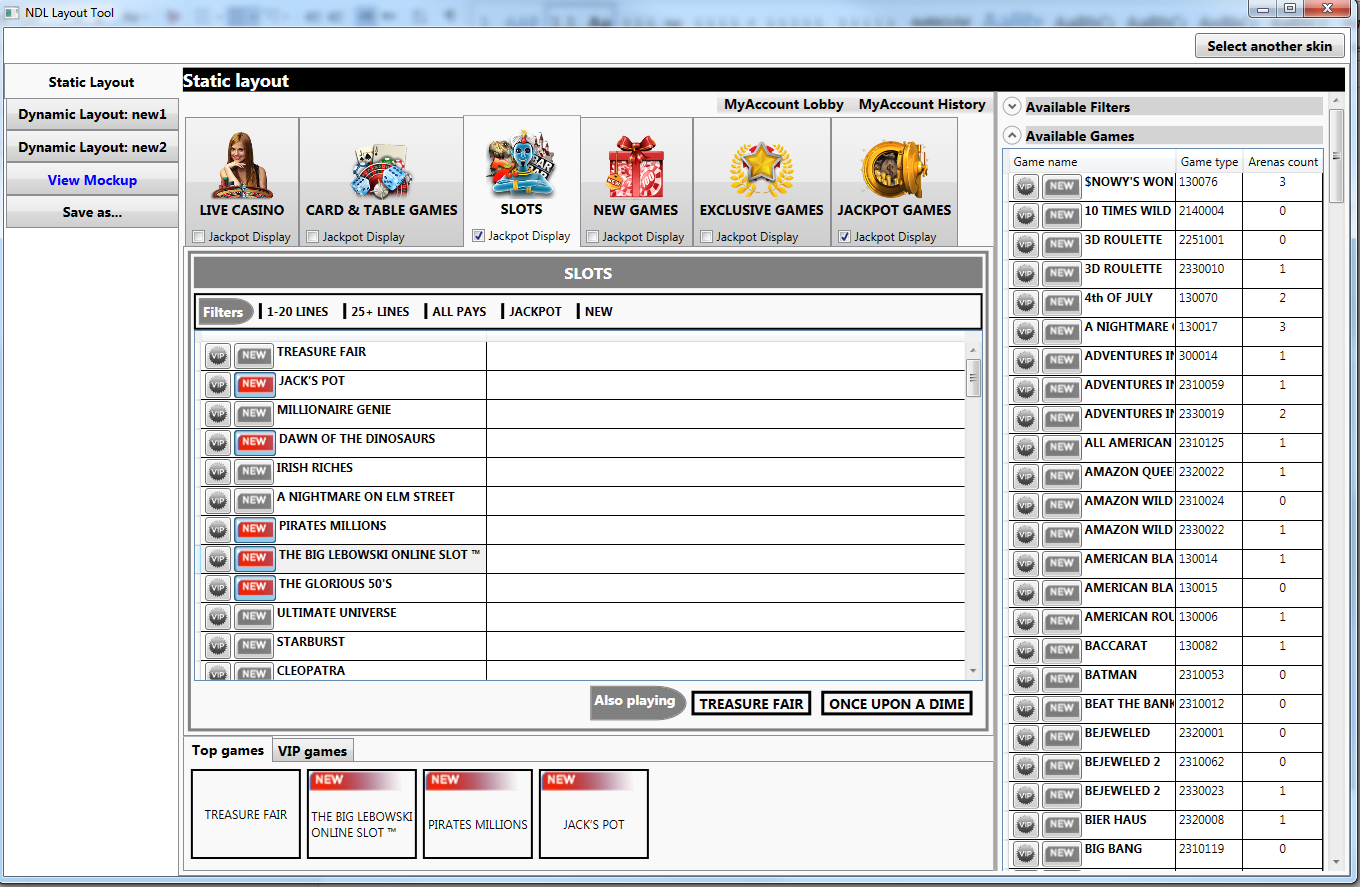
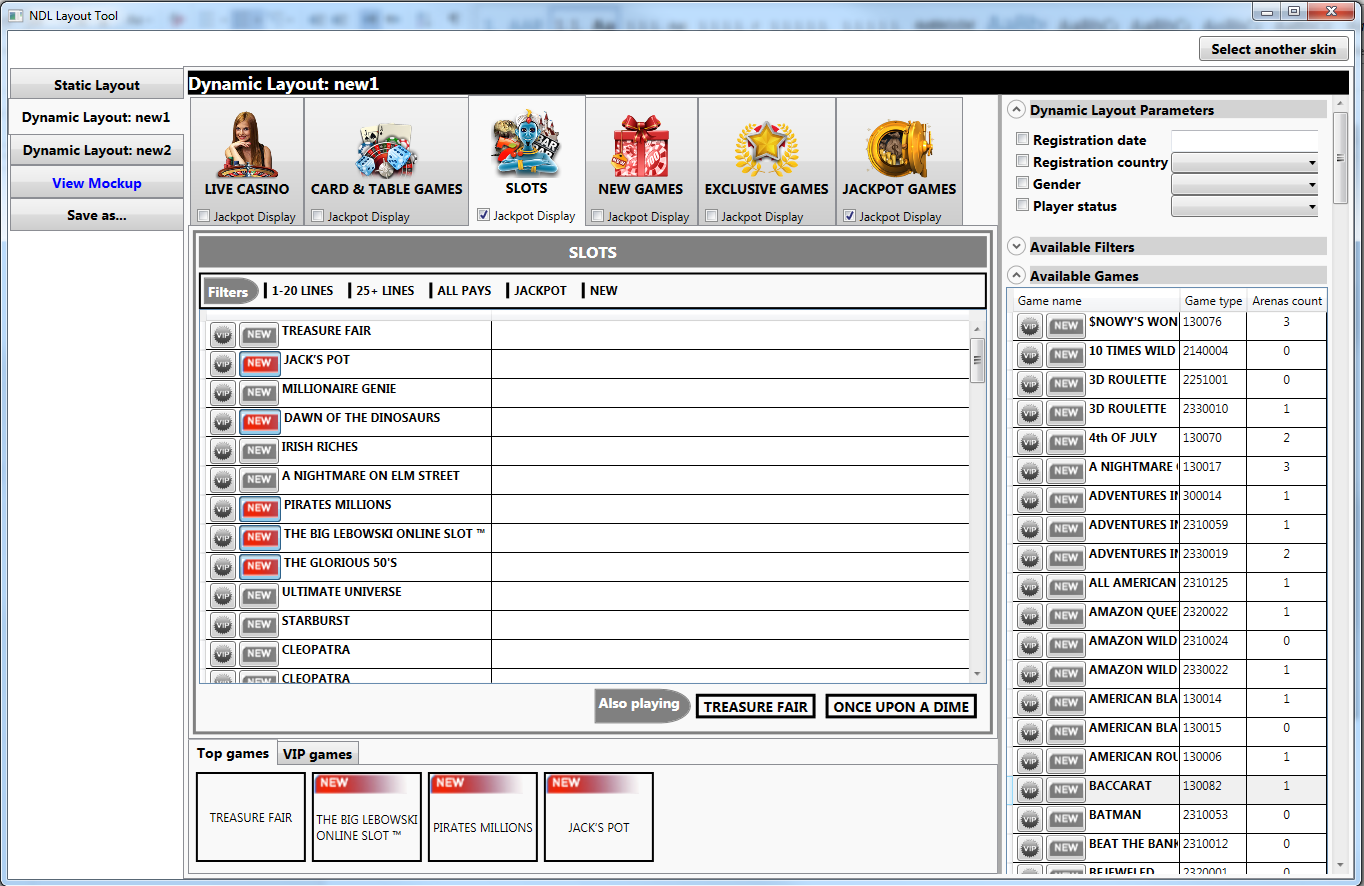
In order to view mock-up we need to apply the current skin definition to the target skin. This is done by the LayouDesigner by calling the NavigationPlanGateway in order to retrieve a NavigationPlanContainer.

Handed with the NavigationPlanContainer the LayoutDesigner can call ViewMockupHandler providing the NavigationPlanContainer and the desired client url to use for viewing the mockup. The ViewMockupHandler will use the FiddlerCore .Net to capture the HTTP traffic and feed the client with the in memory navigation plan files extracted from the NavigationPlanContainer.

* 1. Apply changes for saving



In order to save the navigation plan to disk again the LayoutDesigner first needs to apply changes using the NavigationPlanGateway and then instruct the NavigationPlanContainer to persist the files in a specified folder.

1. Layout designer
   1. Static Layout designer  
      
   2. Dynamic Layout designer  
      
   3. View Mockup  
      