Web Client Configuration Files

Configuration Files

Three levels of configuration exist within the i2b2 web client:

1. Configuration of web proxy and connection to various Hive PM Cells
2. Registration and configuration of code modules (plug-ins / Cells)
3. Configurations for individual code modules.

All configuration files are written in JSON (JavaScript Object Notation) and dynamically loaded by the framework using AJAX calls.

Hive Connection Configuration

All deployments of the i2b2 web client will require modification of the Hive Connection configuration file. This configuration file is located at **/i2b2_config_data.js**. An example file is as follows:

```
{
   urlProxy: "http://localhost/services/proxy/Service.asmx/webProxy" ,

   urlFramework: "js-i2b2/",
   //--------------------------------------------------------------------------------
   // THESE ARE ALL THE DOMAINS A USER CAN LOGIN TO
   lstDomains: [
   { domain: "HarvardDemo",
     name: "VM Java 1.3 RC5",
     project: "Demo2",
     debug: true,
     urlCellPM: "http://services.i2b2.org/PM/rest/PMService/"
   },
   { domain: "HMS_JAVA" ,
     name: "Harvard Demo (Java 1.2)",
     debug: true,
     urlCellPM: "http://services.i2b2.org/PM/rest/PMService/"
   }
   //--------------------------------------------------------------------------------
}
```

The first configuration attribute **urlProxy** must be changed for every deployment. It should be set to point to *your* web proxy cell. In all cases it must be the same base URL (hostname and port) as the website that serves the default.htm file. For more information on the web Proxy communication with the hive please see the document called *Web Client Architecture Guide*.

Name	Null	Type	Description
urlProxy	N	String	The full path URL for the i2b2 web services proxy server.
urlFramework	N	String	The full path URL to the root js-i2b2 directory.
lstDomains	N	Array	An array containing 1 or more domain definition data objects.

The domain definition data object contains the following attributes for each data object:

Name	Null	Type	Description
domain	N	String	A short code used by the proxy server for the domain/group ID.
name	N	String	A human-readable string containing the domain's name.
urlCellPM	N	String	The full path URL that should be used by the back-end PM Cell.
project	Y	String	Login to a specific project without prompting the user to select one from a list.
isSHRINE	Y	Boolean	Should this domain use SHRINE processing functions?
debug	Y	Boolean	Are debugging messages logged? (uses additional memory).

Module Loader Configuration

The framework is aware of various code modules after they are registered in the main component list configuration file. This list is located within the **/js-i2b2/i2b2_loader.js** file in a section containing *JSON-based configuration information* which has the following structure:

```
// THESE ARE ALL THE CELLS THAT ARE INSTALLED ONTO THE SERVER
i2b2.hive.tempCellsList = [
   { code: "PM",
   forceLoading: true // <------- this must be set to true for the PM cell!
   },
   { code: "ONT" } ,
   { code: "CRC" } ,
   { code: "ANALYSIS" ,
      forceLoading: true,
      forceConfigMsg: {
      params: []
   }
  }
];
```

This JSON structure is used to register a list of cells / plug-ins that are able to be loaded if the user has been authorized to use them (via the data returned from the *Project Management* cell during successful login). The above code listing has information for registering the following cells / modules (in order):

1. Project Management Cell (forced to automatically load when the framework is loaded)
2. Ontoloty Cell
3. Data Repository Cell
4. Plugin Viewer Module (used to manage all non-cell plug-in modules)

Unless a custom module is going to be running as an i2b2-compliant Cell module, further configuration options must be included in this file using the "forcing" options below:

Configuration Option	Description
forceLoading: (Boolean)	Is the module automatically loaded during framework initialization
forceConfigMsg: (Object)	This data object is automatically populated to **i2b2.CELLCODE.cfg.config** when the module is loaded

These options can also be used to override configuration information that is being returned to the web client Framework from the Project Management Cell during login authorization. The purpose of the **forceConfigMsg** setting is that it will force information to be automatically loaded and will create a configuration value for later use. For additional information please see the document called *Web Client Plugin Developers Guide*.

Plug-in Configuration

For the framework to be able to properly load a plug-in module, information which defines the new module must be provided. This is accomplished by creating a JSON-based configuration file within the plug-in's root directory. An example would be located at:

/js-i2b2/cells/plugins/examples/ExampHello/cell_config_data.js

This file would have the following structure:

```
// This file contains a list of all files that need to be loaded dynamically
// for this module. Every file in this list will be loaded after the module's Init()

// function is called
{
   files:[ "ExampHello.js" ],
   css:[ "ExampHello.css" ],
   config: {
      // additional configuration variables that are set by the system
      short_name: "Hello World",
      name: "Example #1 - Hello World",
      description: "This plugin cell demonstrates how to register your plugin
            with the i2b2 thin-client framework and display simple HTML.",
      category: ["celless", "plugin", "examples"],
      plugin: {
         isolateHtml: false,
         html: {
            source: 'injected_screens.html',
            mainDivId: 'ExampHello-mainDiv'
         }
      }
   }
}
```

The configuration file has three main parts to it:

1. A list of JavaScript files
2. A list of HTML CSS files
3. A configuration section.

The files and CSS configuration sections are self-explanatory. All filenames listed will be prepended the with the plug-in's base directory to generate the full file access location used by the framework in loading the files.

Be Careful

It is important to note that all version of Microsoft Internet Explorer limit the number of dynamically loaded style sheets to a total of 31 files. The web client framework uses several style sheets and each cell or plug-in may have dynamically loaded style sheets as well.

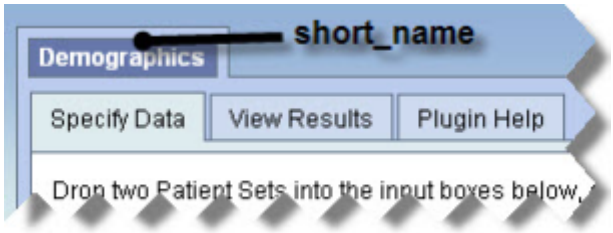
An important best practice is to only define one CSS file in your plug-in's configuration file.

To use more than one CSS file in your plug-in, create a CSS file to subsequently load your other CSS files using the @import (file.css) command.

The configuration section contains various pieces of information that are used by the framework. They are explained below:

JSON Configuration Variable	Description
config.short_name	Is displayed in the title tab area of the plug-in viewer's display window.
config.name	The title string that is displayed in the plug-in viewer's listing window.
config.description	The description that is displayed in the plug-in viewer's listing window.
config.category	A list of categories that this plug-in is a member of. All plug-ins must include "plugin" value. If the plug-in does not have its own backend cell then "celless" value should also be present.
config.icons	JSON object defining one or more icon files. These files must be located in the **assets** directory of the plug-in's base directory.
config.icons.size32x32	Filename for 32x32 pixel icon used in the plug-in listing window when in detailed view mode. (The fill must be in the plug-in's **assets** directory).
config.icons.size16x16	Filename for 16x16 pixel icon used in the plug-in listing window when in summary view mode. (The fill must be in the plug-in's **assets** directory).
config.plugin	JSON object which defines and configures the module as a plug-in.
config.plugin.isolateHtml	Boolean, should the framework isolated the plug-in's HTML an IFRAME.
config.plugin.html	JSON object that contains information about the plug-in's display HTML.
config.plugin.htm.source	Filename for the plug-in's display HTML (in the local **assets** directory).

config.plugin.html. mainDivId	The unique ID of the HTML Element (in the above declared source file) whose contents will be initially displayed.



Most of the information within the configuration section is used by the plugin viewer subsystem in ways that are reflected in the user interface.