HSBCNet VA Panel Integration

This will document the process and changes specifically for HSBCNet

Major Breaking Changes for v2.x

- The code itself has now been refactored into a self-contained module called LivePersonVirtualAssistantModule.
- All code relating to manipulation of styling / showing and hiding of the panel has been **REMOVED**
    - this should now be handled directly by the VA Panel code within CV that listens to specific events raised by this module to indicate when to show/hide the panel based on the interaction with the chat window.
- Events are now triggered by the module when actions should be taken or the button status has changed.

"*When do I start the LivePersonVirtualAssistantModule ?*"

- The module has a .init function that should be called **every time the VA Panel is loaded onto a page**
- This is crucial so that it can bind to key events taking place within the LivePerson chat window and lpTag and report these to the VA Panel

Use the following code to enable the LivePersonVirtualAssistantModule on each page inside the VA Panel

```
LivePersonVirtualAssistantModule.init();
```

"*How do I know when to show/hide the Panel?*"

The LivePersonVirtualAssistantModule broadcasts events when key actions/interactions are made with the LivePerson chat window.

This is done via the lpTag.events function using a specific namespace for all VA Panel events.

Binding to LP_VA_PANEL_MODULE events

- Bind to all the events fired by the LivePersonVirtualAssistantModule within the LP_VA_PANEL_MODULE namespace as follows:

```
lpTag.events.bind('LP_VA_PANEL_MODULE', '*', function
(eventData, info){
  if(info.eventName == 'SHOULD_SHOW_BUTTON_CONTENT')
{
    // put code here to show/hide the relevant live
chat button HTML within the VA panel as required
based on the button status online/offline/busy etc...
    // check eventData object properties for details
of the status...see Event List in README
  }
  if (info.eventName == 'SHOULD_SHOW_VA_PANEL') {
    // code goes here to show the panel again
  }
  if (info.eventName == 'SHOULD_HIDE_VA_PANEL') {
    // code goes here to HIDE the panel
  }
});
```

- Within the callback function the info.eventName property describes the event in question, allowing the VA Panel code to react and perform various actions such as : SHOW PANEL / HIDE PANEL etc...
- info.eventName object will tell you which event has fired and what you should do next.
- A list of all the events that fire and the data available for each via

eventData is detailed below...

Events List

The following events are of interest to the VA Panel and should be subscribed to as per the above code

- SHOULD_SHOW_VA_PANEL => recommended to show the panel again
- SHOULD_HIDE_VA_PANEL => recommended to hide the panel again
- SHOULD_SHOW_BUTTON_CONTENT => a button has been loaded on the page which is related to the VA Panel. Inspect the eventData object to check the button state (ONLINE/BUSY/OFFLINE) and then show the relevant HTML content within the panel accordingly.
    - eventData.status =
        - "ONLINE"
        - "OFFLINE"
        - "BUSY"
    - eventData.state_enums =
        - UNKNOWN: 0
        - ONLINE: 1
        - OFFLINE: 2
        - BUSY: 4
    - eventData.state_descriptions =
        - 0: "UNKNOWN"
        - 1: "ONLINE"
        - 2: "OFFLINE"
        - 4: "BUSY"
    - eventData.state = 0 | 1 | 2 | 4
    - eventData.reason = description why the event was shown

"How do I start a chat from my custom HTML content?

- Once you have displayed some HTML content for the various button states, you must ensure the ONLINE state includes a click event which calls the following function:

LivePersonVirtualAssistantModule.startChat()

- At this point you have the option of passing in an array [] of strings to represent the conversation history so far between the consumer and the FAQ engine.
- e.g.

```
var messages = ['visitor: I am stuck with password
issues!','FAQ: try this
article...http://www.hsbc.co.uk/password'];
// startChat
LivePersonVirtualAssistantModule.startChat(messages);
 // passes the messages as preChatLines into the chat
window.
```

- **NOTE:** sending messages to .startChat requires the _config.SEND_FAQ_CONVERSATION_AS_PRECHAT_LINES option to be set to true (default)

High Level Process Summary

Website page opens

- already tagged with LivePerson code for monitoring etc (via Tealium)

VA Panel is loaded onto page

- at this call the LivePersonVirtualAssistantModule.init() function to start the process of event bindings and listeners behind the scenese

Visitor Opens VA Panel

- Registering the Chat Button Div Container to load the hidden button
    - call LivePersonVirtualAssistantModule.injectButtonContainer()
        - **Only do this once you know you have shown the panel to the customer and it has been expanded** -- otherwise a button could be loaded off screen which will impact reporting and availability in a negative way
- Embedded Chat Button Events fire and return a status of either ONLINE/BUSY/OFFLINE which is stored by the LivePersonVirtualAssistantModule
- Depending on the button state, CV VA Panel will show/hide custom HTML content which is responsive to the display of the device (desktop/mobile)
    - The ONLINE version should include a call to the LivePersonVirtualAssistantModule.startChat(faqHistorySoFar) function which will begin the chat process if agents are ONLINE
    - e.g.

```
  <p id="button-container">
    <a id="lp-va-panel-button-online" href="#"
class="btn btn-success lp-va-panel-button hide-
lp-button"
onclick="LivePersonVirtualAssistantModule.startC
hat(faqHistorySoFar);">Click to chat</a>
    <a id="lp-va-panel-button-offline" href="#"
class="btn btn-warning lp-va-panel-button hide-
lp-button">All Agents are OFFLINE/BUSY</a>
  </p>
```

Visitor has conversation with CV A.I. FAQ engine

- can see the chat button at the bottom of panel

If Visit Clicks Chat Button...

- call custom function to "fake" click the hidden button to start the chat -- see LivePersonVirtualAssistantModule.startChat(faqHistorySoFar)
- (optional) feed in the faqHistorySoFar by passing in an array of strings representing the current conversation history you wish to recap to the agent and consumer inside the LE2 chat window

- CV VA Panel should Listen for SHOULD_HIDE_VA_PANEL Event to hide VA Panel
- CV VA Panel should Listen for SHOULD_SHOW_VA_PANEL Event to re-show VA Panel

Configuration through _config

- The only thing that could potentially require changes is within the _config object
- The idea is that the LivePersonVirtualAssistantModule is a blackbox that does not need any changes once added to the VA Panel.
- The config should be setup as required and then the exposed methods detailed above give you control/access to what is happening to react accordingly.
- This makes a clear separation of responsibilities from v1 and prevents mixing code for styling/control of the VA Panel within this module.
- It now just informs the listening VA Panel when it should be shown/hidden so that part of the page can react accordingly.

Which options can I change?

- _config.USING_PROXY_BUTTON
    - true : tells the code you will be displaying your own HTML

buttons for the online/offline state of the LivePerson Embedded button which will be loaded.

    - **PLEASE NOTE** This option is recommended for responsive requirements as any images deployed by the LivePerson system will not responsive. We can deploy HTML which you style via CSS if needed.
    - If using this option ensure that whatever HTML elements of your **ONLINE** button state have an onclick function call to startChat which will start the chat window by calling our API and fake clicking the actual button loaded on the page – which will probably be hidden/have no actual viewable HTML content.
    - If you do not wish to handle the custom elements then suggest setting this to false
  - false : presumes the button content will be deployed into the named <div> container and handled by LE2...**NOTE this does NOT support responsive design**

- _config.SEND_FAQ_CONVERSATION_AS_PRECHAT_LINES

  - true (default) : allows array of strings to be accepted from the .startChat function to be passed into the chat window as system messages for visitor and agent
  - **NOTE:** if this is set to false then .startChat(['message1','message2']) would **be ignored and no preChatLines would be passed!**

- _config.TRIGGER_CHAT_BUTTON_FROM_BUSY_STATE

  - false (default) : prevents the module from fake clicking the chat button when the state is BUSY
  - true : would allow the module to fake click the hidden chat button even if BUSY **NOTE:** this could allow a large volume of visitors to join the queue

- _config.TRIGGER_CHAT_BUTTON_FROM_OFFLINE_STATE
    - false (default) : prevents the module from fake clicking the chat button when the state is OFFLINE
    - true : would allow the module to fake click the hidden chat button even if OFFLINE **NOTE:** this will display an OFFLINE survey if enabled on the account for this engagement

How to access/run unit tests?

- Install the dependecies using npm

```
npm install
```

- Run the unit test karma file

```
grunt karma:prod
```

Sample output:

```
Running "karma:prod" (karma) task
15 01 2018 19:18:18.720:INFO [karma]: Karma v0.13.22
server started at http://localhost:9877/
15 01 2018 19:18:18.731:INFO [launcher]: Starting
browser Chrome
15 01 2018 19:18:18.740:INFO [launcher]: Starting
browser Firefox
.....
.....

Firefox 57.0.0 (Mac OS X 10.12.0) LOG: '==>
addSurveyHooks'
Chrome 64.0.3282 (Mac OS X 10.12.6): Executed 9 of 9
```

```
SUCCESS (0.119 secs / 0.032 secs)
Firefox 57.0.0 (Mac OS X 10.12.0): Executed 9 of 9
SUCCESS (0.315 secs / 0.059 secs)
TOTAL: 18 SUCCESS

Done, without errors.
```