Finesse Developer Training with the LearningSampleGadget

While similar, these instructions may not contain the latest LearningSampleGadget file contents. See the files packaged with this documentation (SampleGadget_Final.js, SampleGadget_Final.xml, and SampleGadget.css) for the latest gadget source.

Objectives

- Learn how to add a gadget to the Finesse container
- Learn how to build a gadget using the Finesse JavaScript Library API
    - Setup the client services for the gadget
    - Create a user object
    - Utilize the GET/PUT methods of the user object
    - Add a make call button
    - Utilize the GET methods of the dialog object
    - User dialog data to do a web search

You will start with a blank gadget and use some Finesse JavaScript library APIs to build a basic custom gadget. When you are done, you will have a gadget that can display the user's data, make a call, get call data and use a call variable from the call data to do a bing.com search. These concepts can be extended further to use the other APIs from the Finesse JavaScript library.

Step 1: Identifying the Sample Gadget Files

- The unzipped sample gadget from CDN should have the following files:
    - FinesseJavaScriptLibrary
        - readme.txt: Contains the disclaimer, requirements and usage of the JavaScript library
    - SampleGadget
        - SampleGadget_Final.css: Contains the final styling for the gadget
        - SampleGadget_Final.xml: Contains the final HTML for the gadget
        - SampleGadget_Final.js: Contains the final business logic/javascript for the gadget
        - SampleGadget.css: Contains the styling for the gadget
        - SampleGadget.xml: Contains the minimum HTML to make a gadget
        - SampleGadget.js: Contains the minimum business logic for a gadget
    - _readme.txt: Contains the disclaimer, requirements and usage of the sample gadget

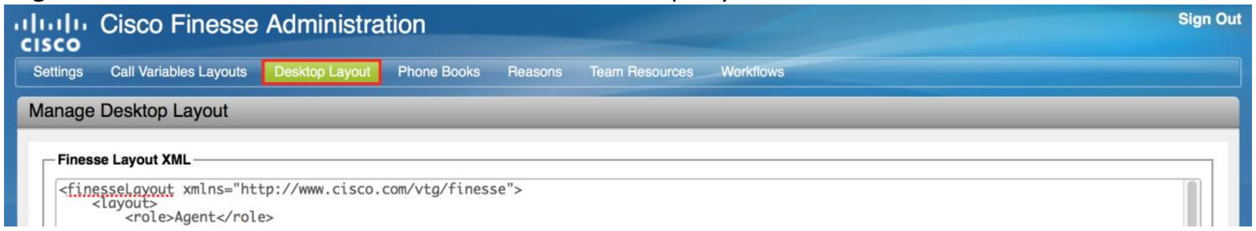
Step 2: Hosting 3rd party gadgets

- 3rd party gadgets (any gadgets that you develop) may be hosted on a separate web server or in the 3rdpartygadget directory on the Finesse server (see chapter 10 of the Finesse WebServices Developer Guide for information on 3rdpartygadget hosting on the Finesse server).
- If you are using your own server as a gadget server, ensure your server is accessible by the Finesse servers.
- Finesse will render the gadgets as defined by the Layout XML
- The sample gadget may be hosted on a separate web server. Place the following files (in the same folder) on your hosting web server:
    - SampleGadget.xml
    - SampleGadget.js
    - SampleGadget.css

Step 3: Adding the gadget to the Finesse Container

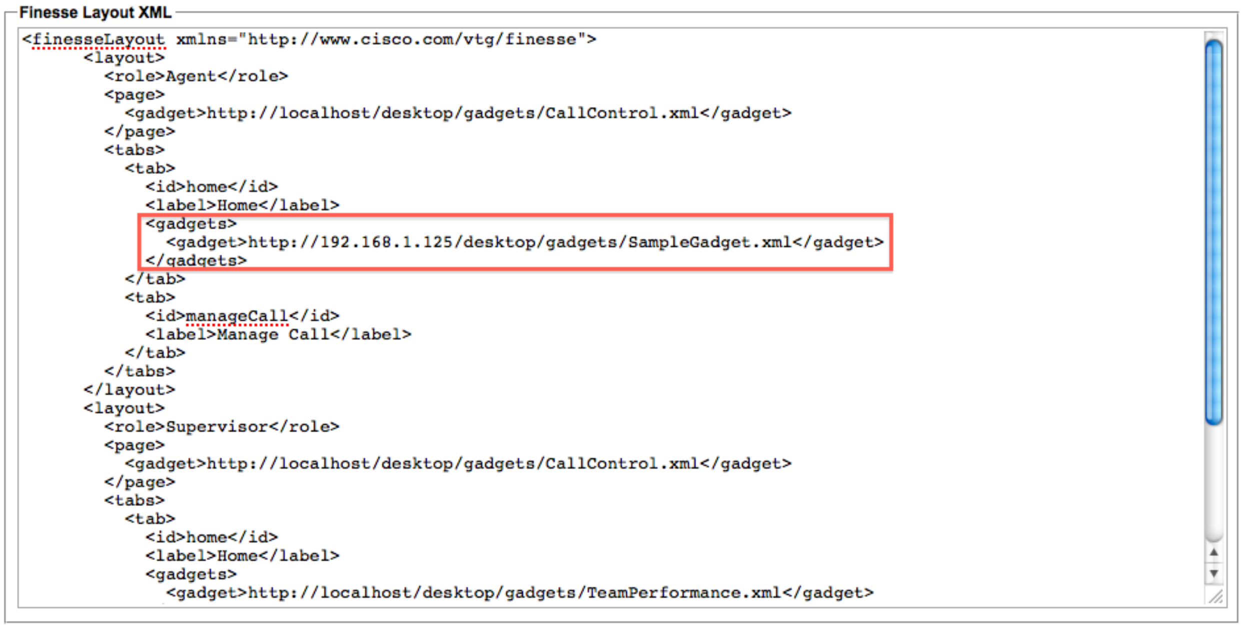
The Finesse Layout XML defines the layout of the Finesse Desktop (both agents and supervisors), including the tab names and the gadgets shown on each tab. Tab names can appear in any language, as long as they are defined in the XML. Use the Manage Desktop Layout gadget to add the gadget to the Finesse Container. This gadget allows the administrator to define/modify the layout of the Finesse Desktop for agents and supervisors.

1. Log into the Finesse Administration and click the Desktop Layout tab.



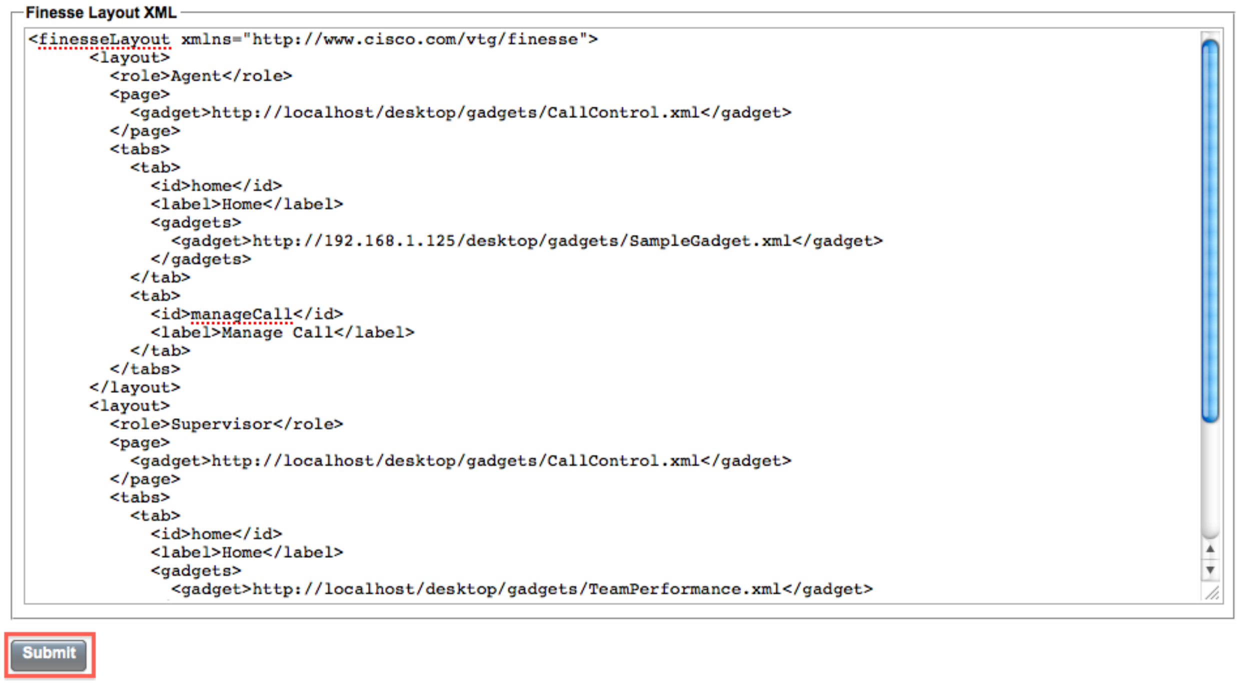
```
<finesseLayout xmlns="http://www.cisco.com/vtg/finesse">
    <layout>
        <role>Agent</role>
```

2. In the Finesse Layout XML, add the URL of the sample gadget (Step 2) to the Agent section of the layout XML



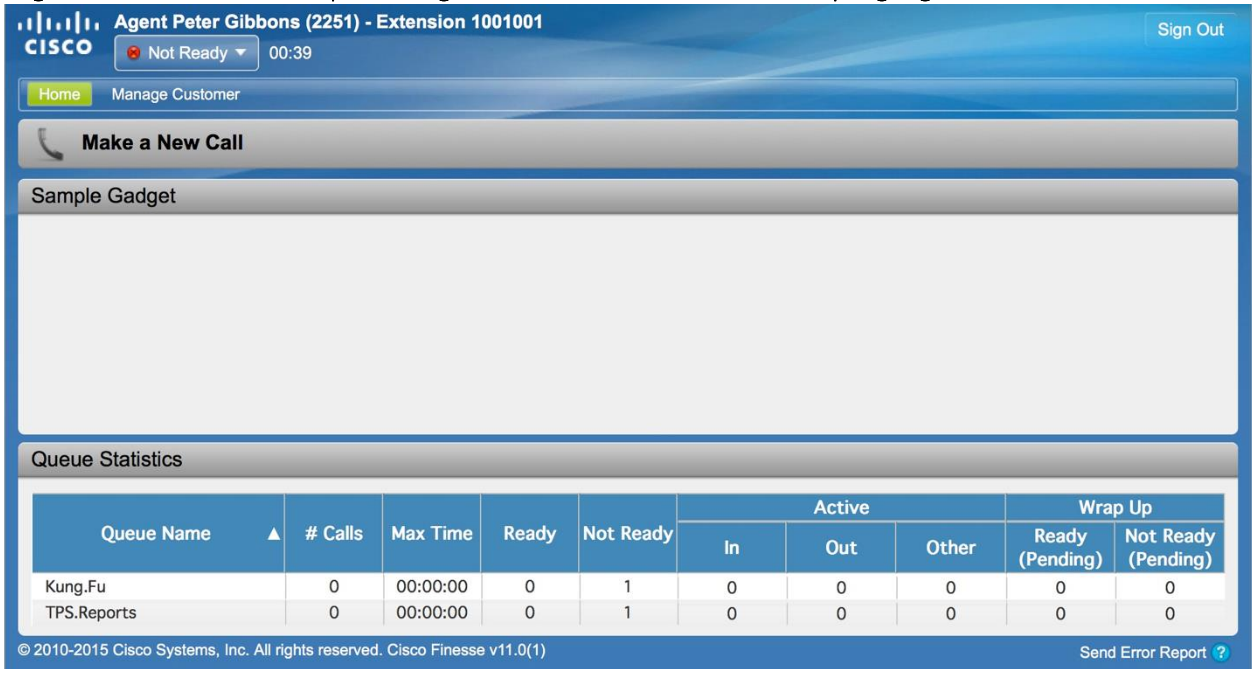
```
<finesseLayout xmlns="http://www.cisco.com/vtg/finesse">
      <layout>
        <role>Agent</role>
        <page>
          <gadget>http://localhost/desktop/gadgets/CallControl.xml</gadget>
        </page>
        <tabs>
          <tab>
            <id>home</id>
            <label>Home</label>
            <gadgets>
              <gadget>http://192.168.1.125/desktop/gadgets/SampleGadget.xml</gadget>
            </gadgets>
          </tab>
          <tab>
            <id>manageCall</id>
            <label>Manage Call</label>
          </tab>
        </tabs>
      </layout>
      <layout>
        <role>Supervisor</role>
        <page>
          <gadget>http://localhost/desktop/gadgets/CallControl.xml</gadget>
        </page>
        <tabs>
          <tab>
            <id>home</id>
            <label>Home</label>
            <gadgets>
              <gadget>http://localhost/desktop/gadgets/TeamPerformance.xml</gadget>
```

3. The Submit button should now be highlighted. Click the Submit button.



```
<finesseLayout xmlns="http://www.cisco.com/vtg/finesse">
      <layout>
        <role>Agent</role>
        <page>
          <gadget>http://localhost/desktop/gadgets/CallControl.xml</gadget>
        </page>
        <tabs>
          <tab>
            <id>home</id>
            <label>Home</label>
            <gadgets>
              <gadget>http://192.168.1.125/desktop/gadgets/SampleGadget.xml</gadget>
            </gadgets>
          </tab>
          <tab>
            <id>manageCall</id>
            <label>Manage Call</label>
          </tab>
        </tabs>
      </layout>
      <layout>
        <role>Supervisor</role>
        <page>
          <gadget>http://localhost/desktop/gadgets/CallControl.xml</gadget>
        </page>
        <tabs>
          <tab>
            <id>home</id>
            <label>Home</label>
            <gadgets>
              <gadget>http://localhost/desktop/gadgets/TeamPerformance.xml</gadget>
```

4. Log into the Finesse Desktop as an Agent. You should see a blank sample gadget.



Agent Peter Gibbons (2251) - Extension 1001001

Not Ready 00:39 Sign Out

Home Manage Customer

Make a New Call

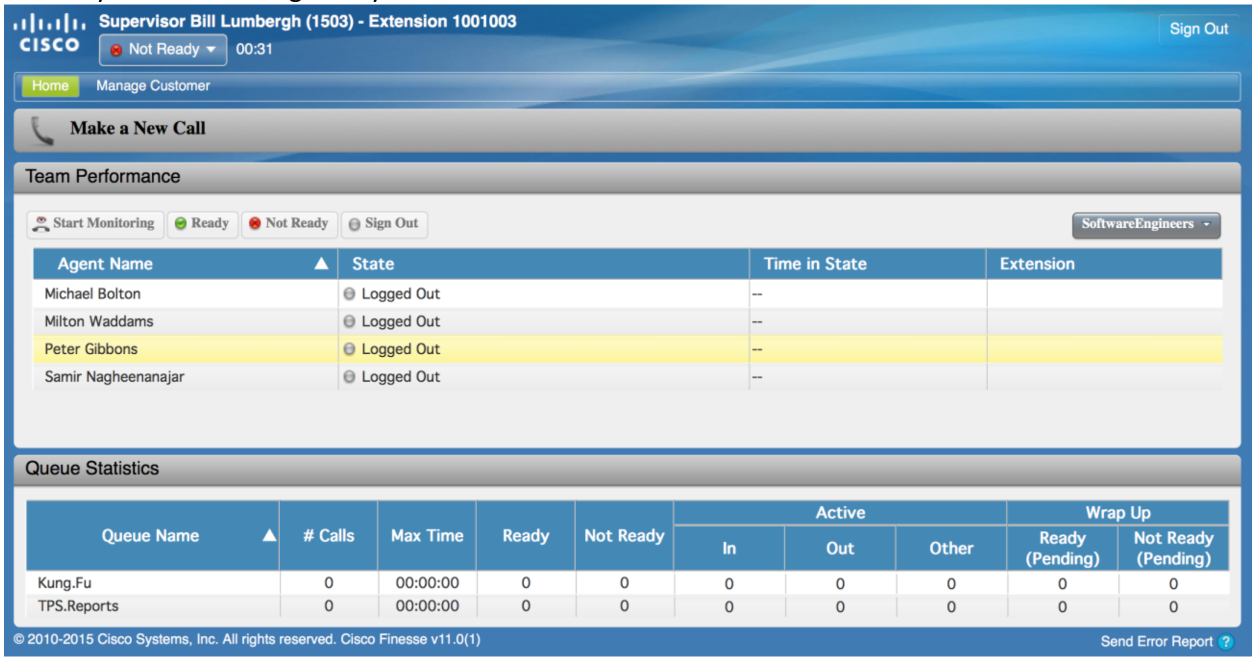
Sample Gadget

Queue Statistics

Queue Name ▲	# Calls	Max Time	Ready	Not Ready	Active			Wrap Up	
					In	Out	Other	Ready (Pending)	Not Ready (Pending)
Kung.Fu	0	00:00:00	0	1	0	0	0	0	0
TPS.Reports	0	00:00:00	0	1	0	0	0	0	0

© 2010-2015 Cisco Systems, Inc. All rights reserved. Cisco Finesse v11.0(1) Send Error Report

5. Log into the Finesse Desktop as a Supervisor. You should NOT see the blank sample gadget because the gadget was only added to the agent layout.



Supervisor Bill Lumbergh (1503) - Extension 1001003

Not Ready 00:31 Sign Out

Home Manage Customer

Make a New Call

Team Performance

Start Monitoring Ready Not Ready Sign Out SoftwareEngineers

Agent Name ▲	State	Time in State	Extension
Michael Bolton	Logged Out	--	
Milton Waddams	Logged Out	--	
Peter Gibbons	Logged Out	--	
Samir Nagheenanajar	Logged Out	--	

Queue Statistics

Queue Name ▲	# Calls	Max Time	Ready	Not Ready	Active			Wrap Up	
					In	Out	Other	Ready (Pending)	Not Ready (Pending)
Kung.Fu	0	00:00:00	0	0	0	0	0	0	0
TPS.Reports	0	00:00:00	0	0	0	0	0	0	0

© 2010-2015 Cisco Systems, Inc. All rights reserved. Cisco Finesse v11.0(1) Send Error Report

Step 4: Get the workspace ready

The rest of this document will explain step-by-step on how to add a user object, a dialog object, and make a call.

- The steps are broken down to be as simple as possible, so it may take many steps to see a usable gadget.
- The instructions will pause at each task and it will require you to re-upload the changed files to the hosted web server (if the files are being modified off of the server) as well as a logout and login of the agent to see the modifications. This step is optional and can be skipped.
- In this demo, the code to be added will be denoted by red font.
- Open up SampleGadget.xml and SampleGadget.js. These are the two files that we will be modifying.
- Note that SampleGadget.xml has the html for the gadget as well as a call to some JavaScript that calls 'init' on the gadget. Init is defined in SampleGadget.js

Step 5: Initialize the ClientServices

The ClientServices is a utility that sets up the BOSH connection and simplifies the construction of Finesse API requests and the consumption of Finesse events.

1. The init function of the ClientServices require a configuration object that contains the properties used for making REST requests. This config object is already in the finesse.js and does not need any modifications.

```
define('gadget/Config',[
    "utilities/Utilities"
], function (Utilities) {
    var Config = (function () { /** @lends finesse.gadget.Config.prototype */
        if (gadgets && gadgets.Prefs) {
            var _prefs = new gadgets.Prefs();
            return {
                authorization: Utilities.getUserAuthString(),
                country: _prefs.getString("country"),
                language: _prefs.getString("language"),
                locale: _prefs.getString("locale"),
                host: _prefs.getString("host"),
                hostPort: _prefs.getString("hostPort"),
                extension: _prefs.getString("extension"),
                mobileAgentMode: _prefs.getString("mobileAgentMode"),
                mobileAgentDialNumber: _prefs.getString("mobileAgentDialNumber"),
                xmppDomain: _prefs.getString("xmppDomain"),
                pubsubDomain: _prefs.getString("pubsubDomain"),
                restHost: _prefs.getString("restHost"),
                scheme: _prefs.getString("scheme"),
                localhostFQDN: _prefs.getString("localhostFQDN"),
                localhostPort: _prefs.getString("localhostPort"),
                teamId: _prefs.getString("teamId"),
                teamName: _prefs.getString("teamName"),
                clientDriftInMillis: _prefs.getInt("clientDriftInMillis"),
                compatibilityMode: _prefs.getString("compatibilityMode"),
                peripheralId: _prefs.getString("peripheralId"),

                /**
                * @class
                * The Config object for gadgets within the Finesse desktop container which
                * contains configuration data provided by the container page.
                * @constructs
                */
                _fakeConstructor : function () {}
            };
        } else {
            return {};
        }
    }());
```

2. In SampleGadget.js, update the init function to initialize the ClientServices:

```
/**
 * Performs all initialization for this gadget
 */
init : function () {
     var cfg = finesse.gadget.Config;

     gadgets.window.adjustHeight();

     // Initiate the ClientServices and load the user object. ClientServices are
     // initialized with a reference to the current configuration.
     finesse.clientservices.ClientServices.init(cfg, false);

     // Initiate the ContainerServices and add a handler for when the tab is visible
     // to adjust the height of this gadget in case the tab was not visible
     // when the html was rendered (adjustHeight only works when tab is visible)
     containerServices = finesse.containerservices.ContainerServices.init();

containerServices.addHandler(finesse.containerservices.ContainerServices.Topics.ACTIVE_TAB,
function() {
         clientLogs.log("Gadget is now visible");  // log to Finesse

         // automatically adjust the height of the gadget to show the html
         gadgets.window.adjustHeight();
     });

     containerServices.makeActiveTabReq();
}
```

3. In SampleGadget.xml, update the onConnect function call to call init function from SampleGadget.js:

```
<script type="text/javascript">
    gadgets.HubSettings.onConnect = function () {
        finesse.modules.SampleGadget.init();
    };
</script>
```

Step 6: Create User object

The user object represents the user that is currently logged in. Function calls on this object only operate against this user.

1. In order to get the data from the User object, you must create an instance of the User. In SampleGadget.js, update the init function to add an instance of the User object:

```
finesse.modules.SampleGadget = (function ($) {
    var user;

    /** @scope finesse.modules.SampleGadget */
    return {

        /**
         * Performs all initialization for this gadget
         */
        init : function () {
            var cfg = finesse.gadget.Config;

            gadgets.window.adjustHeight();

            // Initiate the ClientServices and load the user object. ClientServices are
            // initialized with a reference to the current configuration.
            finesse.clientservices.ClientServices.init(cfg, false);

            user = new finesse.restservices.User({
                id: cfg.id,
                onLoad : handleUserLoad,
                onChange : handleUserChange
            });

            // Initiate the ContainerServices and add a handler for when the tab is visible
            // to adjust the height of this gadget in case the tab was not visible
            // when the html was rendered (adjustHeight only works when tab is visible)
            containerServices = finesse.containerservices.ContainerServices.init();

containerServices.addHandler(finesse.containerservices.ContainerServices.Topics.ACTIVE_TAB,
function() {
                clientLogs.log("Gadget is now visible");  // log to Finesse

                // automatically adjust the height of the gadget to show the html
                gadgets.window.adjustHeight();
            });

            containerServices.makeActiveTabReq();
        }
    };
}(jQuery));
```

2. In SampleGadget.js, create the callback functions handleUserLoad and handleUserChange for the new User instance was added in Step 6.1.

```
finesse.modules.SampleGadget = (function ($) {
    var user,

    /**
     * Handler for the onLoad of a User object. This occurs when the User object is initially read
     * from the Finesse server. Any once only initialization should be done within this function.
     */
    handleUserLoad = function (userevent) { },

    /**
     *  Handler for all User updates
     */
    handleUserChange = function(userevent) { };


    /** @scope finesse.modules.SampleGadget */
    return {

        /**
         * Performs all initialization for this gadget
         */
        init : function () {
            ...
        }
    };
}(jQuery));
```

Step 7: Add HTML DOM for the User GET

This sample gadget will display the the user's id, first name, last name, role, extension, and its current state. In this step, we are going to set up the gadget to have the ability to display the data. It will not populate the fields with user data.

1. In SampleGadget.xml, add the following within the <body> tag:

```
<body class="claro">
    <!-- Sample Gadget -->
    <div>
        <fieldset id="userfieldset" class="outline">
            <legend>User</legend>
            <div><b> User ID: </b><span id="userId"></span></div>
            <div><b> First Name: </b><span id="firstName"></div>
            <div><b> Last Name: </b><span id="lastName"></div>
            <div><b> Role: </b><span id="userRole"></div>
            <div><b> Extension: </b><span id="extension"></div>
            <div><b> Current User State: </b><span id="userState"></div>
        </fieldset>
    </div>
</body>
```

Step 8: Display User data using a GET

This step will populate the fields from step 7 with user data from the User object (GET)

1. In SampleGadget.js, update the handleUserLoad and handleUserChange callback functions from Step 6.2

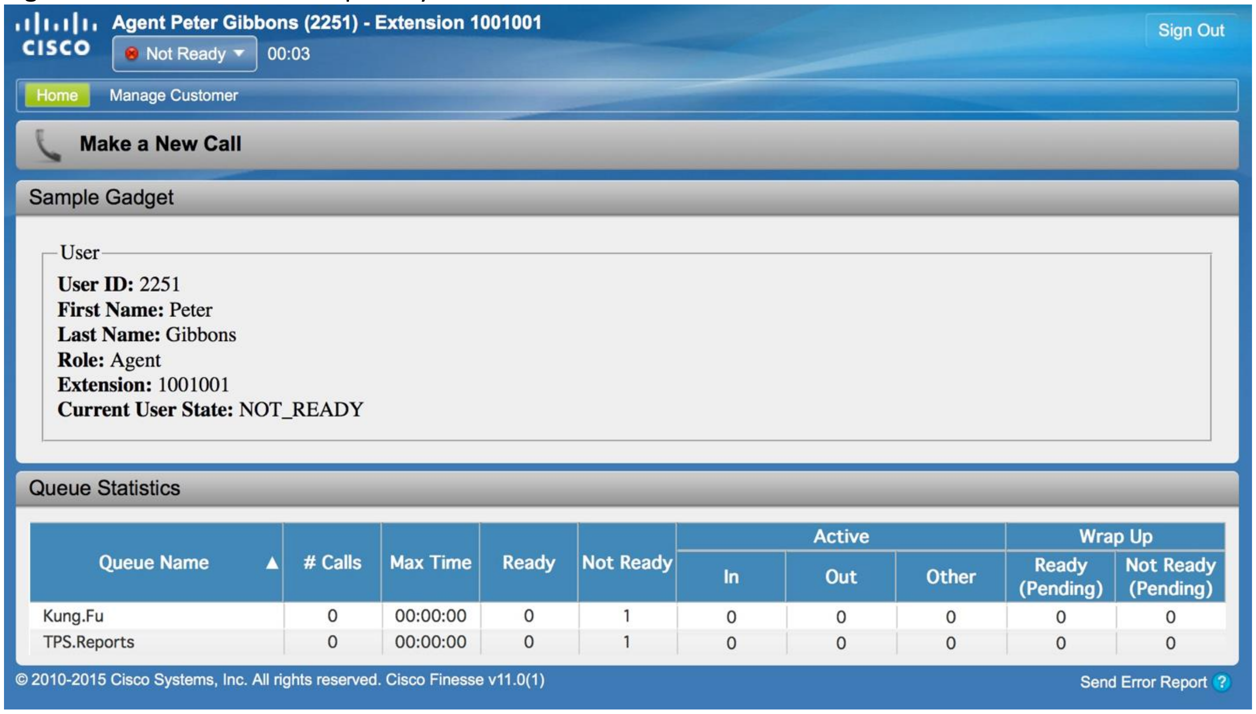
```
finesse.modules.SampleGadget = (function ($) {
    var user,

    /**
     * Populates the fields in the gadget with data
     */
    render = function () {
        var currentState = user.getState();

        // Examples of getting data from the User object (GET)
        $("#userId").text(user.getId());
        $("#firstName").text(user.getFirstName());
        $("#lastName").text(user.getLastName());
        if (user.hasSupervisorRole()) {
            $("#userRole").text('Supervisor');
        } else {
            $("#userRole").text('Agent');
        }
        $("#extension").text(user.getExtension());
        $("#userState").text(currentState);

        gadgets.window.adjustHeight();
    },

    /**
     * Handler for the onLoad of a User object. This occurs when the User object is initially read
     * from the Finesse server. Any once only initialization should be done within this function.
     */
    handleUserLoad = function (userevent) {
        render();
    },

    /**
     *  Handler for all User updates
     */
    handleUserChange = function(userevent) {
        render();
    };

    /** @scope finesse.modules.SampleGadget */
    return {

        /**
         * Performs all initialization for this gadget
         */
        init : function () {
            ...
        }
    };
}(jQuery));
```

Step 9: Upload Gadget with User GET

Uploading the gadget to the hosted web server is optional, but doing so allows you to see your progress.

1. If you haven't already done so, sign out the agent from the Finesse Desktop.
2. Upload the SampleGadget.xml and SampleGadget.js files to the hosted web server (Step 2).

3. Sign in to the Finesse Desktop and you should see:



Sample Gadget

User

User ID: 2251
First Name: Peter
Last Name: Gibbons
Role: Agent
Extension: 1001001
Current User State: NOT_READY

Queue Statistics

Queue Name ▲	# Calls	Max Time	Ready	Not Ready	Active			Wrap Up	
					In	Out	Other	Ready (Pending)	Not Ready (Pending)
Kung.Fu	0	00:00:00	0	1	0	0	0	0	0
TPS.Reports	0	00:00:00	0	1	0	0	0	0	0

© 2010-2015 Cisco Systems, Inc. All rights reserved. Cisco Finesse v11.0(1)

Send Error Report

Step 10: Add the User States

Users can change their own state. The possible states that the user can change to depends on the user's current state. In order to maintain consistency amongst the code, the Finesse JavaScript API provides a user states enum that represents all possible user states. This step is preparing the code for changing the user's state.

1. In SampleGadget.js, update the **init** function to add the user states:

```
finesse.modules.SampleGadget = (function ($) {
    var user, states,

    ...

    /** @scope finesse.modules.SampleGadget */
    return {

        /**
         * Performs all initialization for this gadget
         */
        init : function () {
            var cfg = finesse.gadget.Config;

            gadgets.window.adjustHeight();

            // Initiate the ClientServices and load the user object. ClientServices are
            // initialized with a reference to the current configuration.
            finesse.clientservices.ClientServices.init(cfg, false);
            user = new finesse.restservices.User({
                id: cfg.id,
                onLoad : handleUserLoad,
                onChange : handleUserChange
            });

            states = finesse.restservices.User.States;

            // Initiate the ContainerServices and add a handler for when the tab is visible
            // to adjust the height of this gadget in case the tab was not visible
            // when the html was rendered (adjustHeight only works when tab is visible)
            containerServices = finesse.containerservices.ContainerServices.init();
containerServices.addHandler(finesse.containerservices.ContainerServices.Topics.ACTIVE_TAB,
function() {
                clientLogs.log("Gadget is now visible");  // log to Finesse

                // automatically adjust the height of the gadget to show the html
                gadgets.window.adjustHeight();
            });

            containerServices.makeActiveTabReq();
        }
    };
}(jQuery));
```

Step 11: Add HTML DOM for the User PUT

Now, you will update the sample gadget to have two buttons. The first button changes the user to READY and the second button changes the user to NOT_READY. This step adds the visual piece of the User PUT, but the buttons will not function correctly.

1. In SampleGadget.xml, add the following code to the userfieldset:

```
<!-- sample gadget -->
<div>
    <fieldset id="userfieldset" class="outline">
        ...
        <div><b> Current User State: </b><span id="userState"></div>
        <br>
        <div id="goReady">
            <button onClick="finesse.modules.SampleGadget.setUserState('READY');">Change
state to READY</button>
        </div>
        <div id="goNotReady">
            <button onClick="finesse.modules.SampleGadget.setUserState('NOT_READY');">Change
state to NOT READY</button>
        </div>
    </fieldset>
</div>
```

Step 12: Add business logic for User PUT

In this step, you are going to hook in the support for the button that was added in Step 11.

1. In SampleGadget.js, add a **setUserState** function (matching the onClick value from Step 11) to the return clause. This function calls the user PUT method with the appropriate parameter.

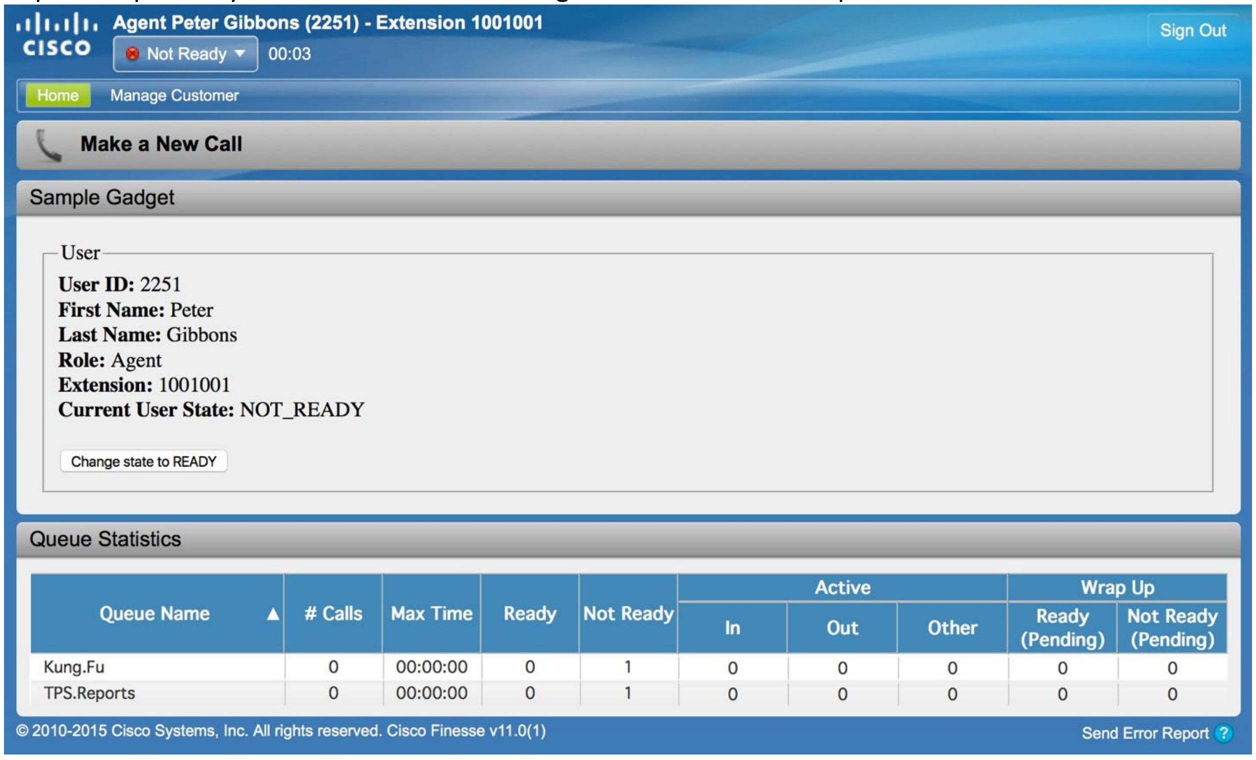
```
return {
    /**
     * Sets the user state
     */
    setUserState : function (state) {
        if (state === 'READY') {
            user.setState(states.READY);
        } else if (state === 'NOT_READY') {
            user.setState(states.NOT_READY);
        }
    },

    /**
     * Performs all initialization for this gadget
     */
    init : function () {
        ...
    }
};
```

2. In SampleGadget.js, update the render function to hide and display the correct button according to the user's current state.

```
/**
 * Populates the fields in the gadget with data
 */
render = function () {
    var currentState = user.getState();

    // Examples of getting data from the User object (GET)
    $("#userId").text(user.getId());
    $("#firstName").text(user.getFirstName());
    $("#lastName").text(user.getLastName());
    if (user.hasSupervisorRole()) {
        $("#userRole").text('Supervisor');
    } else {
        $("#userRole").text('Agent');
    }
    $("#extension").text(user.getExtension());
    $("#userState").text(currentState);

    // Example of setting the user state (PUT)
    if (currentState === states.NOT_READY) {
        $("#goReady").show();
        $("#goNotReady").hide();
    } else if (currentState === states.READY) {
        $("#goNotReady").show();
        $("#goReady").hide();
    } else {
        $("#goNotReady").hide();
        $("#goReady").hide();
    }

    gadgets.window.adjustHeight();
},
```

Step 13: Upload Gadget with User PUT

1. Repeat Step 9 and you should see the following on the Finesse Desktop:



Agent Peter Gibbons (2251) - Extension 1001001

00:03

Sample Gadget

User

User ID: 2251
First Name: Peter
Last Name: Gibbons
Role: Agent
Extension: 1001001
Current User State: NOT_READY

Queue Statistics

Queue Name ▲	# Calls	Max Time	Ready	Not Ready	Active			Wrap Up	
					In	Out	Other	Ready (Pending)	Not Ready (Pending)
Kung.Fu	0	00:00:00	0	1	0	0	0	0	0
TPS.Reports	0	00:00:00	0	1	0	0	0	0	0

© 2010-2015 Cisco Systems, Inc. All rights reserved. Cisco Finesse v11.0(1)

Step 14: Add HTML DOM for the make call

Now, you will update the sample gadget to have a button to make a call to an extension. This step adds the button for the make call, but the button will not function correctly.

1. In SampleGadget.xml, add the following code to the userfieldset. Change the value of the extension to a valid extension in your setup.

```
<!-- sample gadget -->
<div>
    <fieldset id="userfieldset" class="outline">
        ...
        <div id="goNotReady">
            <button onClick="finesse.modules.SampleGadget.setUserState('NOT_READY');">Change 
state to NOT READY</button>
        </div>
        <br>
        <div id="makeCallButton">
            <input type="text" id="phoneId" value="6100"></input>
            <button onClick="finesse.modules.SampleGadget.makeCall(($('#phoneId')).val());">Make 
Call</button>
        </div>
    </fieldset>
</div>
```

Step 15: Add business logic for the make call

In this step, you are going to hook in the support for the make call button that was added in Step 14.

1. In SampleGadget.js, add a **makeCall** function (matching the onClick value from Step 14) to the return clause. This function calls the user PUT method with the appropriate parameter.

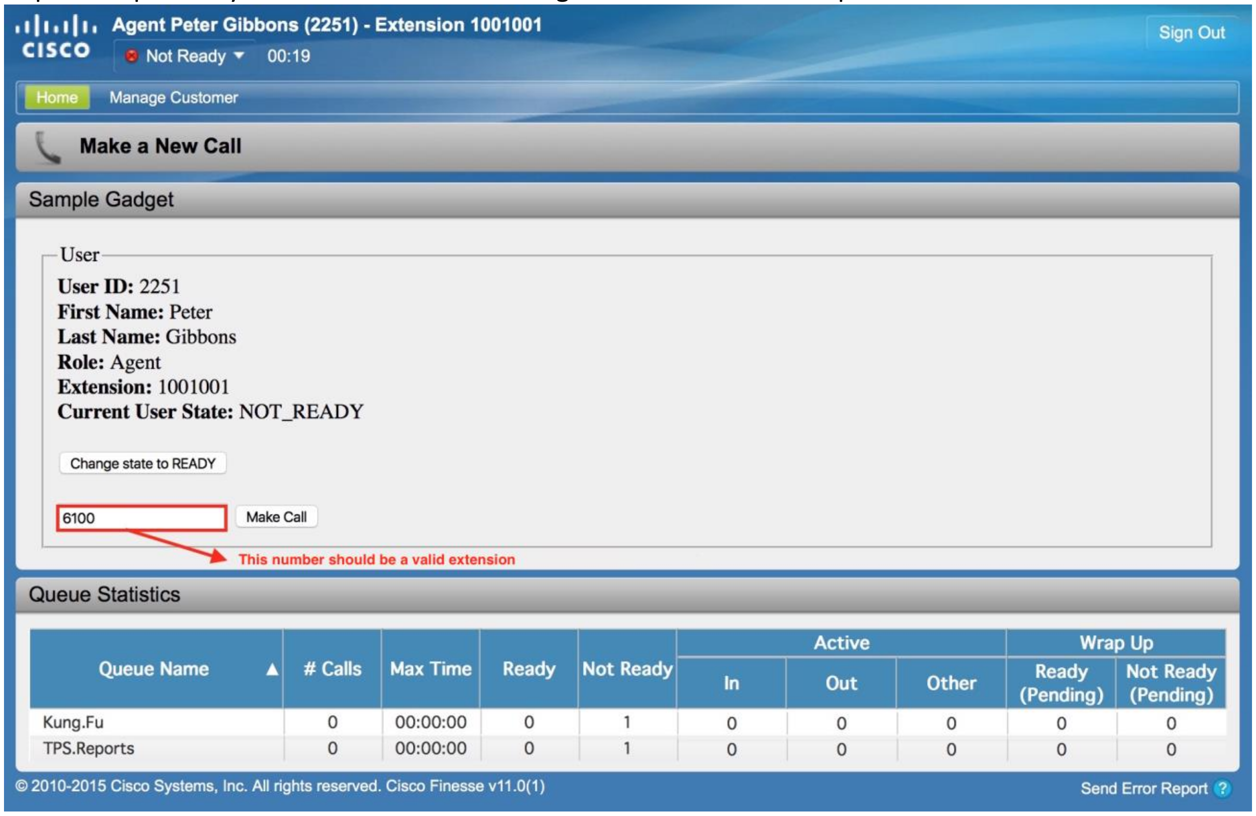
```
return {
    /**
     * Sets the user state
     */
    setUserState : function (state) {
        if (state === 'READY') {
            user.setState(states.READY);
        } else if (state === 'NOT_READY') {
            user.setState(states.NOT_READY);
        }
    },

    /**
     * Make a call to the number
     */
    makeCall : function (number) {
        // Example of the user make call method
        user.makeCall(number, {
            success: makeCallSuccess,
            error: makeCallError
        });

        // Hide the button after making the call
        $("#makeCallButton").hide();
    },

    /**
     * Performs all initialization for this gadget
     */
    init : function () {
        ...
    }
};
```

2. In SampleGadget.js, create empty callback functions **makeCallSuccess** and **makeCallError** to match the success and error of the makeCall function from Step 15.1.

```
finesse.modules.SampleGadget = (function ($) {
    ...

    /**
     * Handler for makeCall when successful.
     */
    makeCallSuccess = function(rsp) { },

    /**
     * Handler for makeCall when error occurs.
     */
    makeCallError = function(rsp) { },

    /**
     * Handler for the onLoad of a User object. This occurs when the User object is initially read
     * from the Finesse server. Any once only initialization should be done within this function.
     */
    handleUserLoad = function (userevent) {
        render();
    },

    ...

}(jQuery));
```

Step 16: Upload Gadget with make call

1. Repeat Step 9 and you should see the following on the Finesse Desktop:



User ID: 2251
First Name: Peter
Last Name: Gibbons
Role: Agent
Extension: 1001001
Current User State: NOT_READY

Queue Name	# Calls	Max Time	Ready	Not Ready	Active In	Active Out	Active Other	Wrap Up Ready (Pending)	Wrap Up Not Ready (Pending)
Kung.Fu	0	00:00:00	0	1	0	0	0	0	0
TPS.Reports	0	00:00:00	0	1	0	0	0	0	0

© 2010-2015 Cisco Systems, Inc. All rights reserved. Cisco Finesse v11.0(1)

Step 17: Add the dialog object

A dialog object represents a conversation between two or more participants. For the media type "voice", this object represents a call. A participant represents an internal or external user's CallConnection, or that user's leg of the call.

1. When the user is loaded, it needs to call the GET dialogs API to fetch the list of dialogs the user belongs to. In SampleGadget.js, update the **handleUserLoad** function to get the list of dialogs for the user. This function calls the user GET method.

```
finesse.modules.SampleGadget = (function ($) {
    var user, states, dialogs,

    ...

    /**
     * Handler for the onLoad of a User object. This occurs when the User object is initially read
     * from the Finesse server. Any once only initialization should be done within this function.
     */
    handleUserLoad = function (userevent) {
        // Get an instance of the dialogs collection and register handlers for dialog additions and
        // removals
        dialogs = user.getDialogs( {
            onCollectionAdd : handleNewDialog,
            onCollectionDelete : handleEndDialog
        });

        render();
    },

    ...

}(jQuery));
```

2. In SampleGadget.js, create empty callback functions **handleNewDialog** and **handleEndDialog** to match the onCollectionAdd and onCollectionDelete of the getDialogs function from Step 17.1.

```
finesse.modules.SampleGadget = (function ($) {

    ...

    /**
     *  Handler for additions to the Dialogs collection object. This will occur when a new
     *  Dialog is created on the Finesse server for this user.
     */
    handleNewDialog = function(dialog) { },

    /**
     *  Handler for deletions from the Dialogs collection object for this user. This will occur
     *  when a Dialog is removed from this user's collection (example, end call)
     */
    handleEndDialog = function(dialog) { },

    /**
     * Handler for makeCall when successful.
     */
    makeCallSuccess = function(rsp) { },

    ...

}(jQuery));
```

Step 18: Add HTML DOM for the Dialog GET

Now, update the sample gadget to display the user's current dialog's call Id, DNIS, from address, to address, and call state, which is taken from the dialog event. In this step, you are going to set up the gadget to have the ability to display the data. It will not populate the fields with user data.

1. In SampleGadget.xml, add the following HTML to the **<div>** containing the userfieldset fieldset.

```
<!-- sample gadget -->
<div>
    <fieldset id="userfieldset" class="outline">

        ...

    </fieldset>
    <br>

    <fieldset id="dialogfieldset" class="outline">
        <legend>Dialog</legend>
        <div><b> Call Id: </b><span id="callId"></div>
        <div><b> Call Type: </b><span id="callType"></div>
        <div><b> DNIS: </b><span id="dnis"></div>
        <div><b> From Address: </b><span id="fromAddress"></div>
        <div><b> To Address: </b><span id="toAddress"></div>
        <div><b> Call State: </b><span id="callState"></div>
    </fieldset>
</div>
```

Step 19: Display Dialog Data using GET

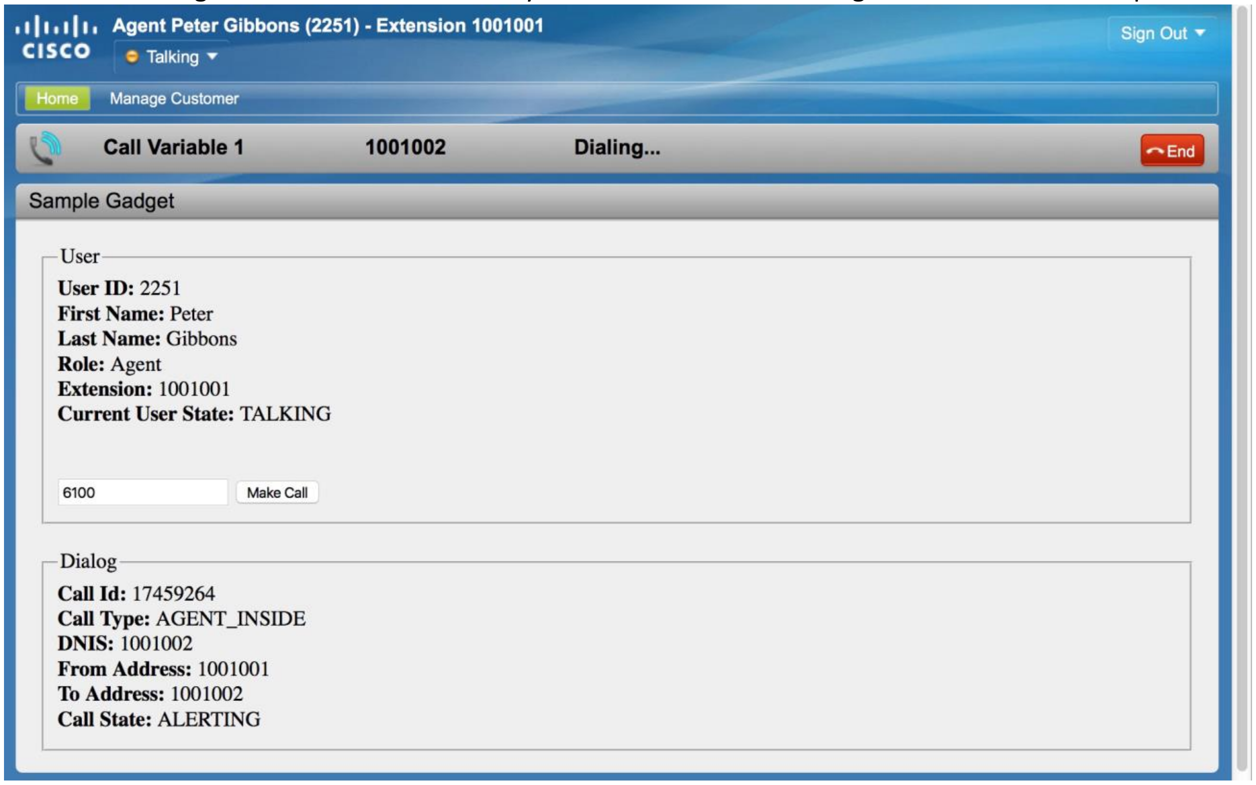
This step will populate the fields from Step 18 with dialog data.

1. In SampleGadget.js, define the callback functions **handleNewDialog** and **handleEndDialog** from Step 17.2.

```
finesse.modules.SampleGadget = (function ($) {

    ...

    displayCall = function (dialog) {
        var callVars = dialog.getMediaProperties();

        // Examples of getting data from the Dialog object (GET)
        $("#dnis").text(dialog.getMediaProperties().DNIS);
        $("#callType").text(dialog.getMediaProperties().callType);
        $("#fromAddress").text(dialog.getFromAddress());
        $("#toAddress").text(dialog.getToAddress());
        $("#callState").text(dialog.getState());
    },

    _processCall = function (dialog) {
        displayCall(dialog);
    },

    /**
     *  Handler for additions to the Dialogs collection object. This will occur when a new
     *  Dialog is created on the Finesse server for this user.
     */
    handleNewDialog = function(dialog) {
        // call the displayCall handler
        displayCall(dialog);

        // add a dialog change handler in case the callvars didn't arrive yet
        dialog.addHandler('change', _processCall);
    },

    /**
     *  Handler for deletions from the Dialogs collection object for this user. This will occur
     *  when a Dialog is removed from this user's collection (example, end call)
     */
    handleEndDialog = function(dialog) {
        // Clear the fields when the call is ended
        $("#callId").text("");
        $("#dnis").text("");
        $("#callType").text("");
        $("#fromAddress").text("");
        $("#toAddress").text("");
        $("#callState").text("");
    },

    ...

}(jQuery));
```

Step 20: Upload Gadget with Dialog GET

1. Repeat Step 9.
2. Make a call using the Make Call button and you should see the following on the Finesse Desktop:



User
User ID: 2251
First Name: Peter
Last Name: Gibbons
Role: Agent
Extension: 1001001
Current User State: TALKING

Dialog
Call Id: 17459264
Call Type: AGENT_INSIDE
DNIS: 1001002
From Address: 1001001
To Address: 1001002
Call State: ALERTING

Step 21: Hide the make call button when user is on a call

When the user is on a call, they cannot make another new call (with an exception of a consult call). This step hides the make call button (from Step 14 and 15) when the user is on a call.

1. In SampleGadget.js, update the **handleNewDialog** and **handleEndDialog** callback functions to hide and show the make call button.

```
displayCall = function (dialog) {
    var callVars = dialog.getMediaProperties();

     // Examples of getting data from the Dialog object (GET)
     $("#callId").text(dialog.getId());
     $("#dnis").text(dialog.getMediaProperties().DNIS);
     $("#callType").text(dialog.getMediaProperties().callType);
     $("#fromAddress").text(dialog.getFromAddress());
     $("#toAddress").text(dialog.getToAddress());
     $("#callState").text(dialog.getState());

     // Hide the make call button when the user is on a call
     $("#makeCallButton").hide();
},

...

/**
 *  Handler for deletions from the Dialogs collection object for this user.  This will occur
 *  when a Dialog is removed from this user's collection (example, end call)
 */
handleEndDialog = function(dialog) {
    // Clear the fields when the call is ended
    $("#callId").text("");
    $("#dnis").text("");
    $("#callType").text("");
    $("#fromAddress").text("");
    $("#toAddress").text("");
    $("#callState").text("");

    // Show the make call button when the call is ended
    $("#makeCallButton").show();
},
```

Step 22: Use Dialog Data to do a search

Using the available data (GET), things like web searches can be easily done. In this example, this gadget will be updated to take the value from call variable 3 and do a bing.com search. **Note**: Call variable 3 must be configured to a value or else ‘undefined’ will be the search text.

1. Add an iframe to the gadget with the main bing.com site. In SampleGadget.xml, update the **dialogfieldset** to include an iframe bing.com as the source. **Note**: Starting with Firefox 23, it does not allow mixed content. So if you are accessing the Finesse Desktop with http, use http://www.bing.com. If you are accessing the Finesse Desktop with https, use https://www.bing.com.

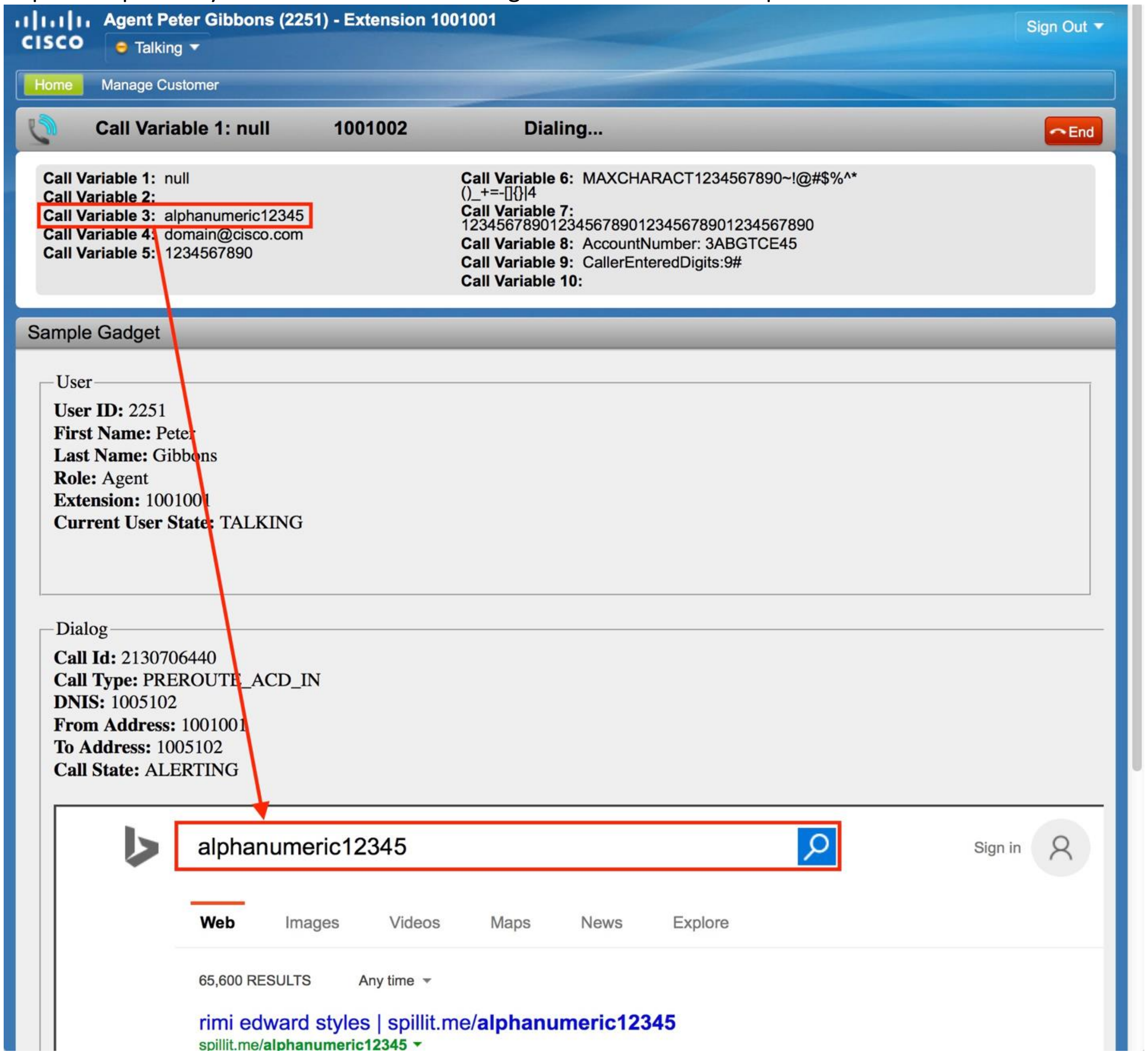
```
<body class="claro">
    ...
        <fieldset id="dialogfieldset" class="outline">
            <legend>Dialog</legend>
            <div><b> Call Id: </b><span id="callId"></div>
            <div><b> Call Type: </b><span id="callType"></div>
            <div><b> DNIS: </b><span id="dnis"></div>
            <div><b> From Address: </b><span id="fromAddress"></div>
            <div><b> To Address: </b><span id="toAddress"></div>
            <div><b> Call State: </b><span id="callState"></div>
            <br>
            <iframe name="bing" id="bing" width="900" height="300" src=""></iframe>
        </fieldset>
    </div>
</body>

<script type="text/javascript">
    gadgets.HubSettings.onConnect = function () {
        finesse.modules.SampleGadget.init();
        $("#bing").attr("src","https://www.bing.com");
    };
</script>
```

2. In SampleGadget.js, update the dialog object's callbacks **handleNewDialog** and **handleEndDialog** to change the URL of the iframe accordingly. When a call arrives, update the URL to include call variable 3's value as the search value. When the call ends, change the URL to go back to the main bing.com page.

```
displayCall = function (dialog) {
    var callVars = dialog.getMediaProperties();

    // Examples of getting data from the Dialog object (GET)
    $("#dnis").text(dialog.getMediaProperties().DNIS);
    $("#callType").text(dialog.getMediaProperties().callType);
    $("#fromAddress").text(dialog.getFromAddress());
    $("#toAddress").text(dialog.getToAddress());
    $("#callState").text(dialog.getState());

    // Hide the make call button when the user is on a call
    $("#makeCallButton").hide();

    // Example of using data from the dialog to do a web search
    $("#bing").attr("src","https://www.bing.com/search?q=" + callVars["callVariable3"]);
},

...

/**
 *  Handler for deletions from the Dialogs collection object for this user. This will
occur
 *  when a Dialog is removed from this user's collection (example, end call)
 */
handleEndDialog = function(dialog) {
    // Clear the fields when the call is ended
    $("#callId").text("");
    $("#dnis").text("");
    $("#callType").text("");
    $("#fromAddress").text("");
    $("#toAddress").text("");
    $("#callState").text("");

    // Show the make call button when the call is ended
    $("#makeCallButton").show();

    // Remove the dialog data from the web search
    $("#bing").attr("src","https://www.bing.com");
},
```

Step 23: Upload Gadget with Dialog GET

1. Repeat Step 9 and you should see the following on the Finesse Desktop:



Agent Peter Gibbons (2251) - Extension 1001001

Talking

Home | Manage Customer

Call Variable 1: null | 1001002 | Dialing...

Call Variable 1: null
Call Variable 2:
Call Variable 3: alphanumeric12345
Call Variable 4: domain@cisco.com
Call Variable 5: 1234567890
Call Variable 6: MAXCHARACT1234567890~!@#$%^*()_+=-[]{}|4
Call Variable 7: 1234567890123456789012345678901234567890
Call Variable 8: AccountNumber: 3ABGTCE45
Call Variable 9: CallerEnteredDigits:9#
Call Variable 10:

Sample Gadget

User

User ID: 2251
First Name: Peter
Last Name: Gibbons
Role: Agent
Extension: 1001001
Current User State: TALKING

Dialog

Call Id: 2130706440
Call Type: PREROUTE_ACD_IN
DNIS: 1005102
From Address: 1001001
To Address: 1005102
Call State: ALERTING

alphanumeric12345

65,600 RESULTS

rimi edward styles | spillit.me/alphanumeric12345
spillit.me/alphanumeric12345

NOTE: You need to have something in Callvariable3 to search on. You should have an ICM or UCCX script that assigns a value to Callvariable3 in order to use this example.

Step 24: Adding gadget logging

Proper client logging is important for maintainability as well as providing technical support. Without client logging, it is impossible to debug problems.

1. Initialize the ClientLogger. The ClientLogger allows gadgets to call the log function to publish client logging messages over the hub. In SampleGadget.js, initialize the ClientLogger.

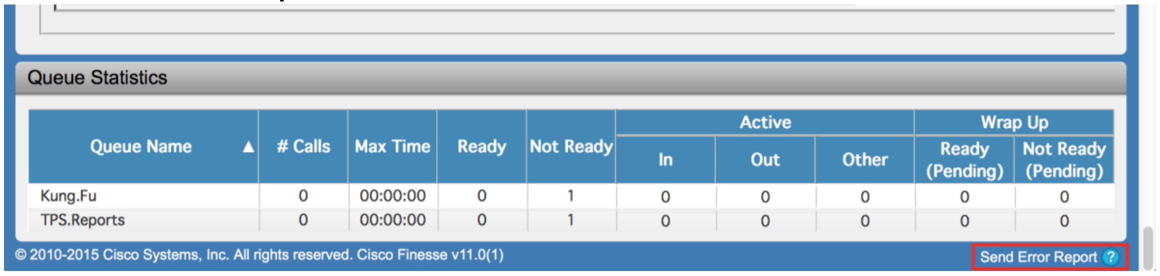
```
finesse.modules.SampleGadget = (function ($) {
    var user, states, dialogs, clientlogs,

    ...

    /** @scope finesse.modules.SampleGadget */
    return {

        ...

        /**
         * Performs all initialization for this gadget
         */
        init : function () {
            var cfg = finesse.gadget.Config;

            clientLogs = finesse.cslogger.ClientLogger;  // declare clientLogs

            gadgets.window.adjustHeight();

            // Initiate the ClientServices and load the user object. ClientServices are
            // initialized with a reference to the current configuration.
            finesse.clientservices.ClientServices.init(cfg, false);

            // Initiate the ClientLogs. The gadget id will be logged as a part of the message
            clientLogs.init(gadgets.Hub, "SampleGadget");

            user = new finesse.restservices.User({
                id: cfg.id,
                onLoad : handleUserLoad,
                onChange : handleUserChange
            });

            states = finesse.restservices.User.States;

            // Initiate the ContainerServices and add a handler for when the tab is visible
            // to adjust the height of this gadget in case the tab was not visible
            // when the html was rendered (adjustHeight only works when tab is visible)
            containerServices = finesse.containerservices.ContainerServices.init();
containerServices.addHandler(finesse.containerservices.ContainerServices.Topics.ACTIVE_TAB,
function() {
                clientLogs.log("Gadget is now visible");  // log to Finesse

                // automatically adjust the height of the gadget to show the html
                gadgets.window.adjustHeight();
            });

            containerServices.makeActiveTabReq();
        }
    };
}(jQuery));
```

2. In SampleGadget.js, add client logging where necessary:

```
setUserState : function (state) {
    clientLogs.log("setUserState(): The user's current state is: " + state);
    if (state === 'READY') {
        user.setState(states.READY);
    } else if (state === 'NOT_READY') {
        user.setState(states.NOT_READY);
    }
},

/**
 * Make a call to the number
 */
makeCall : function (number) {
    clientLogs.log("makeCall(): Making a call to " + number);
    // Example of the user make call method
    user.makeCall(number, {
        success: makeCallSuccess,
        error: makeCallError
    });

    // Hide the button after making the call
    $("#makeCallButton").hide();
},
```

Step 25: Upload Gadget with gadget logging

1. Repeat Step 9.
2. Execute the functions where gadget logging was added.
3. Click the **Send Error Report** link.

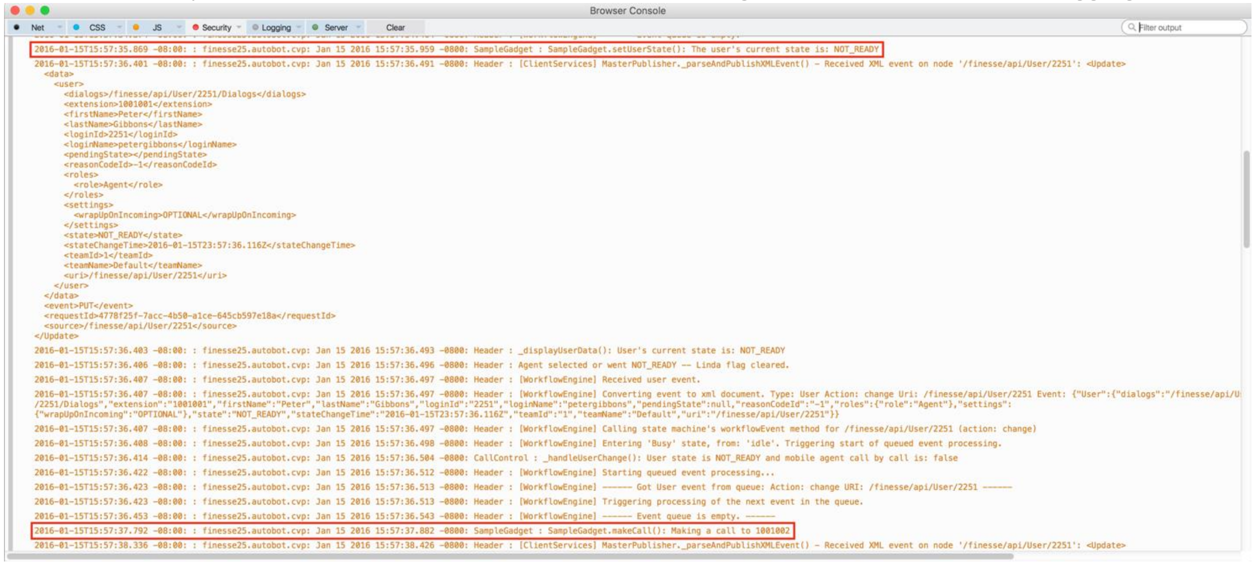


Queue Statistics

Queue Name ▲	# Calls	Max Time	Ready	Not Ready	Active			Wrap Up	
					In	Out	Other	Ready (Pending)	Not Ready (Pending)
Kung.Fu	0	00:00:00	0	1	0	0	0	0	0
TPS.Reports	0	00:00:00	0	1	0	0	0	0	0

© 2010-2015 Cisco Systems, Inc. All rights reserved. Cisco Finesse v11.0(1)

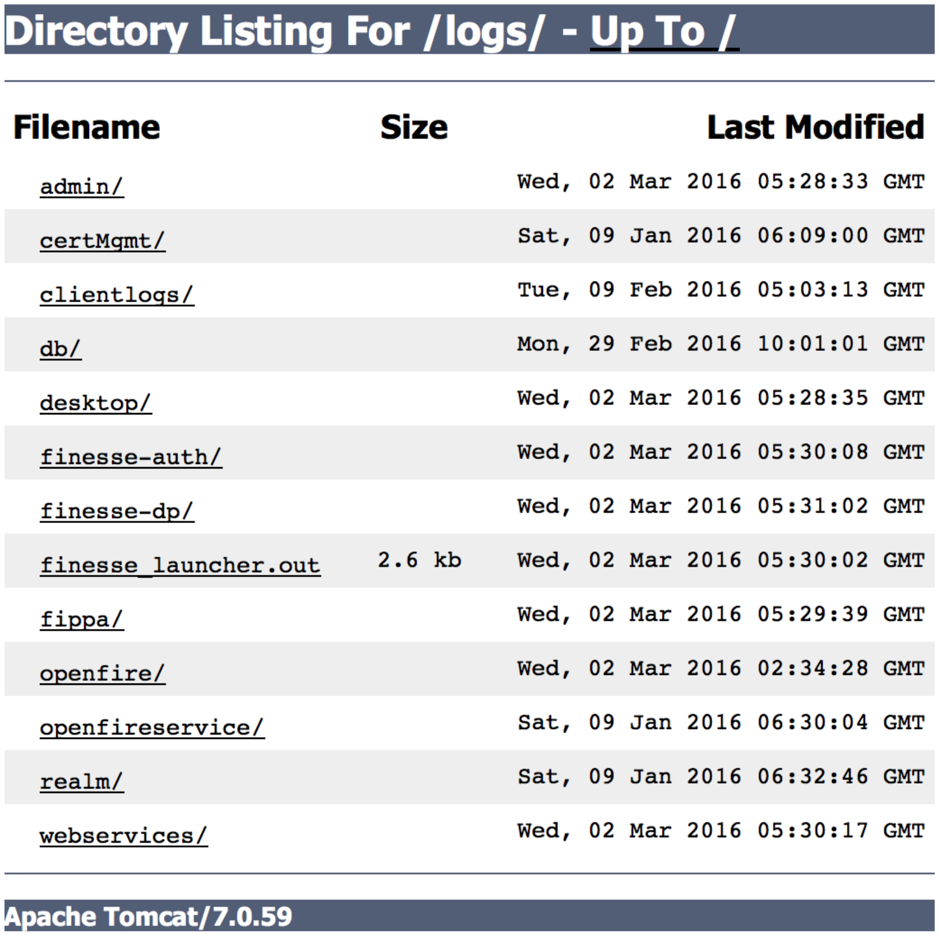
4. Get the client logs from the Finesse system. (See step 5 of the Troubleshooting section below)
5. You can also view the logging in the Console window of the browser's developer tools. Note that the developer tool has to be opened to the console window before executing the functions where the logging was added.



```
2016-01-15T15:57:35.869 -08:00: : finesse25.autobot.cvp: Jan 15 2016 15:57:35.959 -0800: SampleGadget : SampleGadget.setUserState(): The user's current state is: NOT_READY
2016-01-15T15:57:36.401 -08:00: : finesse25.autobot.cvp: Jan 15 2016 15:57:36.491 -0800: Header : [ClientServices] MasterPublisher._parseAndPublishXMLEvent() - Received XML event on node '/finesse/api/User/2251': <Update>
  <data>
    <user>
      <dialogs>/finesse/api/User/2251/Dialogs</dialogs>
      <extension>1001001</extension>
      <firstName>Peter</firstName>
      <lastName>Gibbons</lastName>
      <loginId>2251</loginId>
      <loginName>petergibbons</loginName>
      <pendingState></pendingState>
      <reasonCodeId>-1</reasonCodeId>
      <roles>
        <role>Agent</role>
      </roles>
      <settings>
        <wrapUpOnIncoming>OPTIONAL</wrapUpOnIncoming>
      </settings>
      <state>NOT_READY</state>
      <stateChangeTime>2016-01-15T23:57:36.116Z</stateChangeTime>
      <teamId>1</teamId>
      <teamName>Default</teamName>
      <uri>/finesse/api/User/2251</uri>
    </user>
  </data>
  <event>PUT</event>
  <requestId>4778f25f-7acc-4b50-a1ce-645cb597e18a</requestId>
  <source>/finesse/api/User/2251</source>
</Update>
2016-01-15T15:57:36.403 -08:00: : finesse25.autobot.cvp: Jan 15 2016 15:57:36.493 -0800: Header : _displayUserData(): User's current state is: NOT_READY
2016-01-15T15:57:36.406 -08:00: : finesse25.autobot.cvp: Jan 15 2016 15:57:36.496 -0800: Header : Agent selected or went NOT_READY -- Linda flag cleared.
2016-01-15T15:57:36.407 -08:00: : finesse25.autobot.cvp: Jan 15 2016 15:57:36.497 -0800: Header : [WorkflowEngine] Received user event.
2016-01-15T15:57:36.407 -08:00: : finesse25.autobot.cvp: Jan 15 2016 15:57:36.497 -0800: Header : [WorkflowEngine] Converting event to xml document. Type: User Action: change Uri: /finesse/api/User/2251 Event: {"User":{"dialogs":"/finesse/api/U
/2251/Dialogs","extension":"1001001","firstName":"Peter","lastName":"Gibbons","loginId":"2251","loginName":"petergibbons","pendingState":null,"reasonCodeId":"-1","roles":{"role":"Agent"},"settings":
{"wrapUpOnIncoming":"OPTIONAL"},"state":"NOT_READY","stateChangeTime":"2016-01-15T23:57:36.116Z","teamId":"1","teamName":"Default","uri":"/finesse/api/User/2251"}}
2016-01-15T15:57:36.407 -08:00: : finesse25.autobot.cvp: Jan 15 2016 15:57:36.497 -0800: Header : [WorkflowEngine] Calling state machine's workflowEvent method for /finesse/api/User/2251 (action: change)
2016-01-15T15:57:36.408 -08:00: : finesse25.autobot.cvp: Jan 15 2016 15:57:36.498 -0800: Header : [WorkflowEngine] Entering 'Busy' state, from: 'idle'. Triggering start of queued event processing.
2016-01-15T15:57:36.414 -08:00: : finesse25.autobot.cvp: Jan 15 2016 15:57:36.504 -0800: CallControl : _handleUserChange(): User state is NOT_READY and mobile agent call by call is: false
2016-01-15T15:57:36.422 -08:00: : finesse25.autobot.cvp: Jan 15 2016 15:57:36.512 -0800: Header : [WorkflowEngine] Starting queued event processing...
2016-01-15T15:57:36.423 -08:00: : finesse25.autobot.cvp: Jan 15 2016 15:57:36.513 -0800: Header : [WorkflowEngine] ------ Got User event from queue: Action: change URI: /finesse/api/User/2251 ------
2016-01-15T15:57:36.423 -08:00: : finesse25.autobot.cvp: Jan 15 2016 15:57:36.513 -0800: Header : [WorkflowEngine] Triggering processing of the next event in the queue.
2016-01-15T15:57:36.453 -08:00: : finesse25.autobot.cvp: Jan 15 2016 15:57:36.543 -0800: Header : [WorkflowEngine] ------ Event queue is empty. ------
2016-01-15T15:57:37.792 -08:00: : finesse25.autobot.cvp: Jan 15 2016 15:57:37.882 -0800: SampleGadget : SampleGadget.makeCall(): Making a call to 1001002
2016-01-15T15:57:38.336 -08:00: : finesse25.autobot.cvp: Jan 15 2016 15:57:38.426 -0800: Header : [ClientServices] MasterPublisher._parseAndPublishXMLEvent() - Received XML event on node '/finesse/api/User/2251': <Update>
```

Troubleshooting

1. **Do a sanity check**: Do the Finesse out-of-the-box gadgets work?
2. **Go back to basics**: Does a sample/simple gadget run?
3. **Add breakpoints**: Are the event handlers being triggered?
4. **Digging deeper**:
   - Issues with events: Do you see the BOSH long poll return?
   - Issues with request/response: Manually construct the REST API request. Does it work?
5. **Get server side logs**:
   - Finesse logs can be obtained by using the CLI commands. These commands will prompt the user to specify a secure FTP server location to which the files will be downloaded. To use the CLI commands, SSH to the Finesse server as the platform administrator user and type the following command for Tomcat & Finesse logs: **file get activelog desktop recurs compress**
   - Finesse logs can also be obtained by going to the logs page on Finesse using the platform administrator user credentials: http://<Finesse-IP>:<port>/finesse/logs



Directory Listing For /logs/ - Up To /

Filename	Size	Last Modified
admin/		Wed, 02 Mar 2016 05:28:33 GMT
certMgmt/		Sat, 09 Jan 2016 06:09:00 GMT
clientlogs/		Tue, 09 Feb 2016 05:03:13 GMT
db/		Mon, 29 Feb 2016 10:01:01 GMT
desktop/		Wed, 02 Mar 2016 05:28:35 GMT
finesse-auth/		Wed, 02 Mar 2016 05:30:08 GMT
finesse-dp/		Wed, 02 Mar 2016 05:31:02 GMT
finesse_launcher.out	2.6 kb	Wed, 02 Mar 2016 05:30:02 GMT
fippa/		Wed, 02 Mar 2016 05:29:39 GMT
openfire/		Wed, 02 Mar 2016 02:34:28 GMT
openfireservice/		Sat, 09 Jan 2016 06:30:04 GMT
realm/		Sat, 09 Jan 2016 06:32:46 GMT
webservices/		Wed, 02 Mar 2016 05:30:17 GMT

Apache Tomcat/7.0.59

6. **Subscribe to the Finesse Forums on CDN**: https://communities.cisco.com/community/developer/finesse