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.

How to add a gadget

- In this document, you will learn how to:
  - Add the gadget to the Finesse container
  - Set up the client services for the gadget
  - Create a user object
  - Utilize the GET/PUT methods of the user object
  - Add a make call button
  - Create a dialog object
  - Utilize the GET methods of the dialog object
  - Use the data of the dialog object to do a web search

Step 1: Download the Sample Gadget

(its assumed you have done this to get to this document)

- Download and unzip the sample gadget from CDN
- You should have the following files:
  - 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

- finesse.js: This file contains the Finesse javascript library
- jquery-1.5.min.js the library uses jQuery for some utility functions

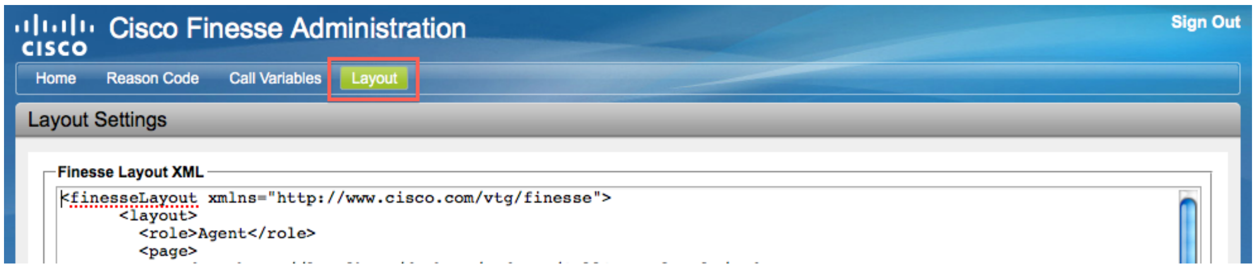
Step 2: Hosting 3rd party gadgets

- 3rd party gadgets (new 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
    - finesse.js
    - jquery-1.5.min.js

Step 3: Adding the gadget to the Finesse Container

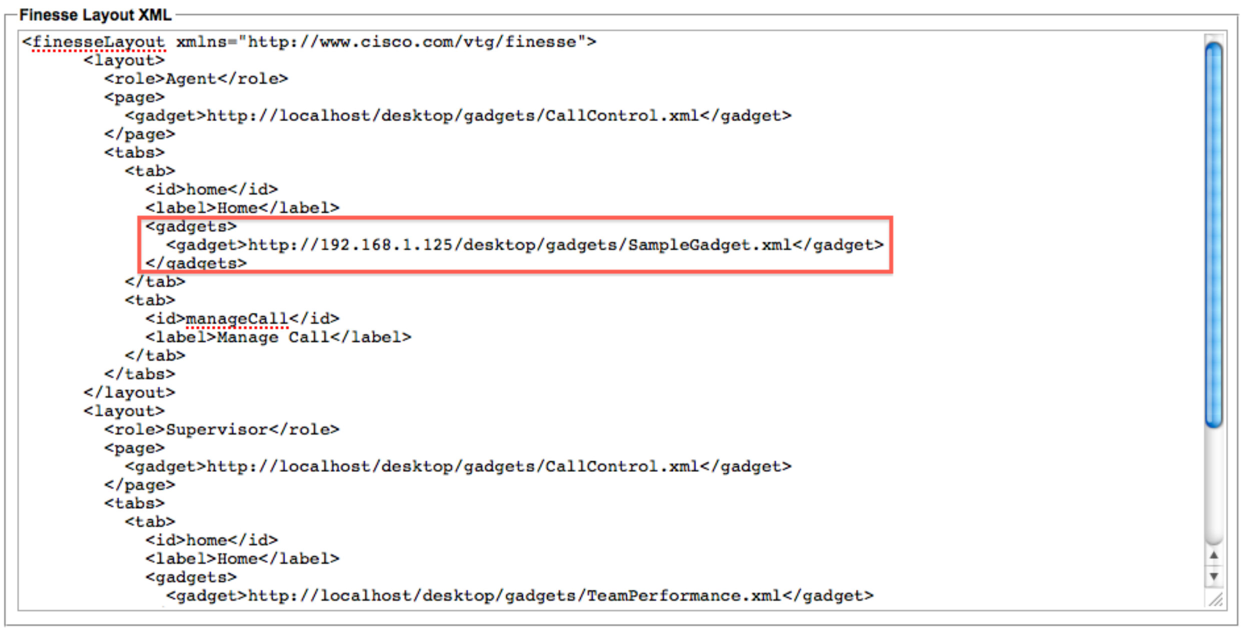
- The Finesse Layout XML defines the layout of the Finesse Desktop (both agents and supervisors), including the tab names and the gadgets shown on each tab. Tab names can appear in any language, as long as they are defined in the XML

a) Use the Layout Settings gadget by logging into the Finesse Administration and clicking the Layout tab. This gadget allows the administrator to define/modify the layout of the Finesse Desktop for agents and supervisors.



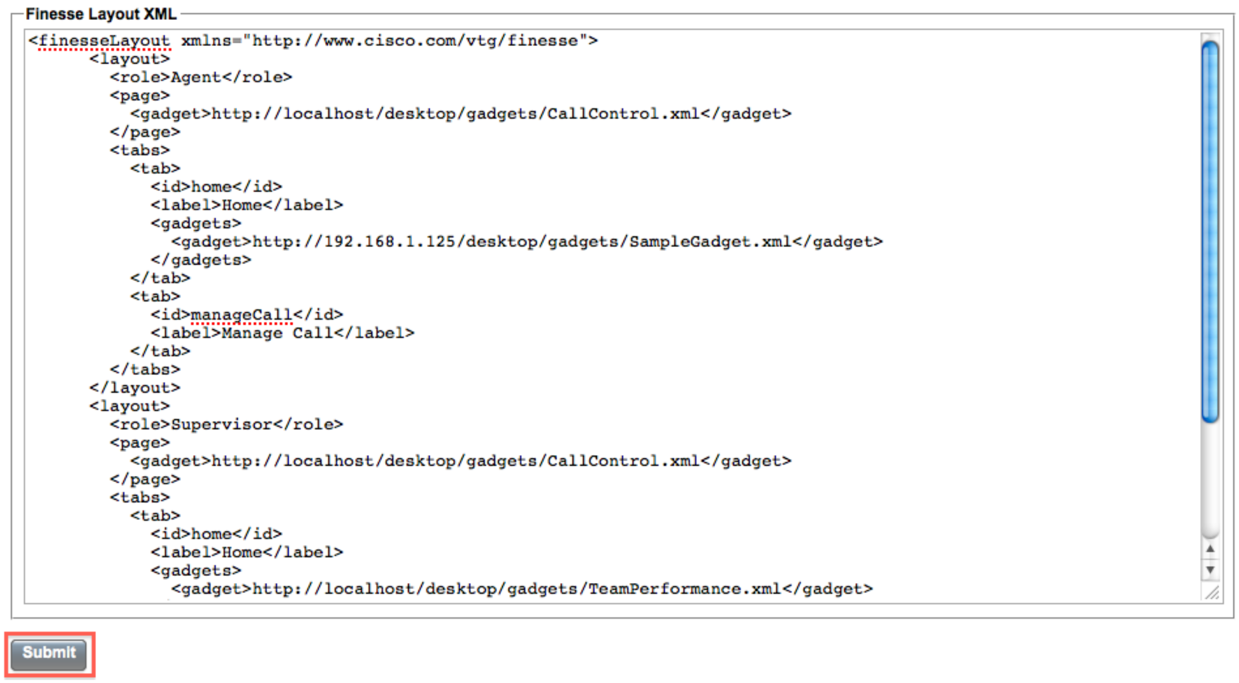
```
<finesseLayout xmlns="http://www.cisco.com/vtg/finesse">
      <layout>
        <role>Agent</role>
        <page>
```

b) 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>
```

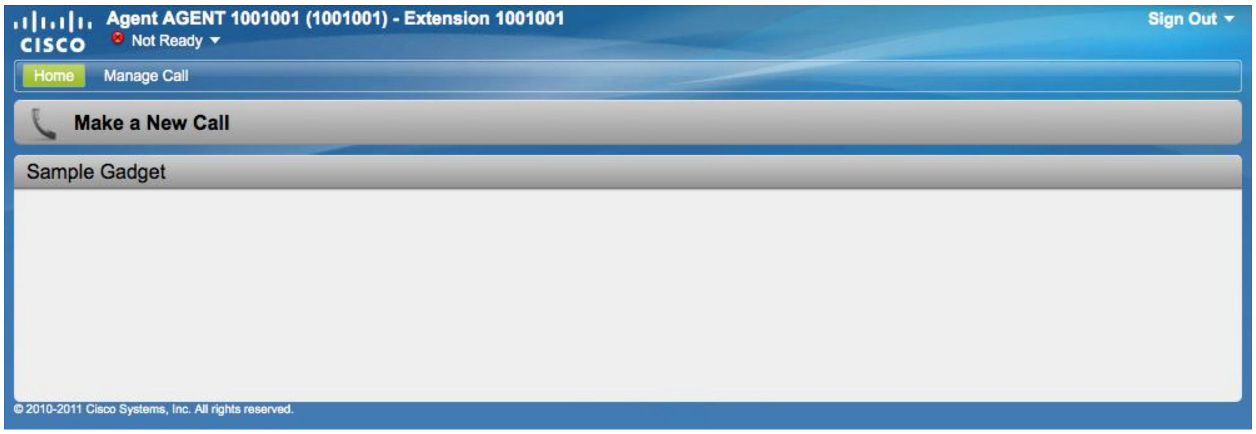
c) The Submit button should now be highlighted. Click the Submit button.



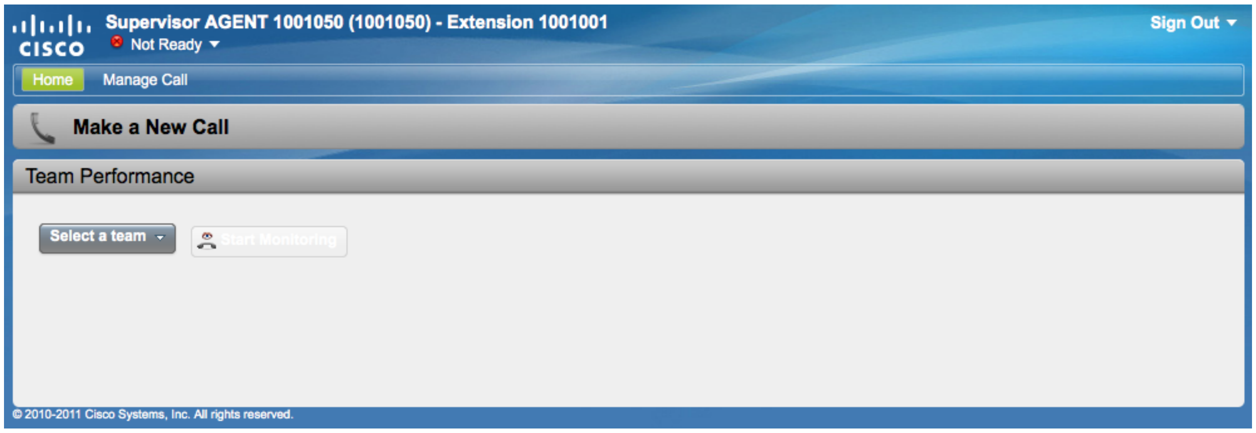
Finesse Layout XML

```
<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>
```

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


Agent AGENT 1001001 (1001001) - Extension 1001001
Not Ready
Sign Out
Home
Manage Call
Make a New Call
Sample Gadget
© 2010-2011 Cisco Systems, Inc. All rights reserved.

e) 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 AGENT 1001050 (1001050) - Extension 1001001
Not Ready
Sign Out
Home
Manage Call
Make a New Call
Team Performance
Select a team
Start Monitoring
© 2010-2011 Cisco Systems, Inc. All rights reserved.

Step 4: Get the workspace ready

The rest of this demo 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 demo 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.

a) 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 SampleGadget.js and does not need any modifications.

```
finesse.gadget.Config = (function () {
   var _prefs = new gadgets.Prefs();
   return {
      authorization: _prefs.getString("authorization"),
      host: _prefs.getString("host"),
      restHost: "localhost”
   };
} ());
```

b) In SampleGadget.js, update the init function to initialize the ClientServices

```
init : function () {
    gadgets.window.adjustHeight();
    var prefs =  new gadgets.Prefs(),
     id = prefs.getString("id");

     // Initiate the ClientServices and load the user object.  ClientServices are
     // initialized with a reference to the current configuration.
     finesse.clientservices.ClientServices.init(finesse.gadget.Config);
}
```

In SampleGadget.xml, update the function call to call init on the gadget:

```
<script type="text/javascript">

   gadgets.HubSettings.onConnect = function () {

      finesse.modules.SampleGadget.init();

            document.getElementById('bing').src =
"http://www.bing.com";

   };

   </script>
```

Step 6: Create User object

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

a) In SampleGadget.js, update the init function to add an instance of the User object

```
var user,
init : function () {
    var prefs =  new gadgets.Prefs(),
     Id = prefs.getString("id");
   …
   finesse.clientservices.ClientServices.init(finesse.gadget.Config);
   user = new finesse.restservices.User({
      id: id,
      onLoad : handleUserLoad,
      onChange : handleUserChange
   }
}
```

b) In SampleGadget.js, create the callback functions from step 6a

```
finesse.modules = finesse.modules || {};finesse.modules.SampleGadget = (function ($) {
/**
 * 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) { },
/**
```

Step 7: Add HTML DOM for the User GET

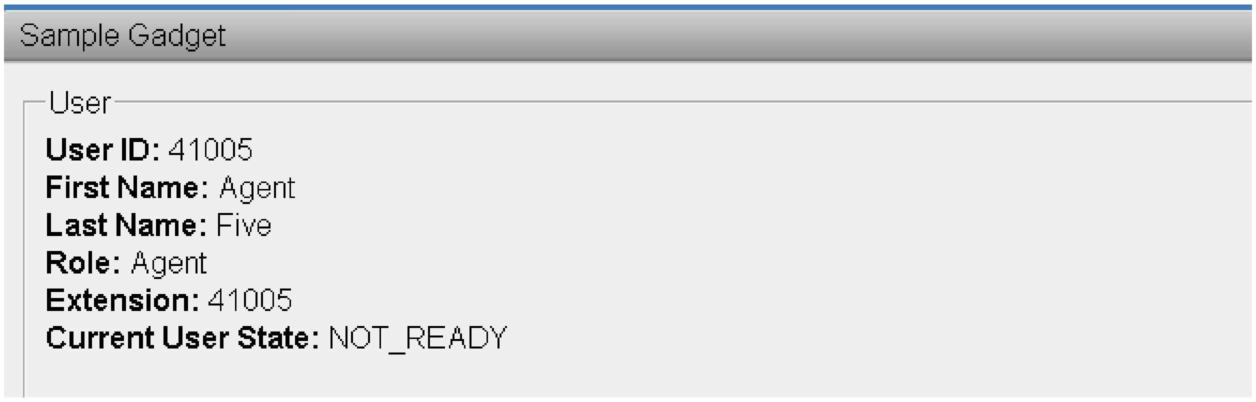
- This sample gadget will display the the user's id, first name, last name, role, extension, and it's current state.
- In this step, we are going to set up the gadget to have the ability to display the data. It will not populate the fields with user data.
- In SampleGadget.xml, add the following within the <body> tag:

```
<body class="claro”>
<div>
    <fieldset id="userfieldset” class="outline">
        <legend>Logged in as:</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>
    <br>
</div>
</body>
```

Step 8: Display User data using a GET

- This step will populate the fields from step 7 with user data.
- In SampleGadget.js, update the callback functions from step 6b

```
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();

      }
```


Sample Gadget

User

User ID: 41005
First Name: Agent
Last Name: Five
Role: Agent
Extension: 41005
Current User State: NOT_READY

Step 10: Add the User States

- User States is an enum of to maintain consistency amongst the code.
- This step is preparing the code for changing the user's state
- In SampleGadget.js, update the init function to add the user states

```
var user, states;
init = function() {
   finesse.clientservices.ClientServices.init(finesse.gadget.Config);
   user = new finesse.restservices.User({
      id: id,
      onLoad : handleUserLoad,
      onChange : handleUserChange
    }
    states = finesse.restservices.User.States;
};
```

Step 11: Add HTML DOM for the User PUT

- Now, we will update the sample gadget to have two buttons. The first button changes the user to READY and the second changes the user to NOT_READY

- This step adds the visual piece of the User PUT, but the buttons will not function correctly.
- In SampleGadget.xml, add the following code:

```
<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>
```

Step 12: Add business logic for User PUT

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

a) In SampleGadget.js, add a setUserState function (matching the onClick of step 11) to the return clause. This function calls the user PUT method with the appropriate parameter.

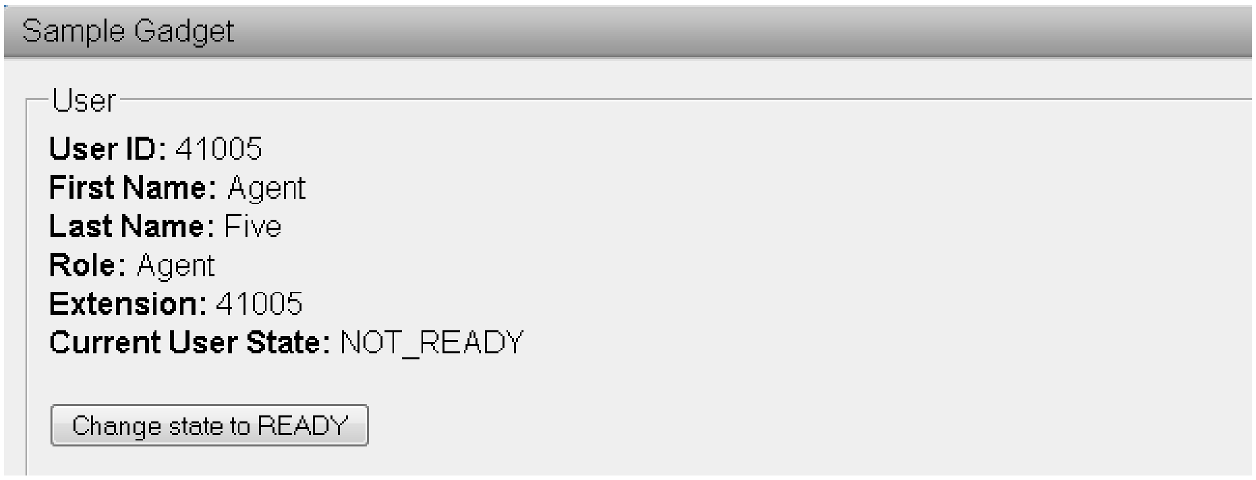
```
return {
   /**
    * Sets the user state
    */
   setUserState : function (state) {
      if (state === 'READY') {
         user.setState(states.READY);
      } else if (state === 'NOT_READY') {
         user.setState(states.NOT_READY);
      }
   },
…
};
```

b) In SampleGadget.js, update the render function to display/hide the correct button according to the user's current state:

```
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();
```

Step 13: Upload Gadget with User PUT

Repeat Step 9 and you should see:


Sample Gadget

User

User ID: 41005
First Name: Agent
Last Name: Five
Role: Agent
Extension: 41005
Current User State: NOT_READY

Step 14: Add HTML DOM for the make call

- Now, we will update the sample gadget to have another button. This button will make a call to extension 1001002.
- This step adds the button for the make call, but the button will not function correctly.
- In SampleGadget.xml, add the following code, but change the extension to a valid extension in your setup:

```
<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="1001002"></input>
          <button onClick="finesse.modules.SampleGadget.makeCall((document.getElementById
('phoneId')).value);">Make Call</button>
          </div>
 </fieldset>
```

Step 15: Add business logic for the make call

- In this step, we are going to hook in the support for the button that was added in step 14.

a) In SampleGadget.js, add a makeCall function (matching the onClick of step 14) to the return clause. This function calls the user PUT method with the appropriate parameter.

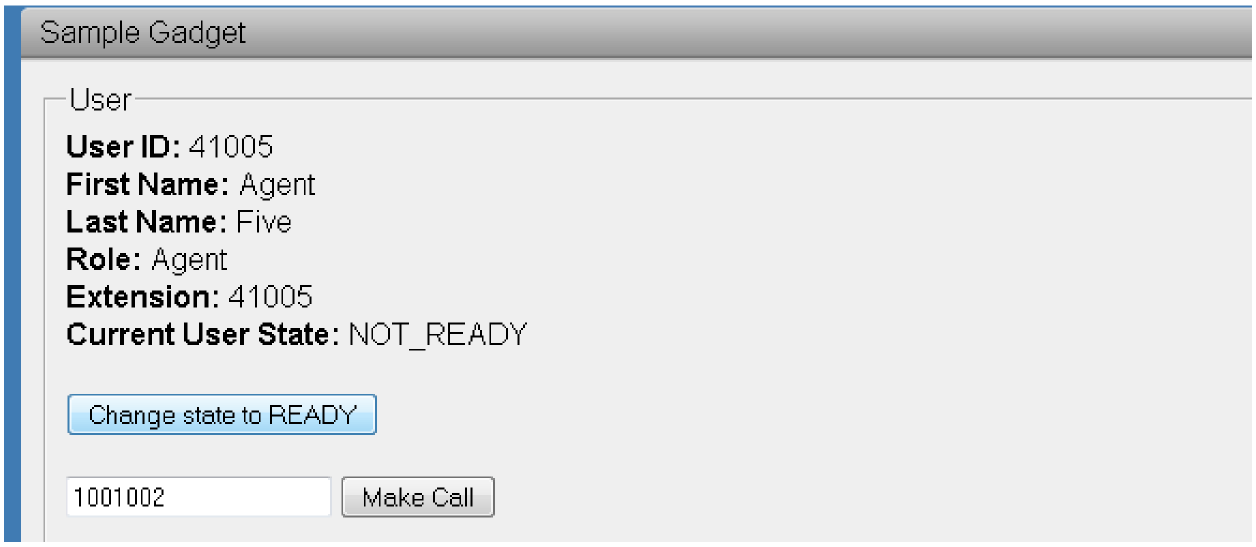
```
return {
  /**
   * Make a call to the number
   */
  makeCall : function (number) {
    // Example of the user make call method
    user.makeCall(number, {
      success: makeCallSuccess,
      error: makeCallError
    });
    // Hide the button after making the call
    document.getElementById('makeCallButton').style.display = "none";
  }
…
};
```

b) In SampleGadget.js, create the callback functions from step 15a

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

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

Step 16: Upload Gadget with make call

Repeat Step 9 and you should see:



User ID: 41005
First Name: Agent
Last Name: Five
Role: Agent
Extension: 41005
Current User State: NOT_READY

Step 17: Add the dialog object

- A dialog object represents a conversation between two or more Participants.

a) In SampleGadget.js, update the handleUserLoad function to get the list of dialogs for the user:

```
var user, states, dialogs,
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();
},
```

b) In SampleGadget.js, add the callback functions from step 17a

```
 var user, states, dialogs,
/**
 *  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) { },
…
handleUserLoad = function (userevent) {
   …
},
```

Step 18: Add HTML DOM for the Dialog GET

- Now, we are updating 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, 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.
- In SampleGadget.xml, add the following:

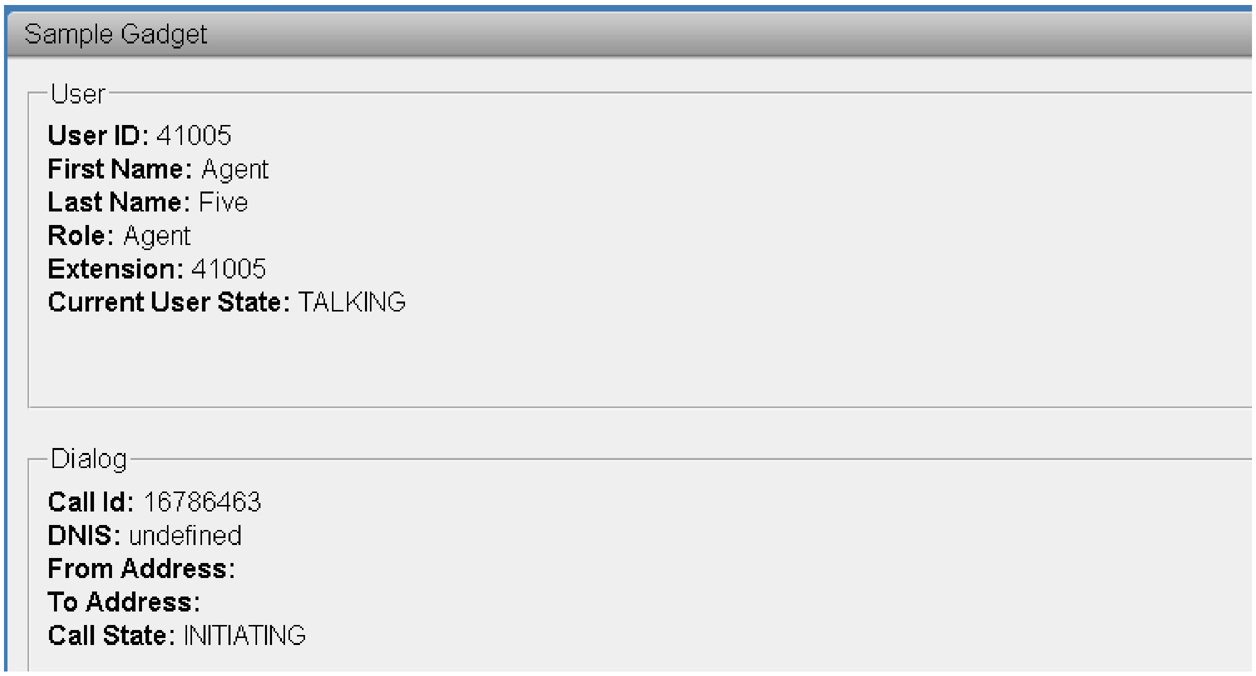
```
<fieldset id="userfieldset" class="outline">
    . . .
</fieldset>
<br>

<fieldset id="dialogfieldset” class="outline">
    <legend>Dialog</legend>
    <div><b> Call Id:  </b><span id="callId"></div>
    <div><b> 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>
```

Step 19: Display Dialog Data using GET

- This step will populate the fields from step 18 with user data.
- In SampleGadget.js, update the callback functions from step 17b

•

```
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);

                        },



handleEndDialog = function(dialog) {

          // Clear the fields when the call is ended       $("#callId").text("");

          $("#dnis").text("");

          $("#fromAddress").text("");

          $("#toAddress").text("");

          $("#callState").text("");                    $("#bing").attr("src","http://www.bing.com");

// Show the make call button when the call is ended

          $("#makeCallButton").show();

  },
```



User

User ID: 41005
First Name: Agent
Last Name: Five
Role: Agent
Extension: 41005
Current User State: TALKING

Dialog

Call Id: 16786463
DNIS: undefined
From Address:
To Address:
Call State: INITIATING

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

- This step hides the make call button (from step 14 & 15) when the user is on a call.
- In SampleGadget.js, update the handleNewDialog & handleEndDialog callback functions.

```
handleNewDialog = function(dialog) {
    ...
    document.getElementById('callState').innerHTML = dialog.getState();
    // Hide the make call button when the user is on a call
    document.getElementById('makeCallButton').style.display = "none";
},

handleEndDialog = function(dialog) {
    ...
    document.getElementById('callState').innerHTML = "";
    // Show the make call button when the call is ended
    document.getElementById('makeCallButton').style.display = "";
},
```

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 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.
- This step will add a iframe with the main bing.com site.

a) In SampleGadget.xml, update the dialogfieldset to include an iframe with bing.com as the source.

```
<fieldset id="dialogfieldset” class="outline">
        …
        <div><b> Call State: </b><span id="callState"></div>
        <br>
        <iframe name="bing" id="bing" src="http://www.bing.com" width="900" height="300"></iframe>
</fieldset>
```

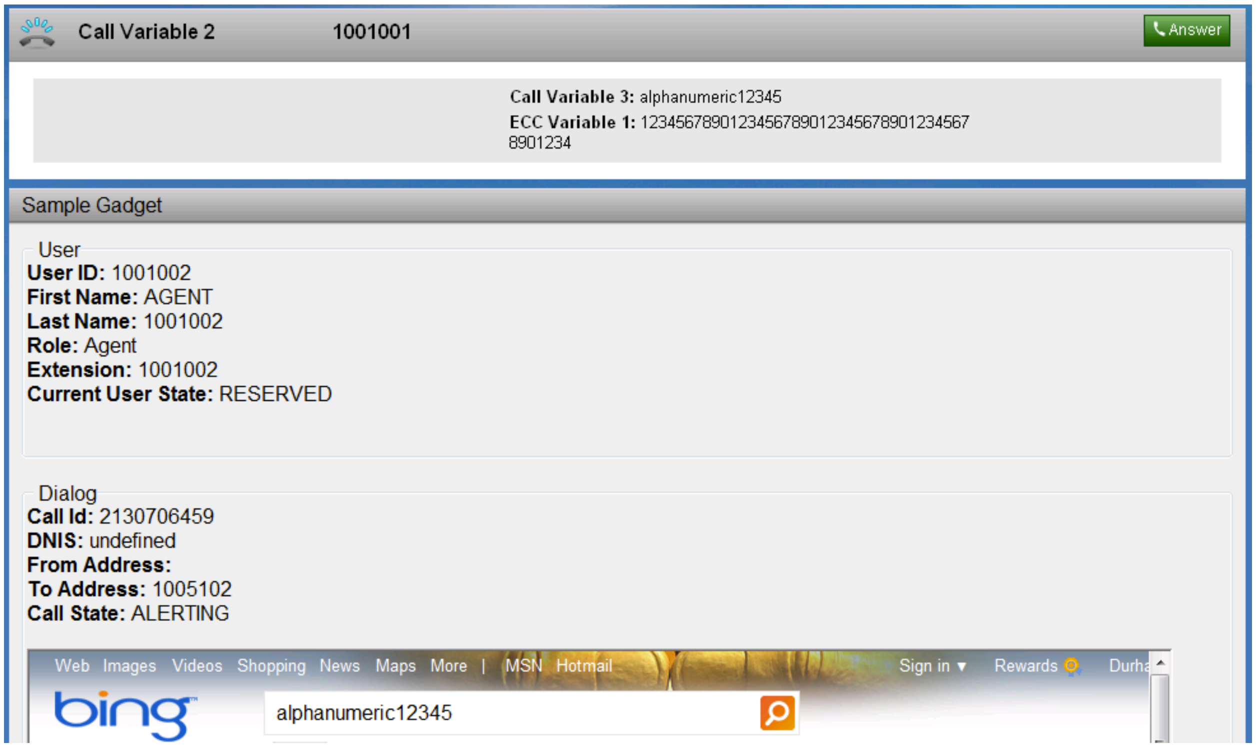
- This step will change the source of the bing.com iframe to search for call variable 3 when the user gets a new dialog event. When the call ends, the source of the iframe will return back to bing.com

b) In SampleGadget.js, update the dialog object’s callbacks:

```
handleNewDialog = function(dialog) {
   // Get call variables from the dialog
   var callVars = dialog.getMediaProperties();
   …
   document.getElementById('bing').src = "http://www.bing.com/search?q=" + callVars["callVariable3"];
},
handleEndDialog = function(dialog) {
   …
   document.getElementById('bing').src = "http://www.bing.com";
};
```

Step 23: Upload Gadget with Dialog GET

Repeat Step 9 and you should see:


Answer
Web Images Videos Shopping News Maps More | MSN Hotmail Sign in Rewards Durha

Call Variable 2 1001001

Call Variable 3: alphanumeric12345
ECC Variable 1: 12345678901234567890123456789012345678901234

Sample Gadget

User
User ID: 1001002
First Name: AGENT
Last Name: 1001002
Role: Agent
Extension: 1001002
Current User State: RESERVED

Dialog
Call Id: 2130706459
DNIS: undefined
From Address:
To Address: 1005102
Call State: ALERTING

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.

Troubleshooting

- Sanity

Do the Finesse out-of-the-box gadgets work?

- Back to basics

Does a sample/simple gadget run?

- Breakpoints

Are the event handlers being triggered?

- Digging deeper

Events

Do you see the BOSH long poll return?

Requests

Manually construct the REST API request

- How to get server side logs…

Finesse logs can be obtained by using the CLI commands. The 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 box as the platform administrator user and type the following command depending on which logs you are trying to obtain:

a) Desktop logs: file get activelog desktop recurs compress

b) Platform Tomcat logs: file get activelog tomcat/logs recurs compress

Or browse to the logs page on finesse:

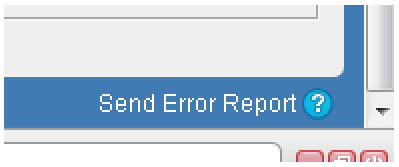
http://<finesse-ip>/finesse/logs

Custom gadget logging

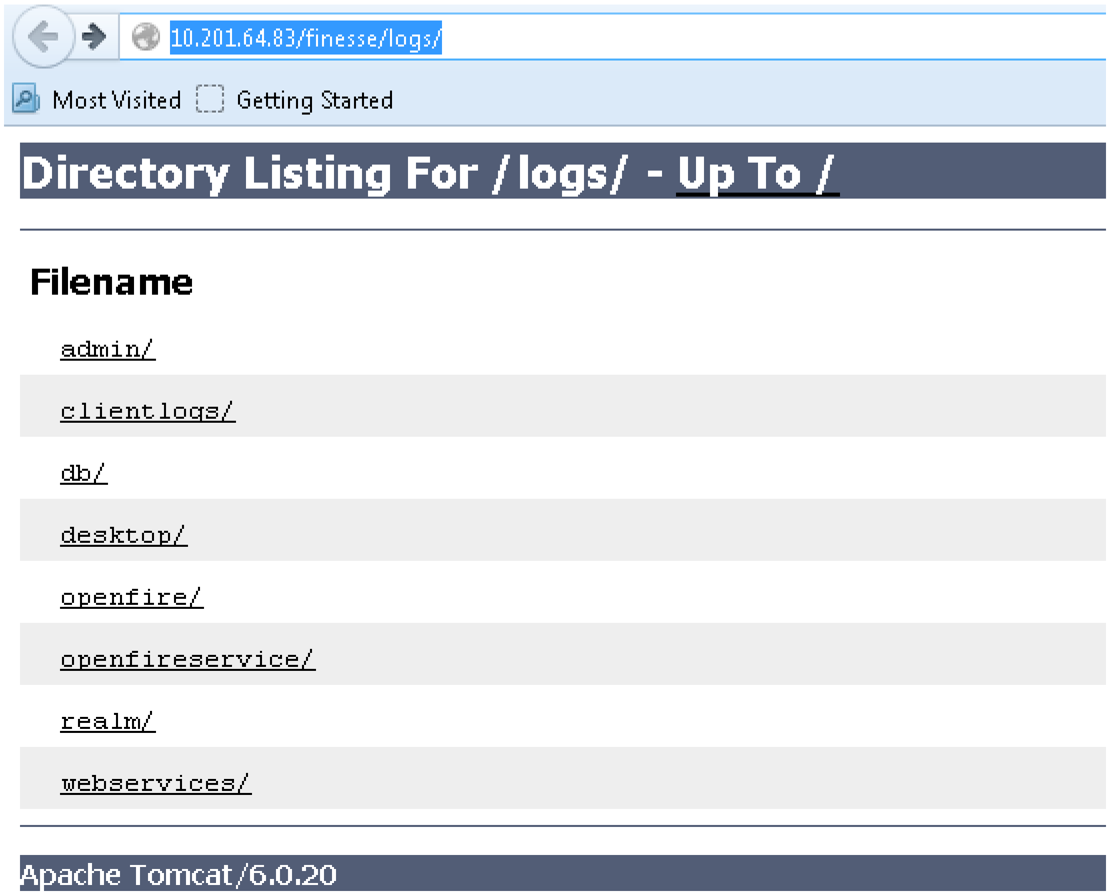
- See the latest version of the LearningSampleGadget which has javascript to declare and initialize the Finesse logger and log things using:

  clientLogs.log("log something");

- Click on "Send Error Report" in the bottom right corner of the Finesse Agent Desktop



- Review clientlogs



Directory Listing For /logs/ - Up To /

Filename

```
admin/
clientlogs/
db/
desktop/
openfire/
openfireservice/
realm/
webservices/
```

Apache Tomcat/6.0.20

- Subscribe to the Finesse forums on CDN

http://developer.cisco.com/web/finesse/community