Creating a custom gadget using the Finesse JavaScript Library API

Denise Kwan, Software Engineer @ DevNet
Questions?
Use Cisco Spark to communicate with the speaker after the session

How
1. Find this session in the Cisco Live Mobile App
2. Click “Join the Discussion”
3. Install Spark or go directly to the space
4. Enter messages/questions in the space

cs.co/ciscolivebot#DEVNET-1998
Agenda

• Introduction to Finesse
• Introduction to Gadgets
• Hands-on Workshop
Cisco Finesse

Revolutionizing the Contact Center Agent Experience

- Browser-based agent desktop for easy management & upgrades
- Empower agents via a personalized all-in-one application
  - Integrate Finesse into existing applications (thick or thin)
  - Integrate existing applications into the Finesse agent desktop
- Easy-to-use REST and JavaScript APIs for performing agent and supervisor actions programmatically
New in Finesse 11.6

- Integrated System generated Not Ready Reason Codes
- Secondary CallId for transfers and conferences
- Alternate host for loading gadgets
- State and call history gadgets
- Direct Transfer
- Ability to make a call from READY state
- Ability to monitor Outbound calls on ACD line

Reserve the FREE UCCX 11.6 sandboxes to try out these new features!
What is a gadget?

- Finesse gadgets are OpenSocial Gadget, which is an XML document that defines metadata for an OpenSocial Gadget container (Finesse desktop).
- Mini webpages within a gadget.
- It is highly cacheable so it does not need a high performance server.
- The gadget includes:
  - XML to define metadata
  - HTML for markup
  - JavaScript for interactivity
  - CSS for presentation
- Gadgets on the same container can communicate and pass data with one another as well as backend servers allowing for a seamless user experience.
What can you build with a gadget?

- Embed an existing webpage in an iframe.
- Integrate external REST APIs from applications such as the CRM.
- Custom functionality using the Finesse JavaScript API that is not available from the out of the box desktop.
What Skills do I need to build a Finesse Gadget?

- Finesse Gadgets are OpenSocial Gadgets, so you will need the following skills:
  - HTML
  - Basic JavaScript
- You can find these type of courses online for free at:
  - Udacity
  - Coursera
  - Khan Academy
  - Many more!
Finesse Developer Guide Documentation

On DevNet’s Finesse site:
- On your browser, go to the DevNet home page: http://developer.cisco.com
- Click on the Technologies tab
- Click on Collaboration on the left hand side and select Finesse under Contact Center
- Click on Docs on the upper right
- Select JavaScript Library under Guides

What is Finesse?
Cisco Finesse is a next-generation agent and supervisor desktop designed to provide a collaborative experience for the various communities that interact with your customer service organization. It
Understanding the Gadget Files

- SampleGadget.xml is the gadget specification that consists of three major parts:
  ```xml
  <?xml version="1.0" encoding="UTF-8" ?>
  <Module>
    <ModulePrefs title="MyGadget">
      <Require feature="pubsub-2" />
    </ModulePrefs>
    <UserPrefs/>
    <Content type="html">
      <![CDATA[
        Your HTML & JavaScript HTML goes Here!
      ]]>'
    </Content>
  </Module>
  ```
  - ModulePrefs define the characteristics of the gadget, such as title, author, preferred sizing, etc.
  - The PubSub feature allows communication between gadgets. In order to receive Finesse events, PubSub is required
  - UserPrefs describes the user-specific settings for the gadget such as input fields
  - Content consists of the gadget’s customization and business logic, such as the programming logic and HTML elements that determine the appearance of the gadget

- SampleGadget.js is the gadget’s JavaScript file for interactivity.
- SampleGadget.css is the gadget’s styling for presentation.
Sample Gadgets

• Download an existing sample gadget from the DevNet Finesse page:
  • https://developer.cisco.com/docs/finesse/#sample-gadgets

• Follow the instructions from the LearningSampleGadget
  • https://github.com/CiscoDevNet/finesse-sample-code/tree/master/LearningSampleGadget
Hands-on Workshop
Finesse JavaScript Library

• Finesse JavaScript library is built on top of the Finesse REST APIs
• Finesse out of the box desktop utilizes the Finesse JavaScript library
• Starting 10.6(1), the Finesse JavaScript library is hosted on the Finesse server under /desktop/assets/js/finesse.js
Finesse APIs and notifications

- The Finesse JavaScript library has a utility, named `ClientServices`, that allows clients to make Finesse API requests and receive Finesse events.

- `ClientServices` establishes the shared BOSH connection and simplifies the consumption of Finesse events.

- This utility can be used by adding a single line of code in your gadget file.
How do I use the ClientServices?

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

```
init : function () {
    var cfg = finesse.gadget.Config; // Used to get the user's id
    clientLogs = finesse.cslogger.ClientLogger; // Used for client logging
    states = finesse.restservices.User.States; // Used to set user state

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

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

    // Create an instance of the User object
    // This will call a GET User API to populate the object

    ...
}
```
Getting the user’s data

• User data such as agent name, extension, agent state can be accessed through the various methods in the User class.

• The user object represents a single user. Function calls on this object will only operate against that particular user.

• Creating an instance of the User class with the logged in user’s id calls the User GET API and returns a User object populated with the user’s data.
How do I create a User object?

- In SampleGadget.js, update the `init` function to create an instance of the User:

```javascript
init : function () {
...

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

// Create an instance of the User object
// This will call a GET User API to populate the object
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
}
```
How do I get the user’s data?

- In SampleGadget.js, update the `render` function to call the getter methods from the user object:

```javascript
render = function() {
    // Examples of getting data from the User object.
    // This will not call the User GET, it displays the
    // latest data from the User object.

    var currentState = user.getState();
    var agentName = user.getFirstName() + ' ' + user.getLastName();
    $('.agentName').text(agentName);
    $('.userState').text(currentState);

    ...
}
```
Changing agent state

• Changing user state is critical to contact center operations and can be done with a simple and single method call on the User object.

• The setState method uses the ClientServices to call the Change Agent State (PUT) API.

• In order to maintain consistency and clean code, the Finesse JavaScript library provides a user states enum (finesse.restservices.User.States) that represents all the user states.
How do I change the user’s state?

- In SampleGadget.js, update the `setUserState` function to call the user change state with the appropriate parameter:

```javascript
/**
 * Sets the user state to the provided state
 */
setUserState : function (state) {
  clientLogs.log("SampleGadget.setUserState(): The user’s wants to change their state to: " + state);

  // Example of using the User - Change Agent State (PUT) method
  if (state === 'READY') {
    user.setState(states.READY);
  } else if (state === 'NOT_READY') {
    user.setState(states.NOT_READY);
  }
},
```
User’s dialogs/calls

• In Finesse, calls are represented as a dialog with media type of Voice.

• A **dialog** object represents a conversation between two or more participants. A participant represents an internal or external user's CallConnection, or that user's leg of the call.

• Getting a list of the user's dialogs can be done by calling the user's getDialogs method. This method internally calls the **GET list of dialogs REST API**.

• When the ClientServices utilities receives a dialog event, it calls the callback handler that was added to the getDialogs method with the updated dialog data.
How do I get a list of the user’s calls/dialogs?

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

```javascript
/**
 * 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) {
    // Example of using the User — Get List of Dialogs (GET) method
    // Get an instance of the dialogs collection and register handlers for when the user gets a
    // new dialog (add) and when a dialog is dropped (delete)
    dialogs = user.getDialogs( {
        onCollectionAdd : handleNewDialog,
        onCollectionDelete : handleEndDialog
    });

    // Render the gadget
    render();
},
```
Making a call

• The User object has a makeCall method to make a call to an extension. This method internally calls the Create a New Dialog REST API.

• This is useful to incorporate a click to call feature to the gadget.
How do I make a call?

• In SampleGadget.js, update the `makeCall` function to call the makeCall method. This function calls the user PUT method:

```javascript
/**
 * Make a call to the provided number
 */
makeCall : function (number) {
    clientLogs.log("makeCall(): Making a call to " + number);
    // Example of using the Dialog - Create a New Dialog (Make Call) (POST) method
    user.makeCall(number, {
        success: makeCallSuccess,
        error: makeCallError
    });

    // Hide the make call button after making the call
    $("#makeCallButton").hide();
},
```
Using Call Data (Call Variables)

- Call data can hold important customer information such as customer name, phone number, account number, etc.

- This information can be used in many ways:
  - Personalized customer service
  - Account lookup against the CRM
  - Scheduling a callback

- All of the call data can be accessed from the Dialog object via getters.
How do I get the dialog data (Call Variables)?

- In SampleGadget.js, update the displayCall method to change the text and onClick argument of the search buttons.

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

    // Example 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']);
};
```
Tips and Tricks

• The only way to test the gadget is to upload it to the hosting webserver. Gadget caching is enabled on the Finesse Desktop. So, use ?nocache while developing a gadget.

• Use an existing sample gadget as a base for your gadget.

• Look at the Finesse Developer guide for more details about the User, Dialog, Team objects.
How to Learn More About Finesse

• At Cisco Live!
  • DevNet Workshop - Finesse APIs: Getting Started with the REST APIs and XMPP
    Wednesday, 1/30 @ 11:00 am | Workshop 3
    Thursday, 2/1 @ 1:00 pm | Workshop 3
  • Finesse Learning Labs

• Finesse DevNet site: http://developer.cisco.com/site/finesse
  • Finesse Community (fourm)
  • Finesse Sample Gadgets
  • REST API & JavaScript library Developer Guides
  • Learning Labs
  • Finesse Sandbox (PCCE & UCCX)
Questions?

Use Cisco Spark to communicate with the speaker after the session

How

1. Find this session in the Cisco Live Mobile App
2. Click “Join the Discussion”
3. Install Spark or go directly to the space
4. Enter messages/questions in the space

cs.co/ciscolivebot#DEVNET-1998
• Please complete your Online Session Evaluations after each session

• Complete 4 Session Evaluations & the Overall Conference Evaluation (available from Thursday) to receive your Cisco Live T-shirt

• All surveys can be completed via the Cisco Live Mobile App or the Communication Stations

Don’t forget: Cisco Live sessions will be available for viewing on-demand after the event at www.ciscoblive.com/global/on-demand-library/. Complete Your Online Session Evaluation
Continue Your Education

• Demos in the Cisco campus
• Walk-in Self-Paced Labs
• Tech Circle
• Meet the Engineer 1:1 meetings
• Related sessions
Thank you
You’re