Oracle® Guided Learning

Knowledge Articles

F38165-11

Oracle Guided Learning Knowledge Articles,

F38165-11

Contents

Preface

This preface describes the document accessibility features and conventions used in Oracle Guided Learning.

Accessibility

OGL is designed with accessibility features to support assistive technologies, such as screen readers, as well as keyboard navigation. For more information about Oracle's commitment to accessibility, see http://oracle.com/accessibility.

Navigation & Keyboard Shortcuts

Users can navigate within OGL, transition between OGL and the host application, and access all relevant functionality using only the keyboard. Assistive technology is not required to use keyboard-only navigation. When a guide is running and a tip appears, the focus will be placed on the tip and then the Tab key may be used to navigate within the tip. Tabbing accesses the OGL Launcher as the last item on a page's Tab order. Using the keyboard shortcut below to open and close the Help Panel is a quicker method to access OGL.

Keyboard shortcuts enable user interface actions without a mouse. Following is a list of shortcuts available for OGL and their function:

Windows Keyboard Shortcut	Mac Keyboard Shortcut	Description
Alt+Ctrl+H	Control+Option+H	Opens and closes the Help panel the Help Panel. Use the Tab key to move through the Help Panel items / and press Enter to select the desired choice.. (For launch guide/ Job Aid options) use arrow keys, then Enter.
Alt+Ctrl+G	Control+Option+G	Toggle the focus from the tip on the screen to the element to which it is associated. The same shortcut is used for toggling focus on the Feedback modal. Please see below for variations for Splash tips with the target element “body”: • For splash without overlay: Give focus to the first focusable element on the page. • For splash with overlay: Screen reader will read "Unable to switch focus from splash tip with overlay".
Alt+Ctrl+L	Alt+Ctrl+L	Use this shortcut to open or close the language selection menu in the Help Panel.
Alt+Ctrl+M	Control+Option+M	For Legacy Help Panel only, this shortcut moves the focus to the first display group on the display group menu in the Help Panel.

Windows Keyboard Shortcut	Mac Keyboard Shortcut	Description
Alt+Ctrl+T	Control+Option+T	Enter/Exit task list mode for autoloaded task lists. Entering into task list mode will select the first task list on the screen and use the tab key to select any action items. Use '[' or ']' to switch between multiple task lists. When using some screen reader, you may need to use 'Alt +]' or 'Alt +]' . When you launch a task list item, it automatically exits from task list mode so that other guide shortcuts should not be get blocked. **Note:** This shortcut is functional for autoloaded task lists only.
Alt+Ctrl+B	Control+Option+B	Toggle the pulsing of active beacons on the page on or off.
Alt+Ctrl+S	Alt+Ctrl+S	Use this shortcut to switch on/off hover mode. When this mode is switched on, the first hover tip in the page will appear. The “[“(left bracket) or “]” (right bracket) will navigate the user around all the hover tips available on the page. (Smart tip and Beacon tip are considered hover tips). Use the same Alt+Ctrl+S shortcut to exit this mode. NB: Navigation keys vary from one screen reader (like JAWS) to another.
Alt+Ctrl+C	Control+Option+C	Toggle the focus from the element to smart tip it is associated with. the same shortcut can be used to navigate back to element.
Esc Key	Esc Key	If the focus is on the Help Panel, the Esc button will close the Help Panel and return focus to the host application. If the focus is on a tip, the Esc button will close the tip AND close the guide.

To disable or adjust keyboard shortcuts apply/embed the following scripts:

Action	Script to Embed
Disable the keyboard shortcut	*iridize.allowKeyboardShortcuts = false;*
Override default keyboard shortcut	*iridize.hotKeys = {guideFocus : 'Alt+G', showOrHideBeacon : 'Ctrl+Alt+B' }*
Disable a specific keyboard shortcut (Set an empty string as per the example)	iridize.hotKeys = {guideFocus : 'Alt+G', showOrHideBeacon : '' }

Here is the complete list of hot key names which can be used in embed JS as explained above:

Hotkey name	Default value (Windows)	Default value (Mac)
helpWidget	Alt+Crtl+H	Control+Option+H
languageSelection	Alt+Ctrl+L	Control+Option+L
displayGroupFocus	Alt+Ctrl+M	Control+Option+M
guideFocus	Alt+Ctrl+G	Control+Option+G

Hotkey name	Default value (Windows)	Default value (Mac)
hoverTipMode	Alt+Ctrl+S	Control+Option+G
taskList	Alt+Ctrl+T	Control+Option+T
showOrHideBeacon	Alt+Ctrl+B	Control+Option+B
currentSmartTip	Alt+Ctrl+C	Control+Option+C

1

Administration

This chapter contains OGL Administration knowledge articles.

Deployment Best Practices

Guided Learning support two deployment modes out of the box: development mode and production mode.

You can determine whether Guided Learning will work in **production** mode or in **development** mode by setting the ***env*** variable in the embed code. Refer to our FAQ about setting up Guided Learning below.

Production Mode

Production mode should be used in your live application; this is where your users engage with the guides that you have published.

In production mode Guided Learning:

- Displays only published versions of your guides. That is, if a guide is not published it will not be available.
- Uses the our CDN to cache your guides for better latency and high availability.
- Uses your internal browser cache to store recently used guides.

Development Mode

Development mode is where you create, edit and test your guides before publishing them.

In development mode Guided Learning:

- Displays the newest versions of your guides even if they are not published yet.
- Does not use any type of caching so that changes that you make are immediately propagated without any delay.

It is highly recommended to embed Guided Learning in development mode in your staging environment that is the environment where you run all your tests before you release to production. This enables the training professionals to create and test the guides on the newest features and improvements before they are released to production.

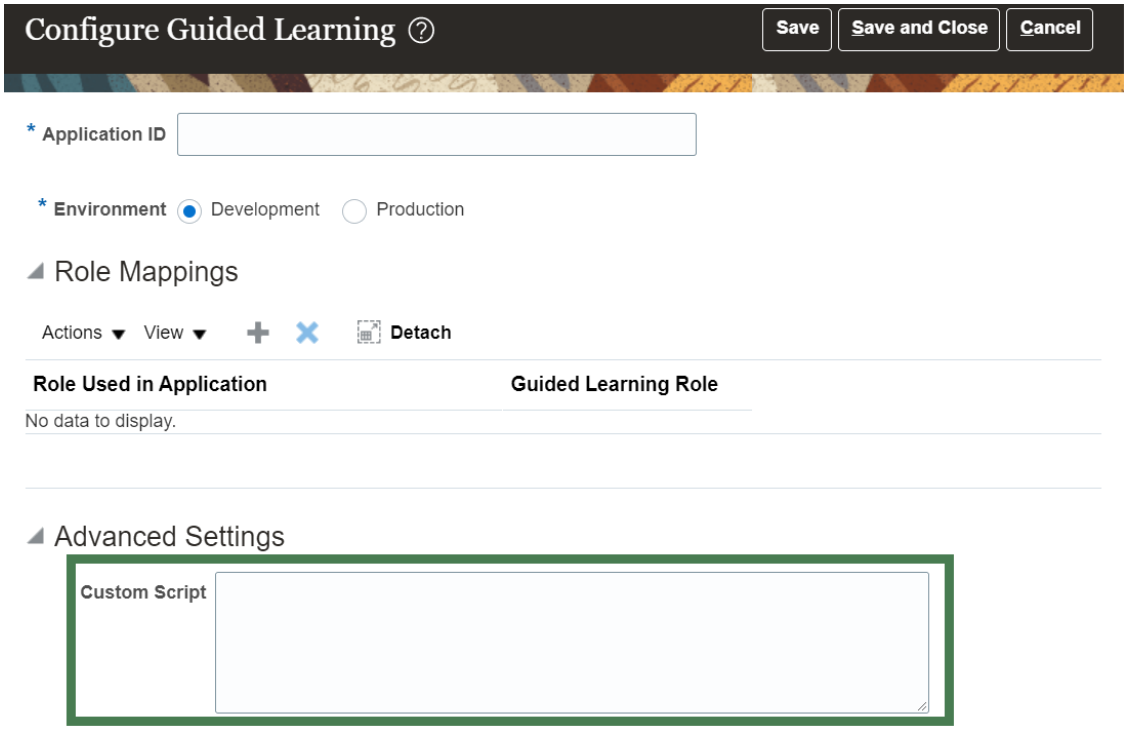
Supported Custom Scripts for OGL

What is a Custom Script?

A Custom Script is simply an OGL JavaScript that allows us to control key features of OGL such as Multi-language support, Role Mapping, etc.

Where do I find the Custom Script field?

Open the **Configure Guided Learning** task in Setup and Maintenance. The Custom Script field is in the **Advanced Settings** section of the page. Refer to the image below:



IMPORTANT: Always use **Paste as Plain Text** when pasting into the Custom Script field.

Which OGL features require a Custom Script?

- Custom Role Mapping
- Enabling Multi-Language for OGL
- Cross Domain Support
- Cross Application Support
- Country Segmentation Support
- EMEA Tenancy
- Third Party URL
- Adding Varset for customer who has more than 300+ characters

Custom Role Mapping

Use a Custom Script for role mapping only if the method outlined in the Configuring OGL - Setting Roles and Scriptsguideline is not suitable for your application. Add the below lines to the Custom Script field:

var user_role =
';**if(#{securityContext.userInRole['**REPLACE_WITH_FUSION_ROLE_CODE**']})**
{user_role+='|replace_with_preferred_role_name**|';}**

window.ir_fields.user_role+=user_role;

IMPORTANT: Ensure these lines precede any existing Custom Script in the field. You can include additional roles by duplicating the entire **IF STATEMENT:**
if(#{securityContext.userInRole['REPLACE_WITH_FUSION_ROLE_CODE**']})**
{user_role+='|replace_with_preferred_role_name**|';}**

Enable Multi-Language for OGL

By default, OGL is set to support one primary language. For Multi-language deployments of OGL, add the below lines to the Custom Script field:

iridize.supported_langs = ['**--**','**fr**', '**nl**', '**ja**', '**it**', '**de**', '**es**'];

iridize.lang = document.documentElement.lang == "en" ? "--" : document.documentElement.lang.toLowerCase();

iridize.lang = iridize.supported_langs.indexOf(iridize.lang) >= 0 ? iridize.lang : '--';

Note: Update your supported languages with the relevant language codes. This applies only to the languages translated in the OGL account and not the application languages. For instance, if a client provides the Oracle application in English, French, and Spanish, and the OGL account only provides English and Spanish, the user will only be able to see OGL in English or Spanish from the OGL Help Panel.

Cross Domain Support

In cases where OGL is required on a page/work area in the application which is in an iFrame (meaning the iFrame loads content from a different location on the same domain), we need to add the below line in the Custom Script field:

```
iridize.stateOnly=window != top ? true : undefined;
```

Cross Application Support

If your Fusion application links to another application (i.e. CPQ) and requires OGL to work seamlessly between the two applications, you need to enable cross application support by adding the below line in the Custom Script field.

```
iridize.crossAppIframe=true;
```

Country Segmentation Support

To map guides based on the user's country, add the below line in the Custom Script field:

```
window.ir_fields["user_country"] = "#{Profile.values.FND_TERRITORY}";
```

EMEA Tenancy

By default, OGL will point to OCI-NAMER-Production Tenancy (guidedlearning.oracle.com) in all the fusion applications. This should be changed for EMEA customers by adding the below lines in Custom Script.

```
iridize.jsPrefix= "https://guidedlearning-emea.oracle.com/player/latest/static";
iridize.reportPrefix= "https://guidedlearning-emea.oracle.com/player/latest";
```

Note: This is required only for EMEA Tenancy customers.

Third Party URL

To fetch a third party URL from fusion we need to add below line into Custom Script. In the below.

For example ‘3rdPartyPageName’ is the page of the URL we are trying to fetch.

iridize("api.vars.set", {"thirdPartyUrl":"#{EndPointProvider.externalEndpointByModuleShortName[‘<<**3rdPartyPageName**>>']}"});

Note: Replace the 3rd party page name in the << >> parenthesis.

Adding Varset for customer who has more than 300+ characters

To fetch more than 300+characters from the JS:

```
iridize("api.vars.set", window.ir_fields);
```

Below code is not required / redundant

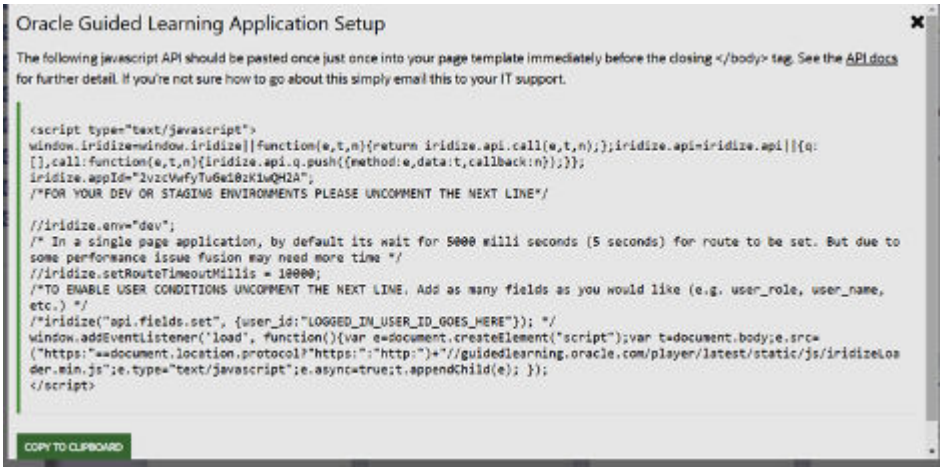
For mobiles guide deployment in Fusion Embed:

```
var ogl_mobile =
'phone'=='#{requestContext.agent.type}'||('simpleMobile'=='#{
applcoreShellBean.shellType}');
iridize("api.vars.set", { ogl_mobile : ogl_mobile });
```

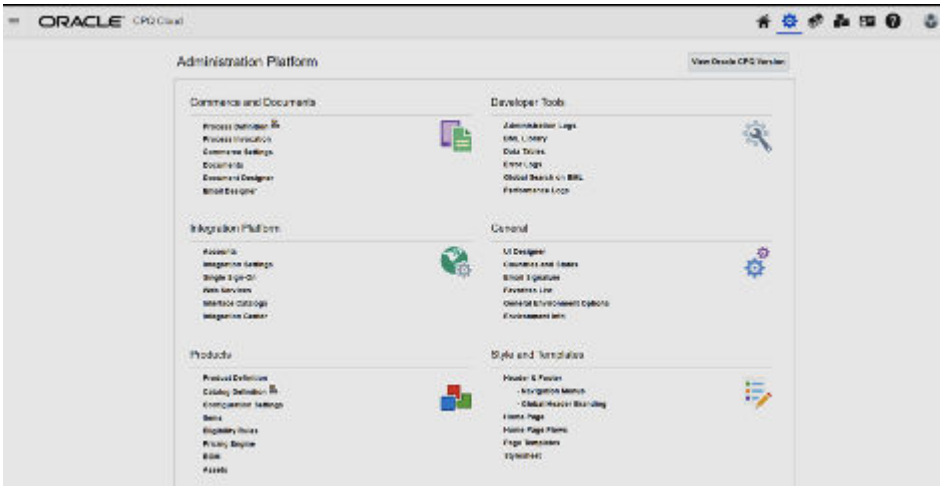
How to Enable OGL in CPQ

Because CPQ is a non-Fusion instance, we can use the default JS which is available in setup.

1. Copy the code



2. In CPQ Cloud, from *Administration Platform*, select **Header & Footer** from **Style and Templates**



3. In **Header & Footer** page add your JS (which is copied to either Header HTML or Footer HTML) then select **Accept**

Possible Errors:

Widget may not appear on any page except the header and footer page.

To fix this, add this line of code just below the iridize.env line:

```
iridize.loadJQuery=true;
```

Note: If you are trying to use a cross app functionality please add the below line into JS.

```
iridize.crossAppIframe=true;
```

How to Participate in Oracle Guided Learning UAT

The OGL Console is available at https://guidedlearning-uat.oracle.com.

To access, please use the same login information that is used for the production environment. The UAT environment will have a snapshot of data on Production taken during the week leading up to the UAT. The data set that OGL Analytics pulls from, therefore, will be static throughout the UAT, so any activity during the UAT will not reflect in the Analytics Dashboard. ***Please note that any work done on UAT will not be saved and will be wiped out when the environment is taken down after 20D is live.***

Oracle Guided Learning is integrated with its host application either via a JavaScript snippet or a browser extension. Making the update so that your application connects to UAT is simple, but the steps are a little different depending on how OGL is implemented.

Step-by-step guide

JavaScript

In the JavaScript there will be one line of code that reads as follows:

("https:"==document.location.protocol?"https:":"http:")+"//guidedlearning.oracle.com/player/latest/static/js/iridizeLoader.min.js";e.type="text/javascript";e.async=true;t.parentNode.insertBefore(e,t);})();

Replace

guidedlearning.oracle.com

with

https://guidedlearning-uat.oracle.com/

Extension

Locate the extension.js and chrome.js files in the js/src folder of the OGL extension.

extension.js

In the extension.js file, there will be a line of code that reads as follows:

'(function(){var e=document.createElement("script");var t=document.getElementsByTagName("script")[0];e.src=("https:"==document.location.protocol?"https:":"http:")+"//guidedlearning.oracle.com/player/latest/static/js/iridizeLoader.js";e.type="text/javascript";e.async=true;t.parentNode.insertBefore(e,t);})();'

Replace

guidedlearning.oracle.com

with

https://guidedlearning-uat.oracle.com/

chrome.js

In the chrome.js file, there will be a line of code that reads as follows:

value = details.responseHeaders[i].value.replace(/;/g, "guidedlearning.oracle.com;");

Replace

guidedlearning.oracle.com

with

https://guidedlearning-uat.oracle.com/

Integrating Oracle Guided Learning with your Learning Management System (LMS)

If you are using Guided Learning for employee training and you have a Learning Management System you might want to integrate the two.

There are several ways in which you can integrate Guided Learning with your LMS of choice.

Linking to a Guided Learning guide from the LMS

You can use a permalink to create a link that will launch a Guided Learning guide as soon as you land on it. You can embed this link anywhere from an email campaign to your LMS.

Linking from a Guided Learning guide to your LMS

There are actually several ways to do this.

Each and every Guided Learning tooltip can embed reach media in it. This can be an image, a video, and depending on your LMS an actual course.

Furthermore, you could add a link to the course page in your LMS inside any Guided Learning tooltip. This means that as the user sees the tip he will be able to click on this link and a new browser tab with your LMS will open.

Both of these options are great ways to bring LMS content into context at the exact time of need.

Exporting training data from Guided Learning into your LMS

One of the key features of any LMS is its ability to track the progress of each and every employee in his/her onboarding process. Often you use the LMS to see which employees did not complete a certain training or course so that you could reach out to them.

Guided Learning keeps track of all your employees and how they interact with your guides. And if you stop to think of it a guide is in itself a micro training entity.

To that end, we allow our customers to export all usage information from the Guided Learning database so that you can upload it into an LMS.

For more information, please create a Service Request on My Oracle Support.

Javascript API

Introduction

The Guided Learning Javascript API (JS API) can be used to customize and personalize your end-users experience. The API makes it is possible to set end-user fields that can be used for setting complex activation rules for your guides. Additionally, you may use the API for personalizing and dynamically set the content of Guided Learning guides. The API also extends the support for single-page applications (SPA), ensuring that customers with SPAs can enjoy the full range of Guided Learning features that non SPAs can.

The API can also be used to attain granular control over the activation of Guided Learning guides within your web application. Using the API you can start or stop guides based on your custom business logic, as well as monitor the progress of a specific customer through your guides, by binding to guide-level or step-level events.

Using the API

The API commands are issued by a call to the Guided Learning method, which is defined on the global scope as part of the Guided Learning JS embed scriptlet. As such, API command can be issued from any JS code that is executed after the embed scriptlet is run. The best practice recommended by Guided Learning is to issue the API commands from a script tag embedded directly after the Guided Learning embed scriptlet.

NOTE: While API commands can be issued before the main Guided Learning JS script is loaded and run, they will only be executed when Guided Learning is fully loaded. The order in which commands were issued is preserved, making the loading and scheduling process transparent to the developer. The single exception to this rule is that the first call to api.fields.set in the queue will be executed before all other API commands when the Guided Learning script is loaded.

The Anatomy of Guided Learning Calls

Calls to the Guided Learning API are of the following generic form:

```
iridize("api.foo.bar", {callData:theData},function optionalCallback(data) {});
```

The first argument is a string indicating the command to be run. For example, "api.guide.start", is the name of the command for starting a guide.

A second argument is a plain object serving as a dictionary of data passed to the API command. For example, when starting a guide with an API name "foobar" using the API the second argument would be of the form {apiName:"foobar"}.

The third argument is a callback function that is run when asynchronous API calls are done or when an API event occurs. This argument is optional and is only executed for API methods that support it, as indicated in the specific method documentation. If data is to be passed to the callback function it is done in the form of a plain object, with the passed data as properties. The specific properties are specified in the documentation of the individual method.

NOTE: The second argument is optional for commands that do not require passing data, but it cannot be omitted if a callback function is supplied as the third argument. In such a case it is recommended to pass an empty plain object, {}, as the second argument.

Customization and Personalization Methods

End-User Field Methods

With Guided Learning, you can set complex activation rules for your guides, based on individual end-user fields. For example, you may wish to automatically start an introductory guide for every new user once. Or you may want to make a guide available for a user two days after she have viewed a different guide. In order to be able to use such rules, you need to set some fields for your end-users, using the API.

api.fields.set

Set fields for the current end-user. This method expects a dictionary of one or more fields as the second argument. For example, the Guided Learning embed code includes the following call (commented out) by default:

```
iridize("api.fields.set", {
    user_id:"USER_ID_GOES_HERE",
    joined_at:UNIXEPOCH_TIMESTAMP
});
```

where we set the reserved-name field "user_id", which can be any string uniquely identifying your end-users, such as a username, user ID, or an email address. Here the field joined_at is a date field, set as a numeric unix epoch timestamp (in seconds) giving the time the user signed-up to the application.

NOTE: You must set the user_id in the very first call to api.fields.set on a page. Subsequent calls may omit this field, but it is considered best practice to always include it.

Setting user group/role: you may want to use this method to customize the experience of users based on group participation or user roles. This information could then be used to set up activation rules, such that members of different groups will have different guides available. An example for such a call could be:

```
iridize("api.fields.set", {
    user_id:"simpsons@springfield.us",
    joined_at:629848800,
    role:"admin"
});
```

Goal Tracking: setting up goal tracking is just a matter of setting a field with the field name prefix "goal_". This information can then be used to track conversions and set up A/B tests for

your guides. For example, if we want to track when a user upgrades her plan from a trial account, we could call:

```
iridize("api.fields.set", {
    user_id:"simpsons@springfield.us",
    joined_at:629848800,
    goal_upgraded_plan:"silver"
});
```

on the upgrade confirmation page, where the goal value string is in this cased used to track the type of plan the user converted to.

Field types

There are three basic javascript field types you can use, which are determined by the field value:

- String
- Integer
- Boolean

For example, we could call:

```
iridize("api.fields.set", {
    user_id:"hpotter",
    joined_at:867618000,
    type:"mage",
    level:3,
    evil:false
});
```

Additionally, there are two types indicated by a prefix/suffix to the field name, goal fields (which are described above, under Goal Tracking), and time fields.

Time fields: to set a time field, simply name the field with an "_at" suffix. The time value should be provided using a unix epoch timestamp (in seconds). For example, to set the date a user joined the application, we could call:

```
iridize("api.fields.set", {
    user_id:"simpsons@springfield.us",
    joined_at:629848800,
});
```

Personalization and Dynamic Content

api.vars.set

Dynamically set page variables which can be used in guides content.

For example, you may want to address your end-users in your guides using their names. In such a case the call to api.vars.set could be:

```
iridize("api.vars.set", {"new user":"F. Bar"});
```

The variable can then be used in any guide that will be running on the page by enclosing the variable name ("new user") within double curly brackets ({{new user}}). When the guide is run and the step is displayed, the part enclosed within curly brackets will be replaced with the value of the page variable if it is set. If it is not set, the variable name will be used instead (without the curly brackets). A default value different than the variable name can be set using the pipe, "|", symbol. For example, the text of a welcoming step could be set to:

Hi {{user_name|there friend}}, thanks for signing up to fooBar. Let us show you around.

With the variable was set as described above, the text of this step will read "Welcome F. Bar, thanks for signing up to fooBar, let us show you around.". If the variable was not set, the text will open with "Hi there friend,..."

Another use case for this API call is for including dynamically generated hyperlinks in your guides text. For example, you may have in your application paths such as "/foo/username/bar/". If you would like to add a hyperlink to such a path within a guide you could simply set the link target to "/foo/{{username}}/bar/", and run "api.vars.set" as follows:

```
iridize("api.vars.set", {"username":"007"});
```

This will yield a hyperlink with the target "/foo/007/bar/".

If the var is not set, but a user field with the same name was set in an api.fields.set call, the value from the user field will be used. If both a var and a field with the same name are set the value of the var will be used.

NOTE: The variable must be set when the guide step is run. However, you may use the below documented "api.guide.onStartStep" event for setting this value immediately prior to the execution of a specific step in a specific guide. This makes it possible to dynamically set the content of a guide step based on the actions of the user in previous steps of the guide.

Single Page Applications (SPA) support

The Guided Learning JS API allows customers with SPAs to easily treat route changes as if they were page url changes. This includes controlling and refreshing the contents of the Guided Learning start panel based on the current route, set guide steps to wait for route change (similar to waiting for page refresh), automatically activating a guide when a specific route is loaded etc.

NOTE: You should only run the Guided Learning embed code, including the call to api.fields.set, once, on the initial page load of the SPA. You should then update the route or refresh the guides on route or state changes. See the next sub-sections for further details.

The Guided Learning API provides both low-level and high-level method calls for setting up Guided Learning for your SPA. The high-level methods provide the simplest way to integrate Guided Learning, transparently differentiating between the first-page load of the SPA, and subsequent route changes. The low-level methods are available for SPAs that require more granular control or setting and updating routes, as well as refreshing the running guides.

Setup using high-level calls

The high-level setup of Guided Learning for a SPA is comprised of two parts. First, we need to tell the Guided Learning JS API to defer running the guides for after the initial SPA route is loaded. Then, we need to update the Guided Learning JS API whenever the SPA route is loaded, including the first SPA route load.

Waiting for the initial SPA route load

When using the high-level JS API calls to integrate Guided Learning into a SPA you should add a call to api.route.wait to or immediately after the Guided Learning embed code, such-

```
iridize("api.route.wait", {});
```

This call notifies the iridize JS API that it should defer loading the iridize guides for after the first SPA route is loaded and a call to api.route.update is made (see below).

Handling SPA route changes

Whenever a SPA route is loaded (including the initial SPA route load), you should make a call to api.route.update, passing the new route that is loaded as part of the parameter object, such-

```
iridize("api.route.update", {"route":"/foo/"});
```

Where "/foo/" is the SPA route that is loaded. That's it. The api.route.update call knows to handle the initial route load differently than route changes transparently and will issue a call to api.guide.refresh (see below) only when the SPA route is actually changed..

Setting user fields when integrating iridize into a SPA

The details of using the api.fields.set calls are provided in the "Customization and Personalization Methods" section above. Please note that if you are not using the api.fields.set call in your iridize implementation you can skip this section. When integrating iridize with a SPA there are two general cases that need to be considered. The simplest case is when the user details are available when the iridize embed code is run. The slightly more involved case is when the user details (such as user_id, joined_at date, account_id etc.) are unavailable when the iridize embed code is run.

User details available when the iridize embed code is run

If the user details are available when the iridize embed code is run, the usage of api.fields.set is identical to that of non-SPA setup, regardless of whether the embed code is embedded directly into the page's HTML or is only run when the SPA is loaded. In this case, the call to api.fields.set should be made as part of or immediately after the iridize embed code, within the same synchronous code flow. An important detail to note is that api.fields.set should only be called once and you should not issue a call for every time the SPA route is changed.

User details only available after the SPA is loaded

It is usual for SPAs that the user details are not available when the iridize embed code is run. This usually is the case when the iridize embed code is embedded as part of the page HTML, but the user details are only fetched at a later time, as the SPA loads. In such a case you should add a call to api.fields.set with an empty fields object as the second parameter as part of or immediately after the iridize embed code (within the same synchronous code flow), such-

```
iridize("api.fields.set", {});
```

Then, once the user details are available you should run the api.fields.set call with the user details as usual. With this setup the iridize JS API will know to wait with the guides initiation for after the user details are provided with the second call to api.fields.set. Please note you should only call api.fields.set with the user details once, and not repeat it every time the SPA route is changed. Setup using low-level calls

Initial Page Load - api.route.set

When integrating iridize to a SPA, you should add a call to "api.route.set" to the initial iridize embed code. The call expects a single property, "route", in the second argument. For example, if the initial route for your application is "/foo/" the call will be:

```
iridize("api.route.set", {"route":"/foo/"});
```

NOTE: if you use hash-based navigation in your SPA you may include the hash sign in the route or not as you prefer. Leading and trailing "/" are also optional.

DEPRECATED: This API call is no longer needed when embedding in a SPA.

Force a page refresh - api.guide.refresh

Unlike regular web applications, SPA's do not have page loads as natural points for refreshing the state of your guides. With the use of the "api.guide.refresh" API call you can refresh the state on demand, optionally updating the current route.

For example, if the route was changed to "/foo/bar/" the call would be:

```
iridize("api.guide.refresh", {"route":"/foo/bar/"});
```

NOTE: This is a low-level API call, in most situations you should use "api.route.update".

The Guide API Methods

API methods that have to do with guides are all namespaced as "api.guide.methodName".

Control Methods:

api.guide.list

Get the list of all available guides for the current domain.

For example, to fetch the guides list and print it to the browser console, you could do:

```
iridize("api.guide.list",  {}, function(data)  {
    var guidesList,guide,i;
    // get the array of guide information objects
    guidesList = data.guides;
    // print the guides to browser console
    for (i = 0; i < guidesList.length; i++)  {
        guide = guidesList[i];
        console.log(guide.apiName + " | " + guide.displayName);
    }
});
```

api.guide.start

Starts a specific guide. The method expects the API name of the guide as the "apiName" property of the second argument. For example, to start a guide with an apiName "foobar" the call would be:

```
iridize("api.guide.start",  {apiName : "foobar"});
```

or, you can also provide the step id to start from:

```
iridize("api.guide.start",  {apiName : "foobar", step:"5"});
```

This method also accepts an additional optional "force" property to the second argument. If this property is set to false, the API method will not force start the guide, as is its default behavior.

NOTE: The apiName of guides can be found and modified under that advanced tab of the Guide options page in the iridize dashboard.

NOTE: The force:false option must be set in order for end-user activation rules to apply for guides started using the API.

api.guide.stop

Stops a specific guide if an apiName is supplied, or all currently running guides if not .

To stop a specific guide if it is running, pass the api name of the guide as the apiName property of the second argument. For example, to stop a guide with an apiName "foobar" the call would be:

```
iridize("api.guide.stop",  {apiName : "foobar"});
```

To stop all currently running guides, simply don't pass an apiName property, so the call would be:

```
iridize("api.guide.stop", {});
```

NOTE: The apiName of guides can be found and modified under that advanced tab of the Guide options page in the iridize dashboard.

api.guide.disable

Stops all running guides and disables the execution of any guides.

```
iridize("api.guide.disable",  {});
```

NOTE: You can still run a guide using the api.guide.start API call.

api.guide.enable

Enables Iridize and relaunches all guides as if the page was loaded again. Call this function following a call to api.guide.disable.

```
iridize("api.guide.enable",  {});
```

Events:

api.onReady

This event is fired after Iridize has been fully loaded.

```
iridize("api.onReady",  {}, function(data)  {
    console.log("iridize is ready");
});
```

api.guide.onStartGuide

The startGuide event is triggered every time a guide is started. The callback is passed two properties - the apiName of the guide for which the event was triggered and a parameter indicating the type of guide activation. The main types of activation values of the runType variable are “autoload”,”startPanel” and ”apiStart”.

For example, the following call would log every guide started to the browser console:

```
iridize("api.guide.onStartGuide",  {}, function(data)  {
    var apiName = data.apiName;
    var runType = data.runType;
    // print the guide’s api name and the activation type
    console.log(apiName + " | "+ runType);
});
```

api.guide.onStartStep

The startStep event is triggered every time the end-user progresses along with any guide. The callback is passed two properties - the apiName of the guide for which the event was triggered

and the id of the step which the user has reached (this id is usually the step number in the guide).

For example, the following call would log every step started to the browser console:

```
iridize("api.guide.onStartStep",  {}, function(data)  {
    var apiName = data.apiName;
    var step = data.step;
    // print the guide's api name and the id of the step reached
     console.log(apiName + " | "+ step);
});
```

api.guide.onStopGuide

The stopGuide event is triggered every time a guide is stopped, either by the user reaching the guide's end or due to the user clicking the close button (x) on the tip. The callback is passed the apiName of the guide which was stopped.

NOTE: data.userClosed will be true if the user has manually closed the guide. If 'remind me later' was clicked it will be false.

For example, the following call would log every guide when it is stopped:

```
iridize("api.guide.onStopGuide",  {}, function(data)  {
        var apiName = data.apiName;
    // print the api name of the guide that was stopped

      console.log(apiName);
    });
```

api.guide.onErrorStep

The errorStep event is triggered every time iridize is unable to display a tooltip (e.g. if the element that the tooltip should point at is not found). The callback is passed as a data object which contains information about the step and the guide that had the error.

For example, the following call would log the guide and the step in error:

```
iridize("api.guide.onErrorStep", {}, function(data) {
    // print the guide's api name and the id of the step.
    console.log(data.apiName + " error on step "+ data.stepId);
});
```

api.guide.onItemsFiltered

The onItemsFiltered event is triggered every time the user interacts with the search inside the Iridize help widget. You can use it to incorporate the list of Iridize guides into another container in your application.

```
iridize("api.guide.onItemsFiltered", {}, function(data) {
    //your code goes here
});
```

Your callback function will be called with the following data object

query: the text that the user searched for

step_id: the id of the step in the guide that caused the event

page_items: the list of content items that match the current page as well as the search criteria

filtered_items: the list of content items that match the search criteria. content not for the current page will also be available here.

entire_items: the list of all items available in the help widget, ignoring the search term.

Working with OGL in the Chrome DevTools Console

- Please refer to this article.

Oracle Guided Learning Setup Instructions

Setup for Fusion Applications

- Enabling OGL - in Fusion Applications
- Configuring OGL - Creating a Sandbox
- Configuring OGL - Setting Roles and Scripts

Setup for Non-Fusion Applications

- Create a support ticket on My Oracle Support.

Compatibility

Guided Learning on-page guidance solution is compatible with any application that can be used with a standard web browser.

Guided Learning technology can offer instructions on any element in the DOM tree.

Guided Learning can be used to provide instructions on embedded objects such as java applets, Flash, and Silverlight. However, Guided Learning cannot inspect elements within these embedded objects.

Setup Oracle Guided Learning in a Single Page Application (SPA)

Before we dive into the steps required to embed Oracle Guided Learning in a SPA, let's explain why a specific setup for the SPA's is needed in the first place.

The thing that's different in a SPA is the fact that even though the browser does not perform any standard navigation the view is still changing. As a result, the application codebase needs to update Oracle Guided Learning that the view has changed so that Oracle Guided Learning could refresh all of the content.

1. Basic embed code

To set up Oracle Guided Learning just as in any other web application by placing the Oracle Guided Learning embed code in the relevant pages. Your specific embed code can be found on your account setup page and should simply be copied before the closing </body> tag.

The code will look something like this:

```
/* Iridize.com*/window.iridize=window.iridize||function(e,t,n){return
iridize.api.call(e,t,n);};iridize.api=iridize.api||{q:[],call:function(e,t,n)
{iridize.api.q.push({method:e,data:t,callback:n});}};

iridize.appId="XXXXXXXXXXXXXXXXXXX";

iridize.env="prod";
```

```
iridize("api.fields.set",{user_id:"USER_ID_GOES_HERE"});

(function(){var e=document.createElement("script");var
t=document.getElementsByTagName("script")
[0];e.src=("https:"==document.location.protocol?"https:":"http:")+"//
d2p93rcsj9dwm5.cloudfront.net/player/latest/static/js/iridizeLoader.min.js";e.type="text/
javascript";e.async=true;t.parentNode.insertBefore(e,t)})();
```

2. Notify Oracle Guided Learning that this is an SPA

```
iridize("api.route.wait", {});
```

This call notifies the Oracle Guided Learning JS API that it should defer loading the Oracle Guided Learning guides for after the first SPA route is loaded.

3. User Identification (part 1)

Oracle Guided Learning needs to know which user is currently logged in so that it could provide that user with relevant content.

If you do not know who the logged in user when the code of the basic embed code (step #1) is executed. You can notify Oracle Guided Learning that you will be providing this information later.

To do so, you should call the 'api.fields.set' function with an empty object.

```
iridize("api.fields.set",{})
```

NOTE: User conditions will be disabled if you do not make this notification.

Your embed code should now look like this:

```
/* Iridize.com*/window.iridize=window.iridize||function(e,t,n){return
iridize.api.call(e,t,n);};iridize.api=iridize.api||{q:[],call:function(e,t,n)
{iridize.api.q.push({method:e,data:t,callback:n});}};

iridize.appId="XXXXXXXXXXXXXXXXXXX";

iridize.env="prod";

iridize("api.route.wait", {});

iridize("api.fields.set",{})
```

--- OR ---

```
iridize("api.fields.set",{user_id:"USER_ID_GOES_HERE"});

(function(){var e=document.createElement("script");var
t=document.getElementsByTagName("script")
[0];e.src=("https:"==document.location.protocol?"https:":"http:")+"//
d2p93rcsj9dwm5.cloudfront.net/player/latest/static/js/iridizeLoader.min.js";e.type="text/
javascript";e.async=true;t.parentNode.insertBefore(e,t)})();
```

4. User Identification (part 2)

If you have already provided user identification in the basic embed code you can skip this step.

If you used Oracle Guided Learning("api.fields.set",{}) in the previous step, you will need to update Oracle Guided Learning of the logged in users as soon as possible. To do so call:

```
iridize("api.fields.set",{user_id:"USER_ID_GOES_HERE"});
```

NOTE: You can provide Oracle Guided Learning other user data (e.g. user_role, location, etc.) by adding more items to the passed javascript dictionary.

5. Update route

Whenever an SPA route is loaded (including the initial SPA route load) you need to notify Oracle Guided Learning of it by calling:

```
iridize("api.route.update", {"route":"/foo/"});
```

Where "/foo/" is the SPA route that is loaded. The api.route.update call knows to handle the initial route load differently than route changes transparently. It will also automatically refresh the content based on the new route.

The call to "api.route.update" must be made after the route has successfully changed and not before. This way Oracle Guided Learning will be able to update the content according to the newly loaded route.

When using the AngularJS's $route for view routing you can bind on the $routeChangeSuccess event. When using ui-router you can bind on the analogous $stateChangeSuccess event. Both events provide the new route/state which can be passed to the api.route.update call.

BEST PRACTICE: It is recommended to omit random ID's from the route. For example, if you run a CRM and each contact has a unique (e.g. https://myapp.com/dashboard#contact/169627) where 169627 is the ID of the contact in your database. Instead of setting the route to '/contact/169627' it is recommended to make the route more generic. For example:

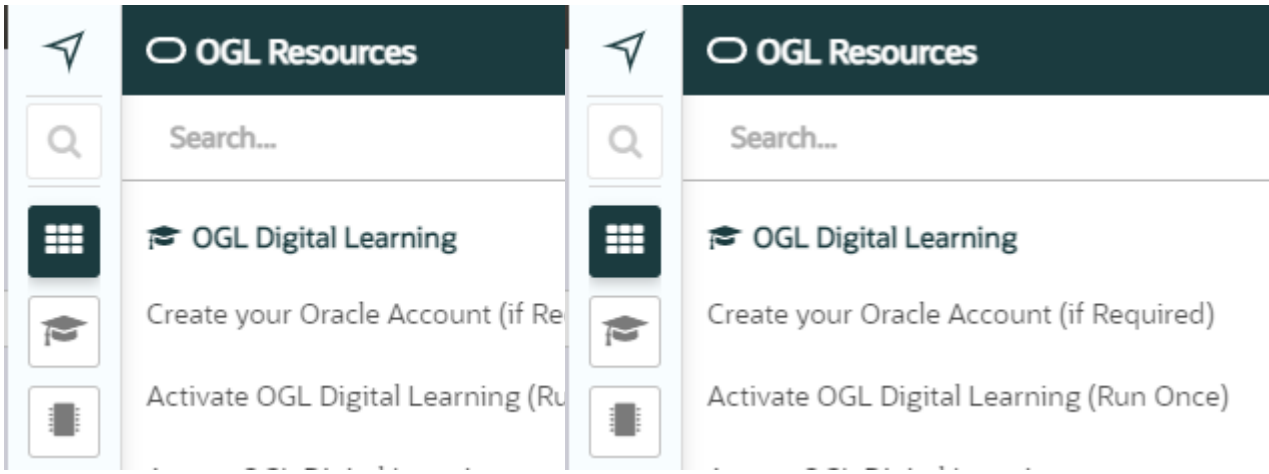
```
iridize("api.route.update", {"route":"/contact/view"});
```

IMPORTANT: The people who will be writing the content using the Oracle Guided Learning platform need to know what scheme was used to map the urls to routes in Oracle Guided Learning. This is needed when creating guide activation conditions.

Step and Video Guide Enabling/Disabling

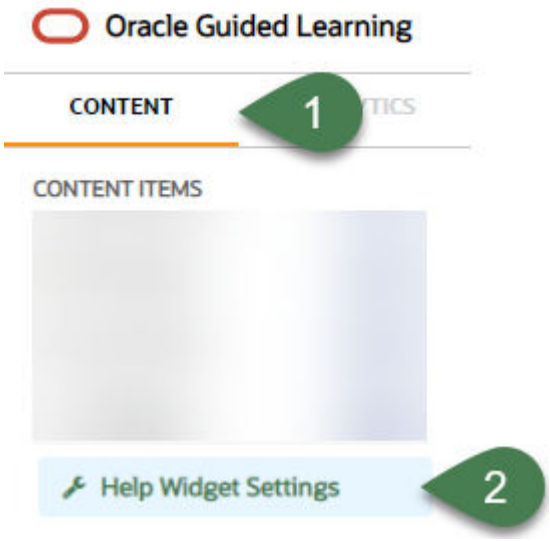
Step Guides are an in-app presentation of the guide's pdf, showing the user the entire process in question without having to leave the application. Users can access Step Guides by selecting the **Steps** icon next to the guide name. This also provides access to the Video Guide.

Step Guides are ENABLED (The Steps icon is displayed)	Step Guides are DISABLED (The Steps icon is *not* displayed)

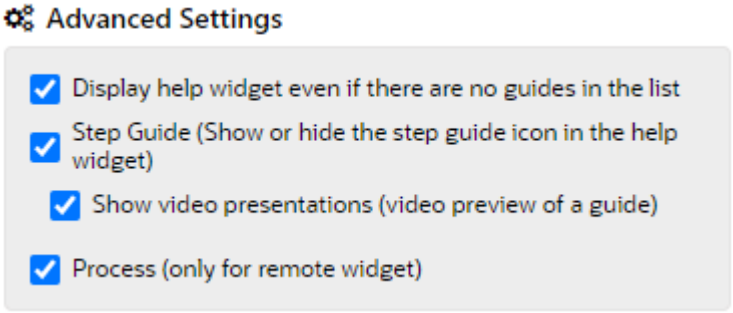


IMPORTANT: Enabling/Disabling Step and Video guides is set at the account/appID level, not on a per-guide basis.

1. Select the **Content** tab in the **OGL Console**, then select **Help Widget Settings**



2. Locate the **Advanced Settings** section



3. Check or uncheck the '**Step Guide (Show or hide the step guide icon in the help widget)**' checkbox (Note: Checked = Enabled; Unchecked = Disabled)
4. Check or uncheck the '**Show video presentations (video preview of a guide)**' checkbox (Note: Checked = Enabled; Unchecked = Disabled)
   IMPORTANT: *The 'Show video presentations (video preview of a guide)' setting is automatically unchecked when the 'Step Guide (Show or hide the step guide icon in the help widget)' setting is unchecked.*
5. Select the **Save to Dev** button to validate in your DEV environment and, when ready, select the **Publish to Prod** to save the changes to your production environment

What is the difference between Viewer, Editor, and Owner?

Oracle Guided Learning allows you to collaborate when creating content.

To that end we support three user roles: viewer, editor, owner.

Below is a list of permission for each role. note that an editor has all the permission of a viewer and an owner has all the permission of an editor.

Viewer

1. view user analytics
2. view guide analytics
3. preview a guide

Editor

All Viewer permissions and

1. create/edit/delete guide
2. publish guide

Owner

All Editor permissions and:

1. change help widget
2. change theme
3. change application settings (e.g. enable guide feedback)
4. reset user data
5. add/edit/remove users from account

How to Install, Extract, and Package the OGL Editor Extension from the Chrome Web Store

This article covers the following procedures:

- Installing the OGL editor extension for Google Chrome.
- Extracting the OGL Editor Extension.
- Packing the extension CRX files.

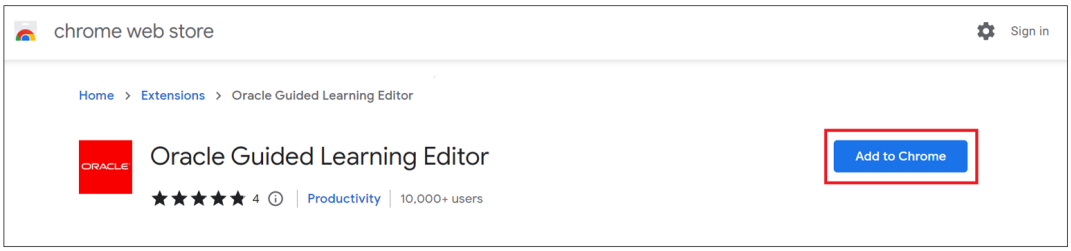
Installing the OGL editor extension for Google Chrome

Prerequisites

- Google Chrome browser.

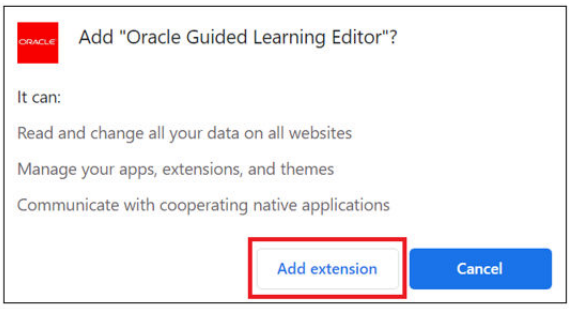
Instructions

1. Go to the **Chrome Web Store**. The page below opens; select **Add to Chrome**.

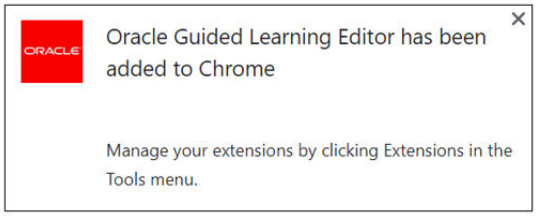


   A confirmation dialog box opens.

2. Select **Add extension**.



3. A confirmation message is displayed once the setup is completed.



Extracting the OGL Editor Extension

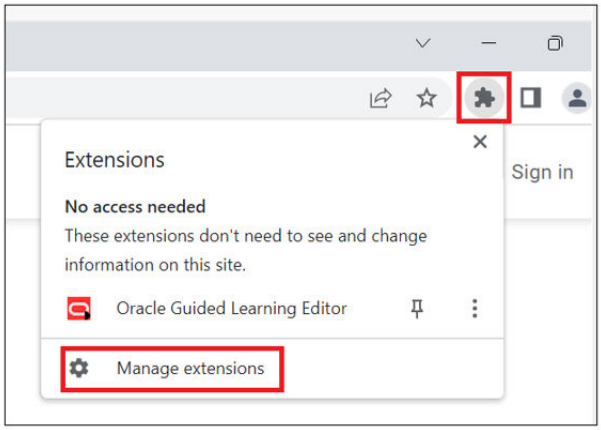
Prerequisites

- Google Chrome browser.
- Oracle Guided Learning (Iridize) extension added to Chrome.

Instructions

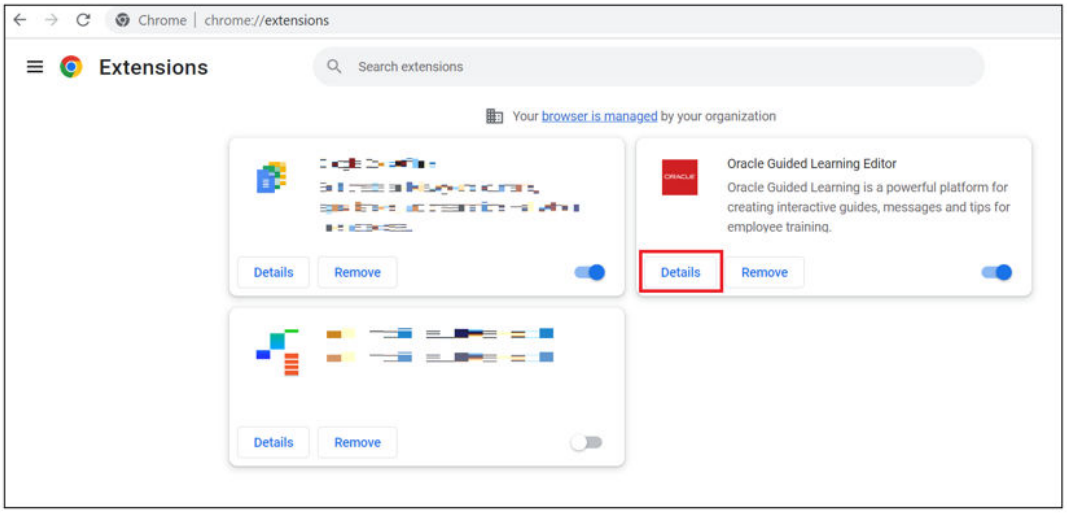
1. Open **the Google Chrome** browser.

2. At the top right, select the **Extensions** icon( )and then select **Manage extensions**.



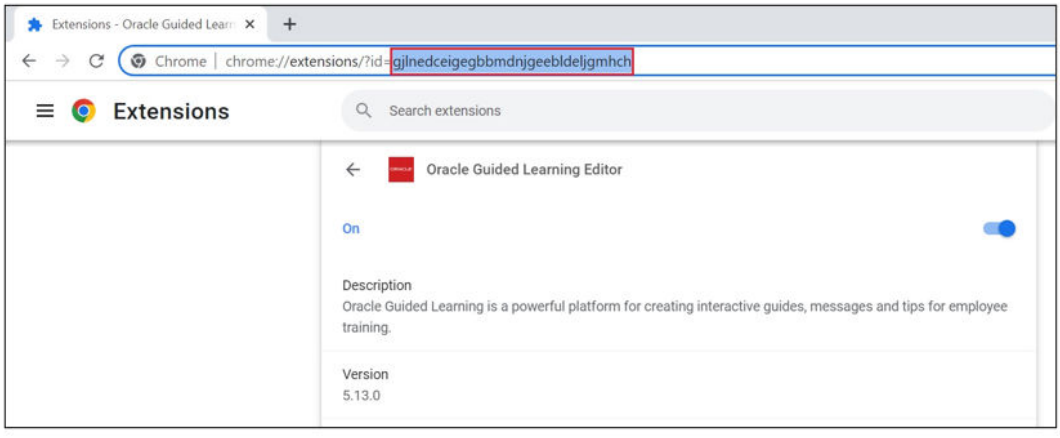
The **Manage extensions** window appears with the list of extensions installed on the Chrome browser.

3. On the **Oracle Guided Learning Editor** extension card, select **Details**.



The details of the OGL Editor extension are displayed now.

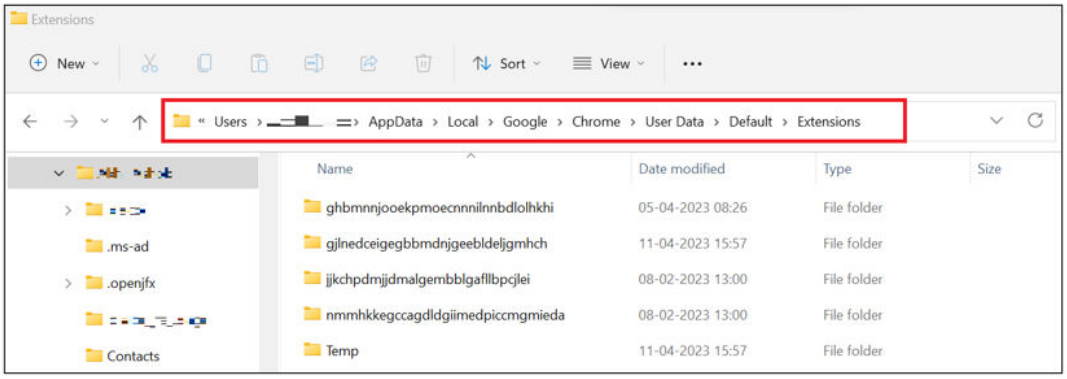
4. Look at the URL. The **Extension ID** is the long string of characters at the end of the URL.



5. Identify the **Extension ID** and copy it.

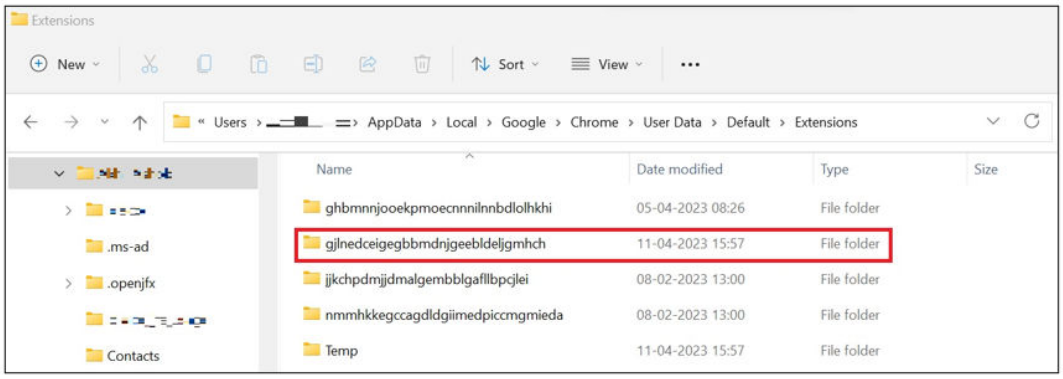
6. On your PC's local drive, navigate to the extensions folder, where all the extensions are stored.
   Your default directory may look like this:

   C:\Users\<*Your_User_Name*>\AppData\Local\Google\Chrome\User Data\Default\Extensions



> **Note:**
>
> You might have to enable visibility for hidden files.

7. Find the folder matching the OGL Editor's **Extension ID.** Then copy and paste it to a new directory *(i.e., C:\Users\<Your_User_Name>\Documents\OGL Editor extension\)*.



You have now successfully extracted the OGL Editor extension files. To create a **.crx** file, please see the next section.

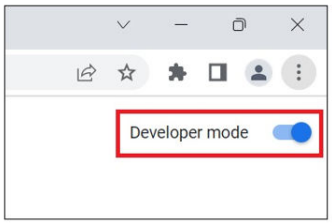
Packaging The Extension CRX Files

Prerequisites

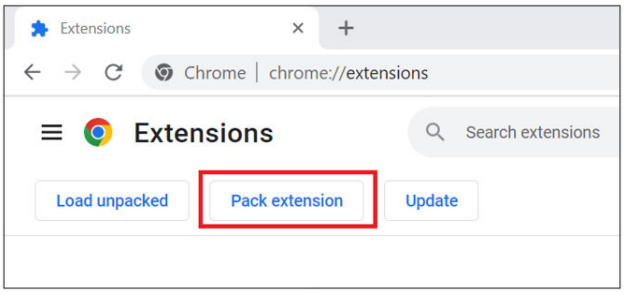
- Google Chrome browser.
- An extracted OGL Editor extension.

Instructions

1. Open **the Google Chrome** browser.
2. At the top right, select the **Extensions** icon( )and then select **Manage extensions**
3. Use the toggle button on the **Manage extensions** window to enable the **Developer mode**.

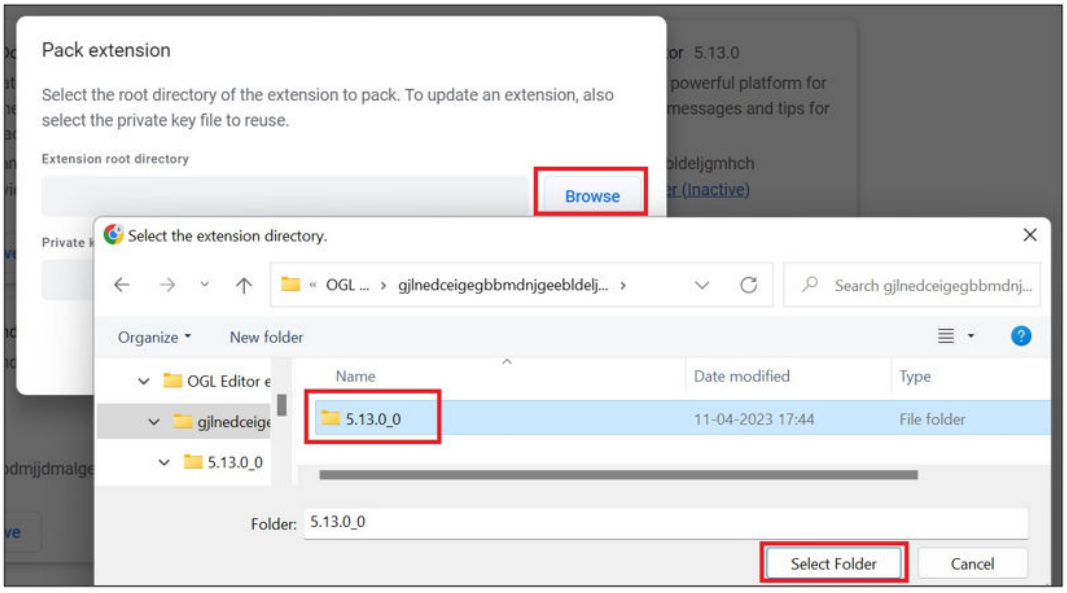


4. Select **Pack extension**.

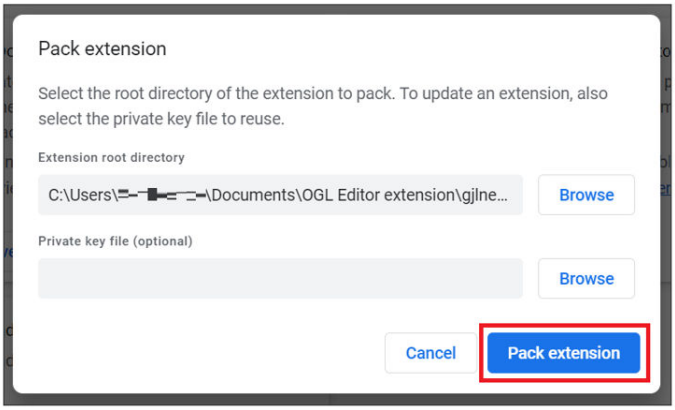


The **Pack extension** modal window is displayed.

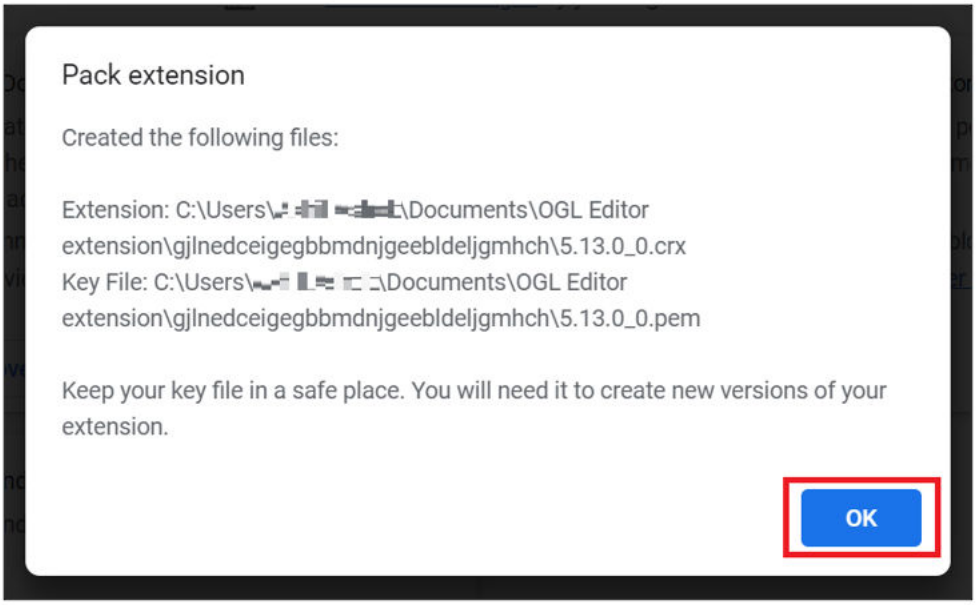
5. Browse the **Extension root directory** and select the folder in which the OGL Editor extension is present.
Your root directory should be similar to **C:\Users\<*Your_User_Name*>\Documents\OGL Editor extension\gjlnedceigegbbmdnjgeebldeljgmhch\5.13.0_0**



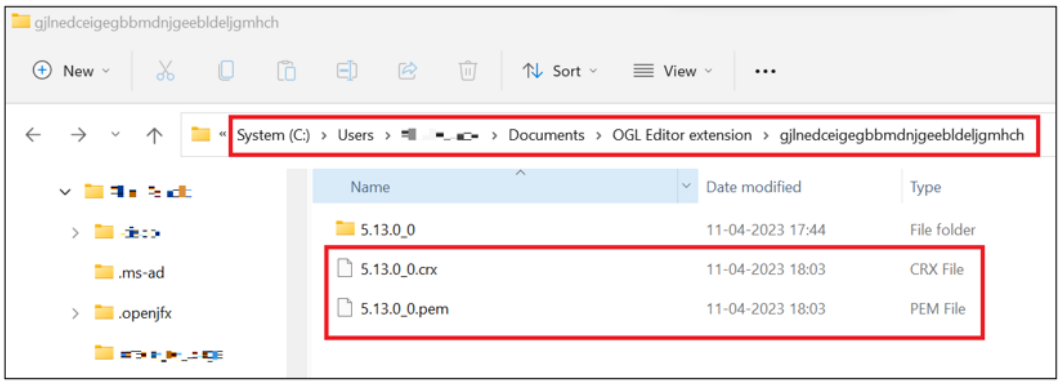
6. Select **Pack extension**.



A confirmation message is displayed. Select **OK** to close the confirmation window.



7. On your PC's local drive, navigate to the extension directory to verify.



The **.crx** file has now been successfully created.

2

Analytics

This chapter contains OGL Analytics knowledge articles.

Guide Report

Purpose:

See an overview of all your content.

Details and Orientation:

Each record provides the details for a specific guide.

The report provides the following information:

- meta data about the guide like the name
- aggregated information like the engagement grade of the guide
- engagement data for the guide like how many users completed it

Legend:

Below is a listing of all the columns in the report and their description.

Api Name:

A unique identifier for the guide.

Title:

The 'name' of the guide. This is the name you see in the dashboard as well as in the help widget.

Engagement Grade:

The number of times that the guide was engaged with divided by the number of times it was viewed.

Api Name:

A unique identifier for the guide.

Title:

The 'name' of the guide. This is the name you see in the dashboard as well as in the help widget.

Completion Grade:

The number of times that the guide was completed divided by the number of times it was viewed.

Last viewed:

When was the last time a user viewed this guide.

Last advanced:

When was the last time a user engaged this guide.

Last completed:

When was the last time a user completed this guide.

Guide Views:

The number of times the guide was viewed.

Guide Engagements:

The number of times the guide was engaged.

Guide Completions:

The number of times the guide was completed.

Guide Closes:

The number of times the guide was closed.

Unique User Views:

The number of unique users that viewed the guide.

Unique User Engagements:

The number of unique users that engaged the guide.

Unique User Completions:

The number of unique users that completed the guide.

Started from Help Widget:

The number of times the guides was started from help widget.

Started Automatically:

The number of times the guides was started automatically.

Started from Permalink:

The number of times the guides was started using the permalink.

Started from API:

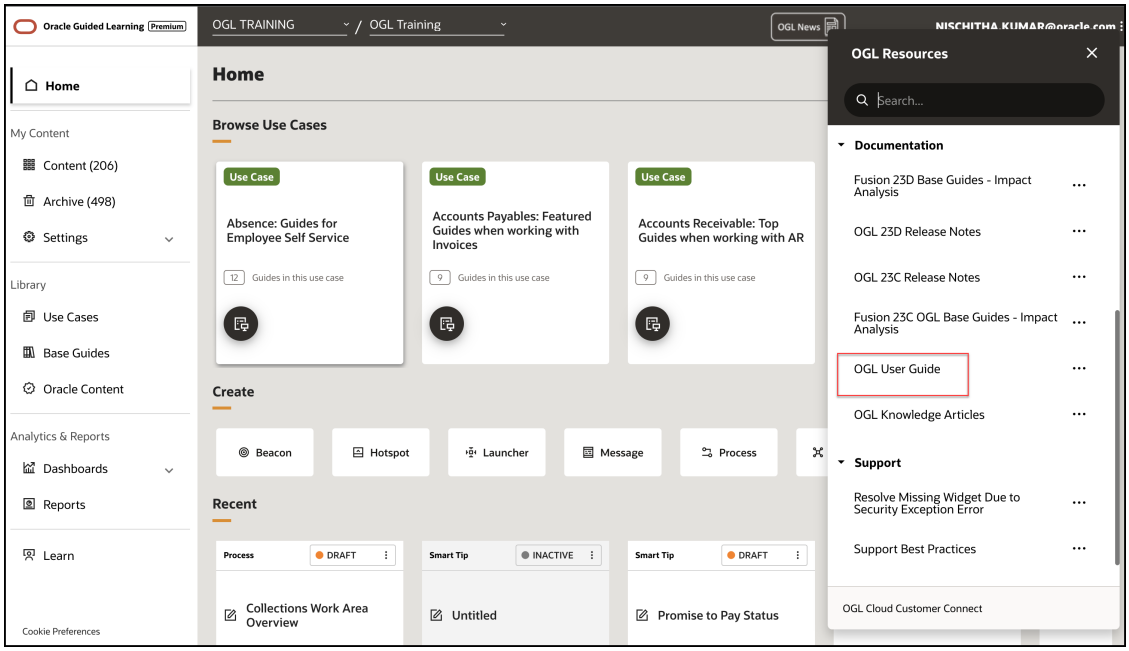
The number of times the guides was started using Javascript API.

Started from another Guide:

The number of times the guides was started from another guide (e.g. branches).

OGL ANALYTICS

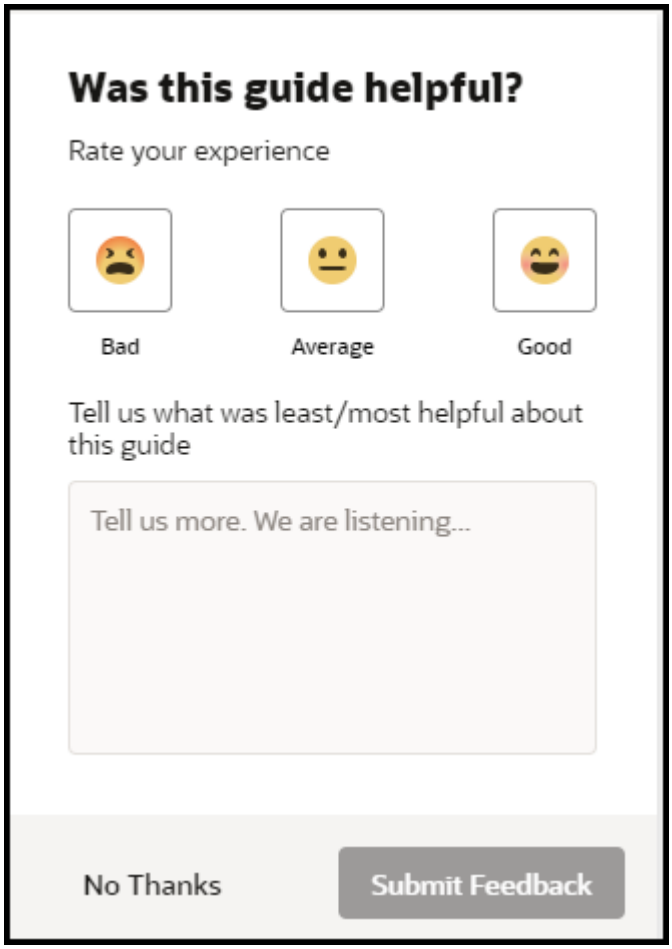
Please refer to the OGL User Guide for a detailed breakdown of #unique_27 , available from the Help Widget in the OGL console.



User Feedback

Guided Learning allows you to collect user feedback about the guides that you created.

With this feature enabled, whenever a user completes a guide or closes it in the middle a feedback widget like the one below will be displayed allowing the user to rate the experience and provide feedback.

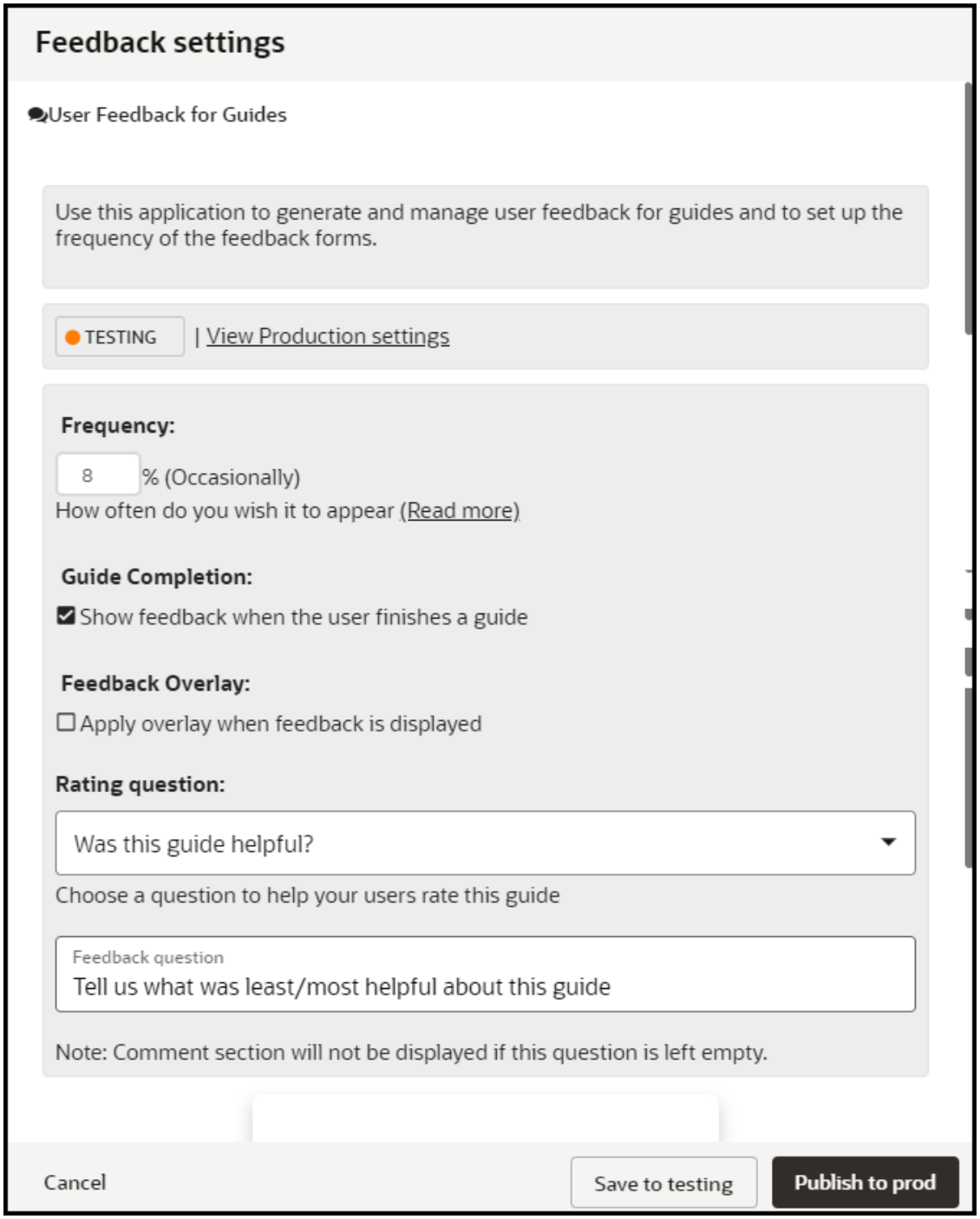


In order to balance between your need to get as much feedback as possible from your users with not wanting to display the feedback widget too often thus affecting the user experience we have put in place two **safety precautions**:

1. The feedback widget will never display twice for the same guide
2. The feedback widget will never display more than once in 10 minutes. So if your user is running 3 guides within a 10 minute time frame s/he will only see the widget after the first guide.

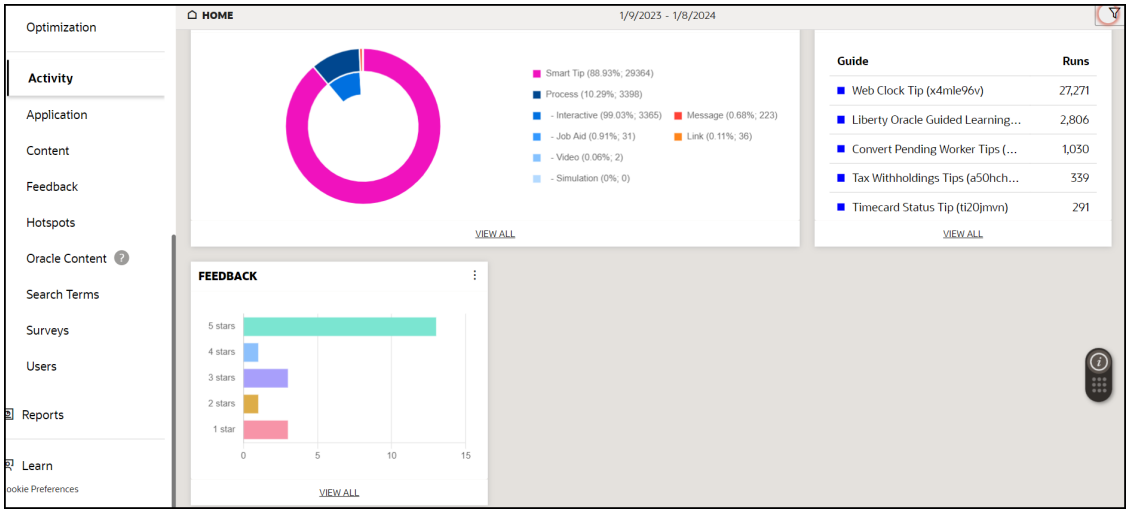
Setup - Enabling User Feedback

1. Open the Feedback link on the left side of your OGL Console under the Settings option.
2. Set your parameters and save your settings. You can enable the Feedback in your dev or test instance before enabling it in Production



Analyzing the Feedback

Guided Learning collects all feedback and attaches it to both the relevant guide and the user who provided it.

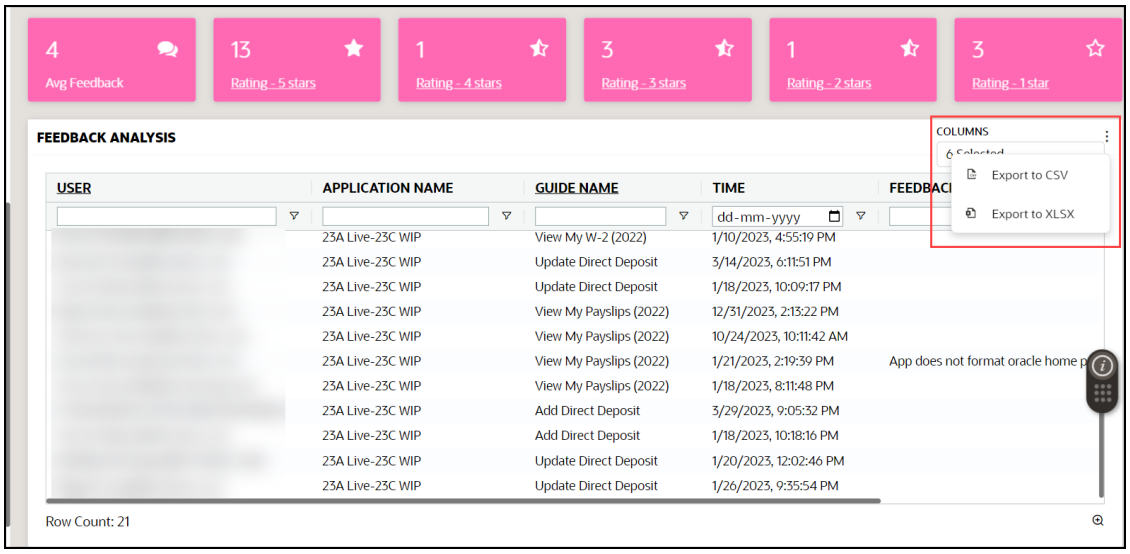


The image above shows a visual representation of the feedback provided by the users.

Reports

You can download complete reports customized to meet your unique specifications.

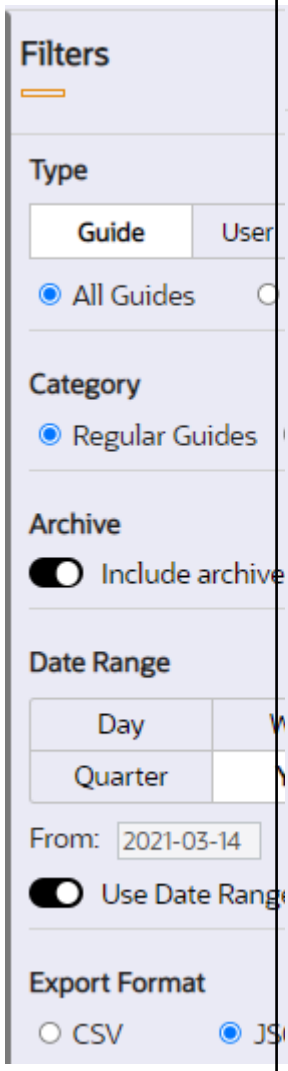
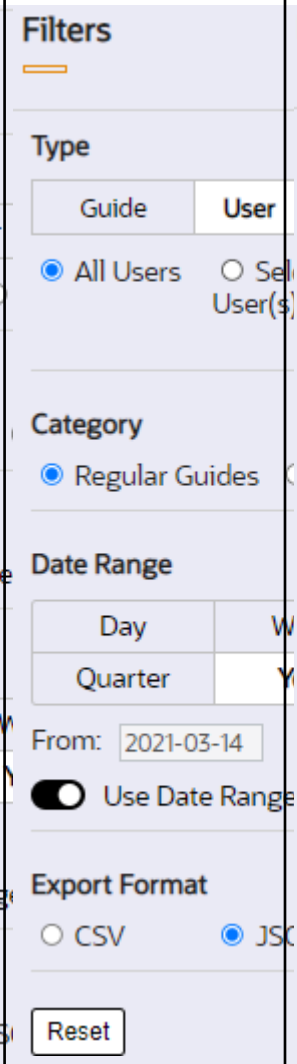
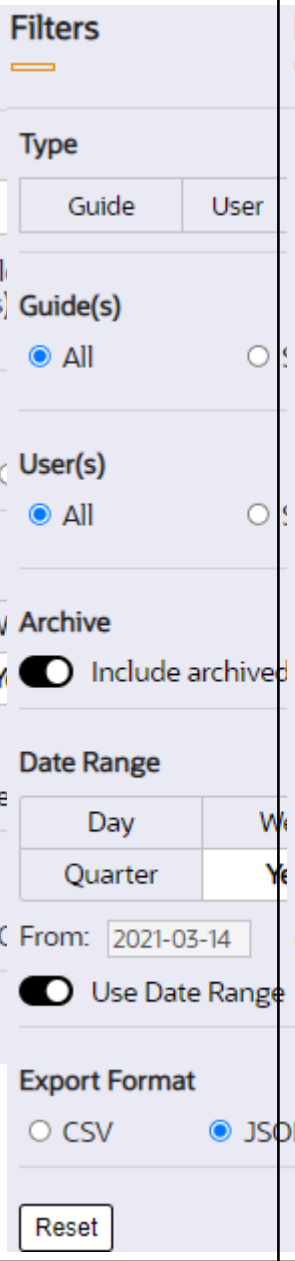
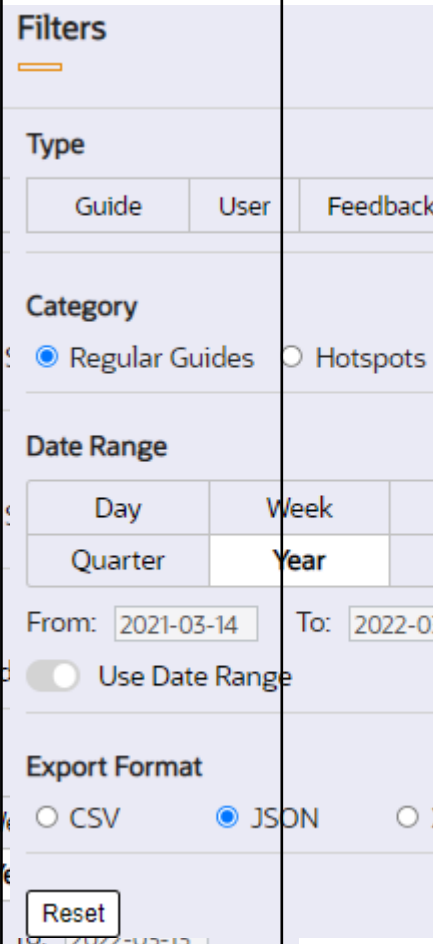
Reports are accessed from the three dot ellipses at the top of each page:



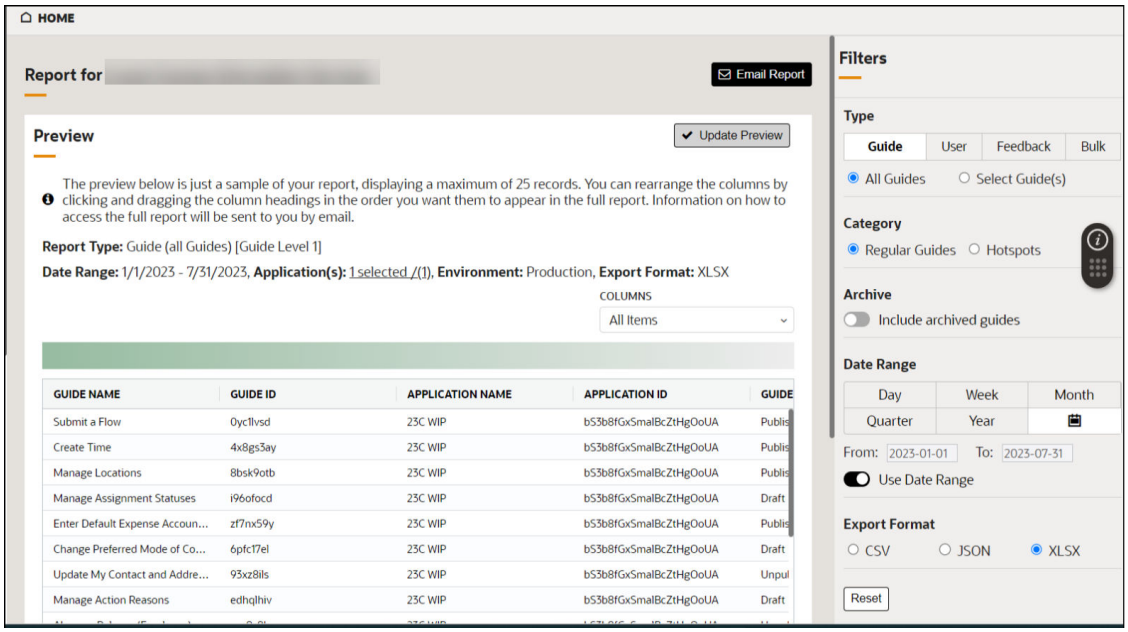
Reports Types

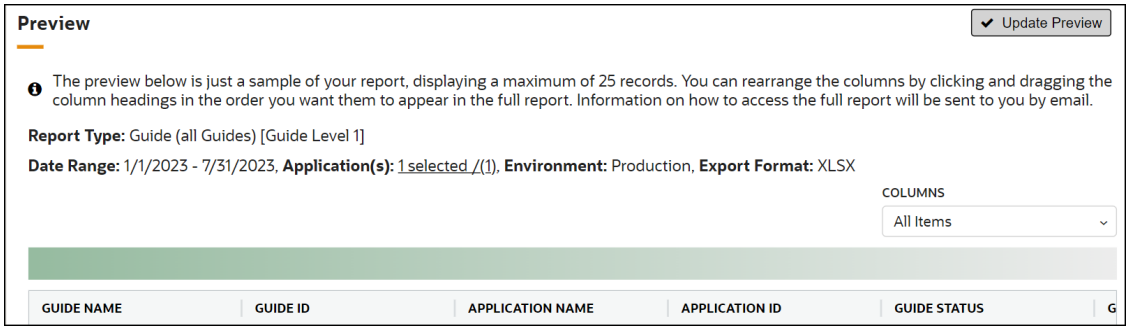
There are 4 main types of Reports that can be downloaded:

- **Guide**
- **User**
- **Feedback**
- **Bulk**

Guide	User	Feedback	Bulk
			
Notes: • When a date range is used, the report displays "active/engaged" data in the selected date range, if it is not used, then it displays "active/engaged" data of all times. • "Select Guide(s)" and "Select User(s)" dropdowns in the sidebar always display "total" data, so you can search amongst all existing guides/users. • The maximum number of guides/users listed in the sidebar's dropdown is 100. You can search for additional guides/users by (partial) name. Also note, that a maximum of 100 guides/users can be selected. If you need to work with more, please select the 'All' option, then download it and filter it as needed.			Note: • Bulk data is provided in a gzipped file. Processing the request might take a longer time. The browser will display a notification when the download is complete. If you leave the page, then the request will be canceled.

Once the options are selected, preview the data set by clicking on the **Update Preview** button in the main section of the window:





A summary of the criteria used for the report is presented above the preview table:

Select the **Email Report** button to request the required data report to be sent to your email address.

3
Console

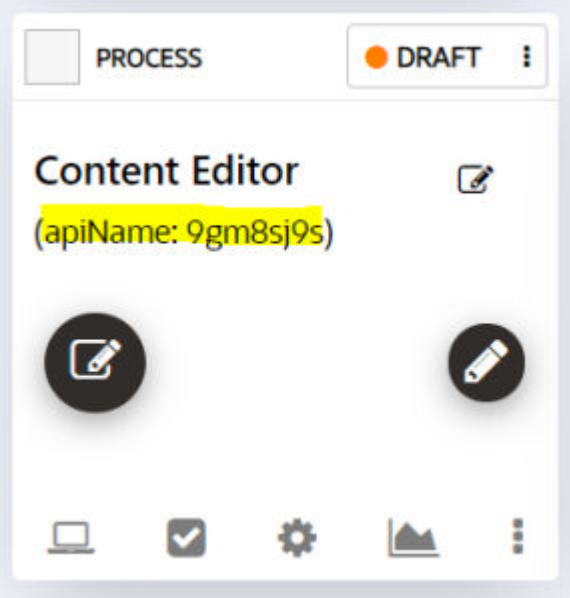
This chapter contains OGL Console knowledge articles.

What is an API Name?

Every Oracle Guided Learning guide has a unique identifier. These identifiers are called the API names. The **API name** of a guide can be seen on the Guides page, right below the guide's name.

The API name can be used to:

- Launch or stop the guide using Javascript API
- Create a permalink



Timestamps in OGL

This table provides a list of OGL attributes and their respective time zones.

Description	Timezone of timestamp
OGL Analytics Data/Reports	UTC
OGL Item attribute: Last Updated By	User's local time
OGL Comments	User's local time
OGL Guide Activation Conditions	User's local time

Timestamp standardized format is: "**MM/dd/yyyy hh:mm a**"

(e.g. **09/10/2021 11:35 AM**)

How long does it take for OGL updates to reflect in the host application?

Updates made in the OGL reflect on the host application at different intervals. The below table details the time it usually takes for updates to reflect:

> **Tip:**
>
> Maximum time limit for updates to take place should be 90 minutes (after 90 mins, when all Akamai edge servers' cache should expire).

> **Note:**
>
> OGL Analytics data will be captured only by the production environment, and the analytics data will be updated every day at 4:30 AM UTC/ 10 AM IST.

Response Time for New Guides

Area/Description	Time	
	Production Environment	**Development Environment**
Display Group Manager	5 - 10 minutes.	Immediately following a refresh.
Publishing an OGL Item	Immediately following a refresh.	Immediately following a refresh.
Publishing the Theme	Immediately following a refresh.	Immediately following a refresh.
All other actions	Immediately following a refresh.	Immediately following a refresh.
Activate/Inactivate the guides	5 minutes.	<1 minute.
Guide Activation Condition	5 minutes.	<1 minute.
Publish/Un-publish Guides	<5 minutes.	<1 minute.

Response Time for Existing Guides

Area/Description	Time	
	Production Environment	**Development Environment**
Display Group Manager	5 - 10 minutes.	0 - 20 minutes.
Publishing an OGL Item	5 - 10 minutes.	Immediately following a refresh.
Publishing the Theme	5 - 10 minutes.	Immediately following a refresh.
All other actions	Immediately following a refresh.	Immediately following a refresh.

4
Creating Content

This chapter contains OGL content creation knowledge articles.

Add a Background Image to a Tip

You can add a background image to a tip by using some CSS tricks.

1. Open the source editor in order to change the raw HTML of the tooltip.
2. Surround the part that you want to add a background to with a tag.
3. add the following style attribute to the div tag

```
style="background-image: url(image.jpg); background-size: cover;"
```

See example below.

```
<div style="background-image: url(//d2p93rcsj9dwm5.cloudfront.net/static/
tipcms/img/LOGO.png); background-size:cover;"><p>This text will have a
background image</p></div>
```

Advance from Multiple Actions

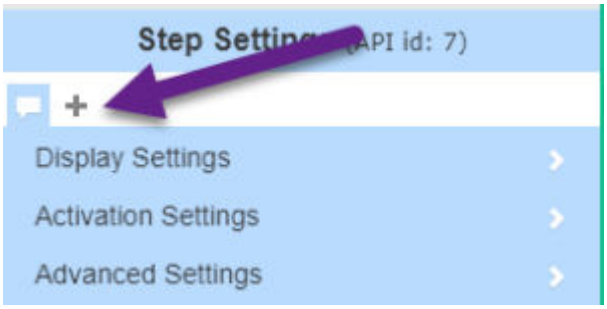
In some cases you need to advance a step two ways. For example, when searching a user can both click Search or press Enter on their keyboard. In some cases, this is required for guides to pass SOX and legal compliance.

To account for both types of interaction, you can add an invisible step that listens for the second way to advance the guide. These instructions describe how to manage a situation where you need to add an invisible step that listens for the user to press enter. However, this same approach can be applied to other interactions. Instead of creating a splash tip as these instructions state, you'd actually tie the invisible tip to an element.

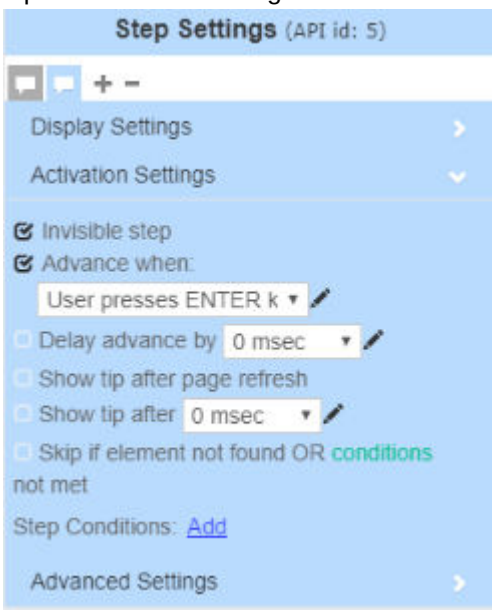
Step-by-Step Guide

To add an invisible tip that advances when a user presses Enter:

1. Go to the step where you need to add the second interaction, and then click the wrench icon to open Step Settings.
2. Click the + icon to add another tip to the step.



3. Click Add Splash.
4. Open Activation Settings and make the following changes:



- – Select **Invisible Step**.
  – Select **Advance When**, and then change the activation to **User presses ENTER key**.

That's it. The step will now advance either when the user clicks a button or when they press Enter.

Assigning Multiple Roles to Guides

We have 4 main roles and guides are commonly shared between them. How can I set up a guide to show to more than one specific role, but not all?

Updated October 22 2020 and available in the OGL Console from November 15 2020.

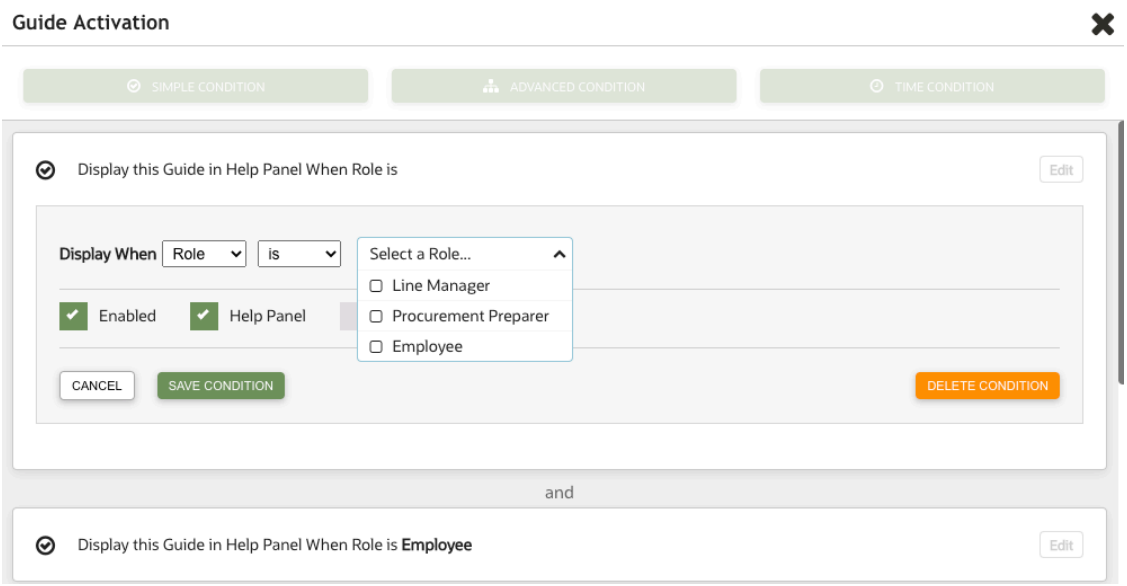
The Simple Condition allows the OGL developer choose to always display the guide or to set a condition based on a page or role.

Note: Pages and roles must have already been configured in the OGL console.

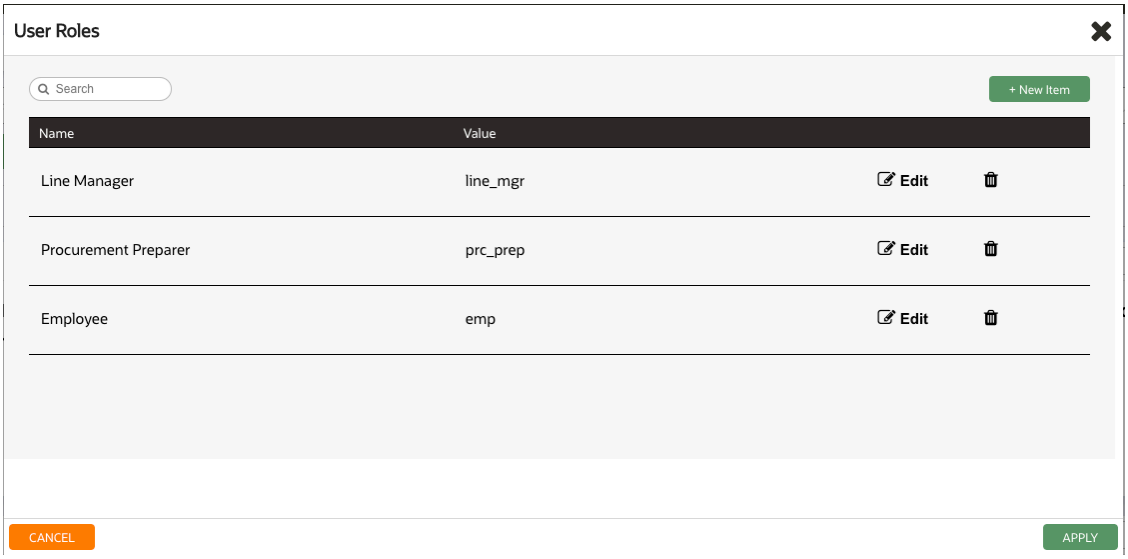
When always is selected the OGL developer only has the option to set if the guide is:

- Enabled
- Displayed on the Help Panel
- Autoloaded

If the Role option is selected the OGL developer can now set the Role condition of “Is” or “Is NOT” and select the relevant Role(s) from the drop down:



In order to work with Simple conditions OGL now utilizes the Roles setting in the OGL console. The OGL roles MUST be set up as per the roles set in your OGL Fusion embed or Javascript. Click here to download a list of Role mappings.



Branch Based on Dropdown Value

When a guide needs to branch based on the value of a drop down, if "form field - drop down" is not working, there are other options to achieve the same functionality.

Instructions

1. Create a step that instructs the user to select a dropdown value.
   - The step will need to advance using the "Next" button
2. In Advanced Settings/Step Branches, edit the Conditions for a branch and use the following:

```
When page has visible element select[title$="DROP DOWN VALUE"]
```

- Another option is something like:

  ```
  When page has form field - text .xdw > span:first > input:last with value
  matching DROP DOWN VALUE
  ```

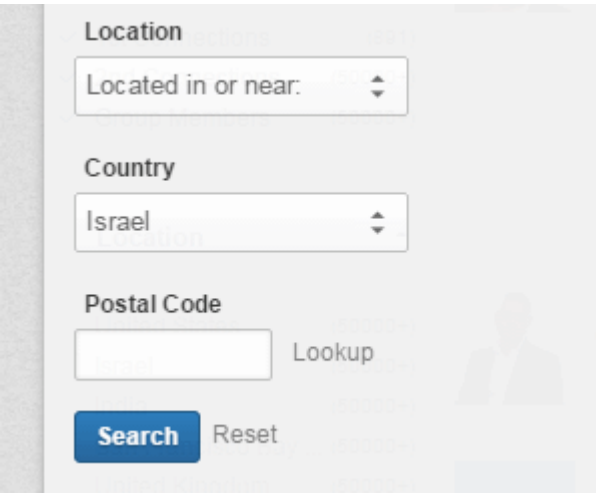
Branches and Filling Forms

Branching is one of the most powerful features in the Guided Learning editor. It can be used to create very complex flows.

Still, its most common use-case isn't all that complex. The most common use-case for using branches is to control the flow of the guide based on user choices in forms.

Real Life Example - Linkedin Job Search

Below is an image of the Linkedin job search form,

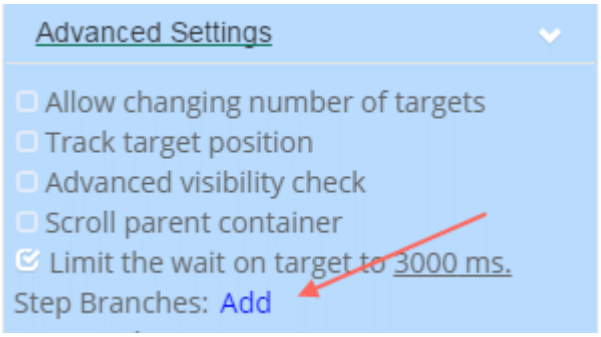


Now, for the sake of this example, let's say that if the user selects *United States* we want to recommend that he/she also enters a *Postal Code.* If a different country is selected we'll point the user directly to the search button.

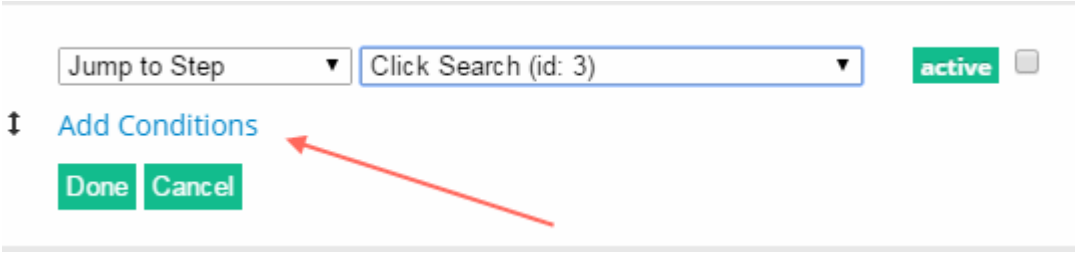
So, our guide will look like this:

1. Select a country
2. It is recommended to enter a postal code for searches inside the US.
   (this will only display if US was selected)
3. Click *Search*

The 'natural' flow of an Guided Learning guide is to go from one step to the next; so 1 then 2 then 3. In our case, we want Guided Learning to jump directly from step #1 to step #3 if the user **did NOT** select US.

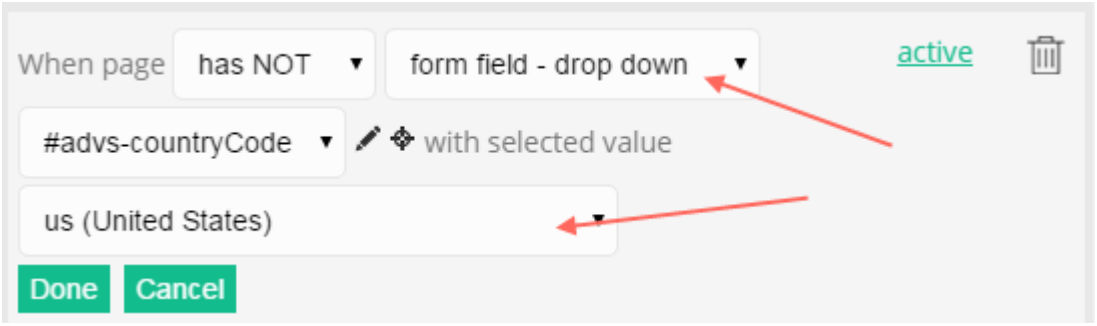


In step #1, under the 'advanced settings' click to add a branch.



Select *Jump to Step*, choose step #3 from the drop down menu.

Now we need to make sure this branch is activated only if the user did NOT select US. So let's click on *Add Conditions.*



Last step, choose the ***form field - drop down*** condition and hover over the drop down to select it. Guided Learning will automatically fill the selected value (e.g. US) for you.

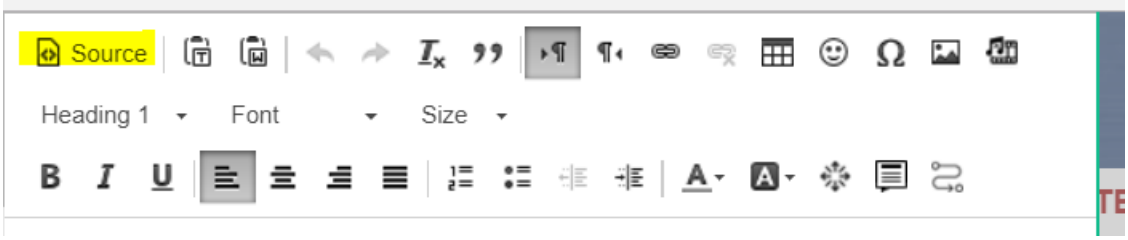
Can I hide a tooltip if another tooltip is displayed?

Yes.

Before solving this, you will need to understand the more general use-case.

In a drop-down menu example, a Guided Learning tooltip is no different that any other element on the screen.

The question is how to identify a tooltip. To do so, we will add a hidden HTML element to the source code of the tooltip.



In the source view of the tooltip we will add the following code

```
<div id="ir__only-one-tip"> </div>
```

Now, when our tooltip is displayed, this invisible div (this div is invisible because it has no content) will be part of it.

So all that is left to do is to add the following step condition to the other tooltip.

Can I launch a guide if a user gets stuck?

Yes.

You can configure a guide (with an overview of a page, for example) to automatically launch if the user does not move his mouse for a certain amount time.

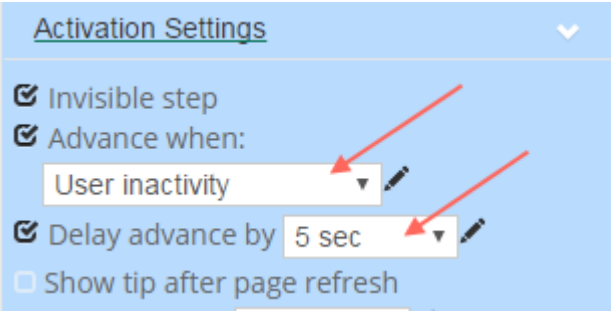
Or you could use this feature to highlight the help widget if the user is inactive.

How do I do it?

Refer to our knowledge-base article about launcher guides. There you will learn about invisible steps. For now, know an invisible step allows you to "listen in" on user activity (e.g. clicking a button) but **also to user inactivity.**

This is how you do it:

In your launcher guide, the first and only step, should be an invisible step that will look like this:



Notice that in our example we want to advance after 5 seconds of inactivity.

And that's it! You can then choose an action as described in the invisible steps article

Can I launch a guide when a user field contains a certain value?

Yes.

Consider the following use case. You want to provide your users with content relevant to the features that they use in your product.

Let's assume your product has 10 features which we'll name: feature01, feature02, ... feature10.

In your embed code you are telling Guided Learning which features the currently logged in user has enabled like so:

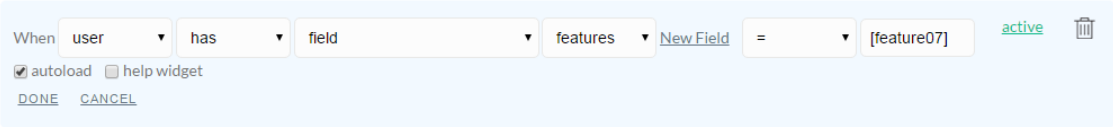
```
iridize('api.fields.set', {user_id: 'jane@acme.com', features: 'feature01, feature07, feature08'});
```

Now say you want to launch a guide only to users that have feature07 enabled.

Creating the condition "user has field features = feature07" will not do because it will only match if the user has only this feature enabled. (i.e. Guided Learning does string matching between the field and the value you provide)

Instead, we allow you to user regular expressions. Surrounding the text with [ ] tells Guided Learning that the string is in fact a regular expression. This actually allows you to create conditions much more complex than simply 'contains'.

In our example you will need to:



But say you want to have the guide show up if the user has feature07 *or* feature06, no problem:

```
[feature07|feature06]
```

The above will do just that.

Can I launch two guides simultaneously?

Yes.

There is no technical limit on the number of Guided Learning guides that can run simultaneously. Nevertheless, please consider that this may be confusing for your end users.

Can I link one guide to another?

Yes.

To create a link between 2 guides, you need to know the API name of the guide you want to link to.

Open the first guide (from which you want to link) in the editor; select the step after which the second guide should launch and open its advanced settings. Click on add/edit branches and follow these instructions.

Launch a Guide Branch Type

Choose the Launch a Guide branch type to launch another guide from within the current guide. When used without branch conditions, this is often used to string together several guides into a unified guide. When branch conditions are used this is often used for forking a guide based on user choices or application state.

Editing a Launch a Guide branch with advanced settings. This branch is set to Launch a Guide with apiName "98j68vyj".

1. Select Branch Type

Branch type select box.

2. Guide apiName

The apiName of the guide to launch. The apiName of a guide can be found in the guide's listing on the Guides page of the Guided Learning dashboard.

3. Step API Id

By default the launched guide is started from its first step. When this textbox is not empty, the launched guide will start from the step with the API id in the textbox. The step API id may be found in the Step Settings Panel of the Guide Editor.

4. Keep Current Guide Open

The Launch a Guide type of branch normally launches the guide and then closes the current guide. When the Keep Current Guide Open checkbox is checked the current guide is kept open. Additionally, the branch choosing process continues on to the following branches (if any) and to the default next step if no other branch is chosen. This is often useful when launching a guide which is run in a popup window (see Popup Window Name setting below), where the current guide stays open waiting for the user to get back from following the launched guide on the popup window.

5. Add/Edit Conditions

Click to open the conditions editor, where you can add/edit Conditions to the branch.

6. Show/Hide Advanced Settings

Click to toggle whether advanced branch settings (items 7-11 below) are visible.

7. Wait for Page to Load

When this checkbox is checked the launched guide will not start until a page load takes place (either a full page load or a SPA route load on a properly integrated SPA). This is almost always necessary when launching a guide in a popup window (see Popup Window Name setting below). Another use for this setting is when combined with the Redirect to Page or Reload Page settings.

8. Redirect to Page

When the textbox is not empty, the user will be redirected to the URL specified in the textbox when launching the guide. Please note that it is almost always a good idea to also set the Wait for Page to Load setting when using Redirect to Page.

9. Reload Page

When this checkbox is checked, the page will be reloaded when launching the guide. Please note that it is almost always a good idea to also set the Wait for Page to Load setting when using Redirect to Page.

10. Popup Window Name

By default the launched guide starts in the current browser tab/window. When this textbox is not empty the guide will be launched in a tab/window with the Javascript window.name property matching the text in the textbox. This is useful for launching guides within a popup window or another tab. For use with a popup window, this setting almost always requires also setting the Wait for Page to Load setting to work properly.

11. On Close Return-Here/Launch-a-Guide

This setting sets an action to take place when the launched guide is closed (either by the user or by finishing the guide).

- **Return Here**: with this setting the launcher guide is suspended upon launching the guide. When the launched guide is closed the launcher guide resumes from the same step it was when suspended.
- **Launch a Guide**: with this setting the launcher guide is closed as usual. When the launched guide is closed, the guide with the apiName set in the textbox is launched.

12. Done/Cancel

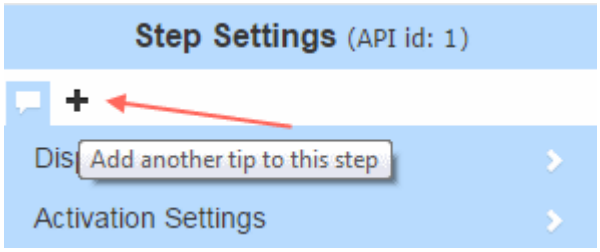
Click on **Done** to finish editing the branch and keep your changes. Click on Cancel to discard the changes made to the branch.

Can I open more than one tooltip at once?

Yes.

There is no limit to the number of tooltips that can be opened simultaneously.

You can tell Guided Learning to display more than one tooltip at any step in your guide by using the + icon



Tip: This can be very useful for creating hover help for a certain page in your app.

Can you have a guide with only one step, and not show the step counter?

Yes.

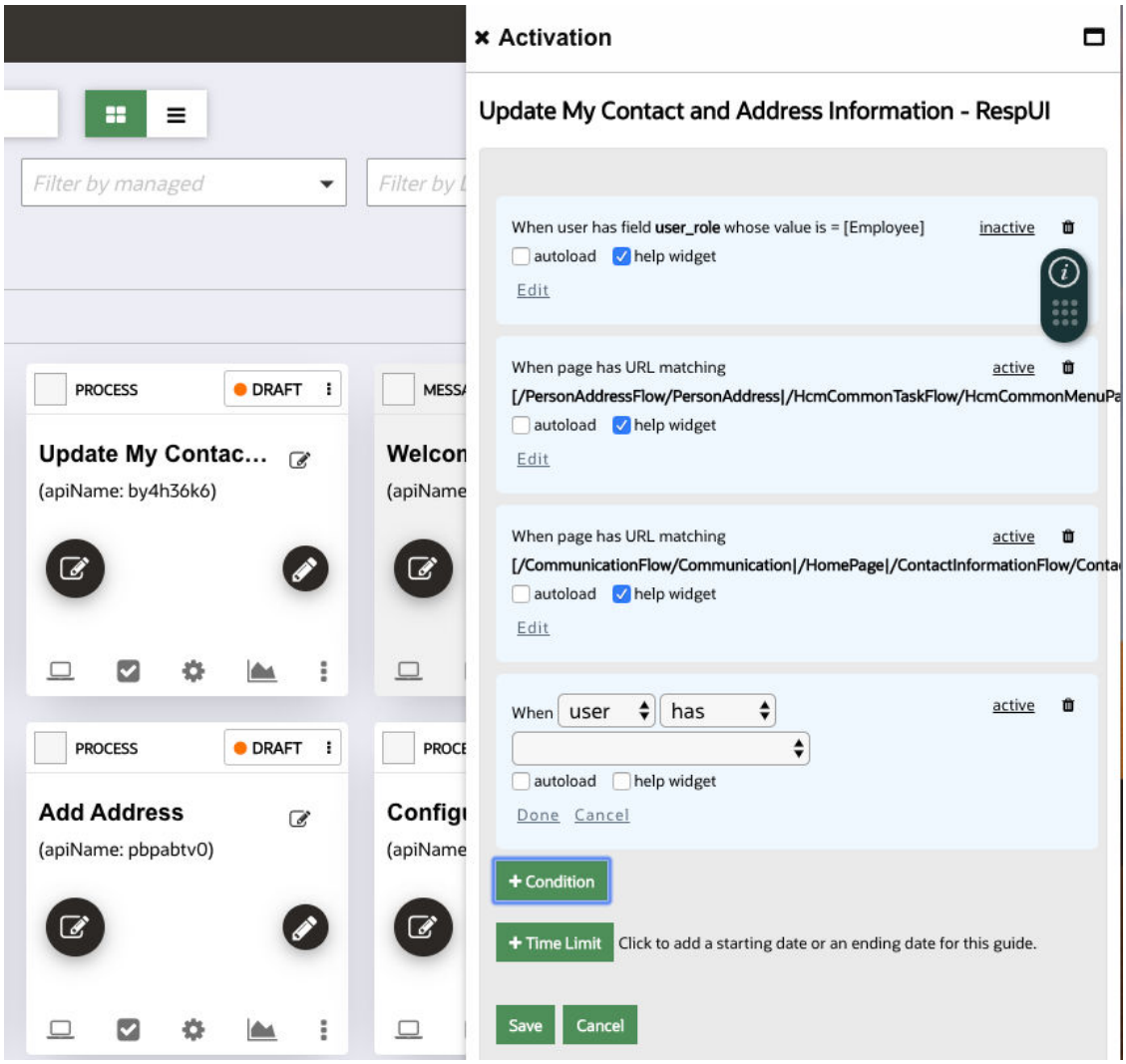
Actually a common use-case for that is when showing an **overview 1-minute video** to a first time user. In the display setting of a tooltip, uncheck the step counter option.

Can you set a condition for a guide to only show once?

Yes.

Under 'Activation'

Add a condition to "show when user has seen guide less than 1 time"



Changing the Flow Based on a Users's Choice

Sometimes you need to ask the user a question and **change the flow of the guide based on the answer that the user provides**. The answer does not necessarily have to be something in the UI of the application. It can be something specific to what the user wants to do now.

A good example of this is creating a troubleshooting guide or, as some call it, a diagnostics guide.

You have probably already been on one of these phone calls with a customer support representative (CSR). You explain the problem to the CSR and in return s/he asks you a list of

questions. Each question depends on the answer you gave to the one before it. At the end of these questions, hopefully, your problem is solved.

A troubleshooting/diagnostics' guide can be of tremendous help to either party here:

1. If you didn't have to call anyone to ask these questions you'd be happy to service yourself!
2. If the CSR did NOT have to read this list of questions from a printed document or worse her/his torn notebook, the service would be much quicker and less prone to error.

Actual Use Case

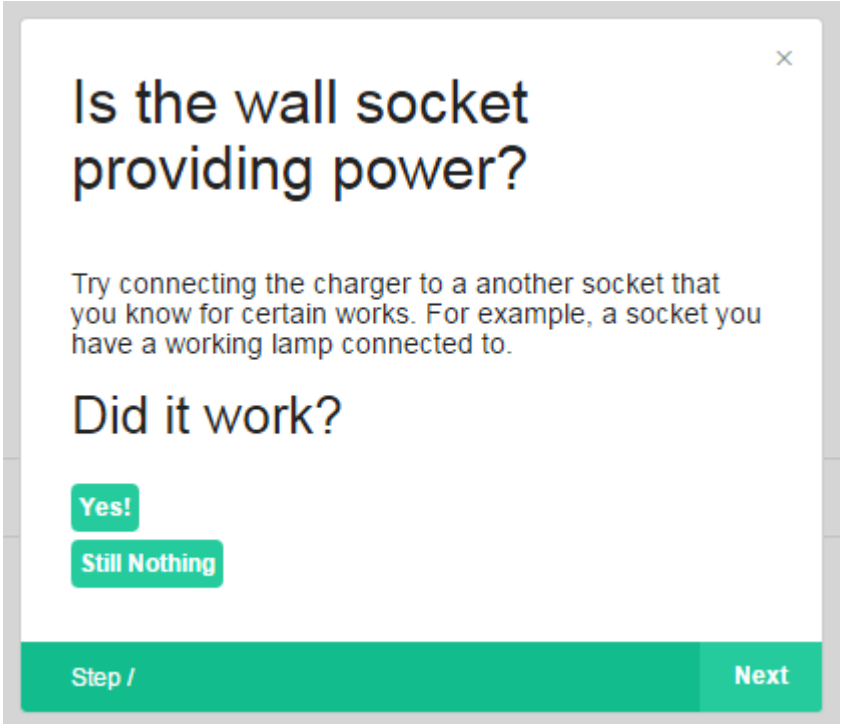
Let's see this in action with an real life scenario, **"smartphone battery does not charge"**.

This would be the document your CSR will be reading from:

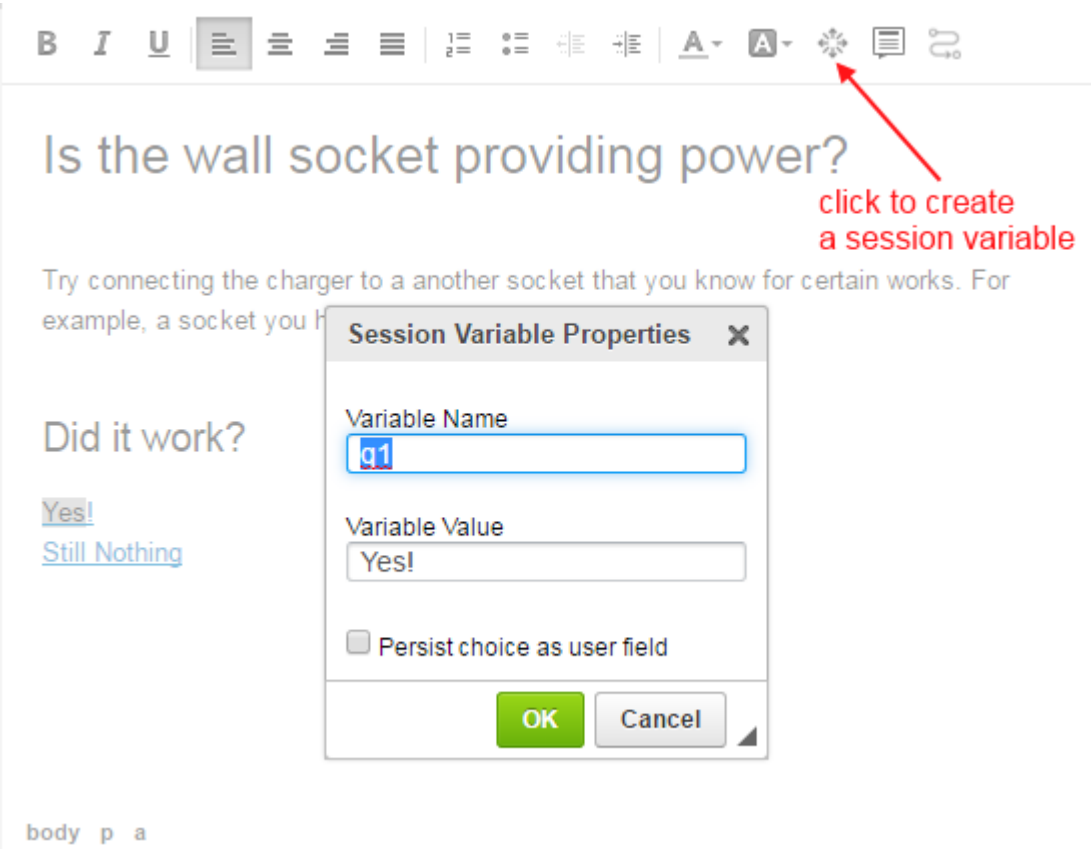
1. CSR: Is the wall socket providing power? Try connecting the charger to a another socket that you know for certain works. For example, a socket you have a working lamp connected to.
2. You (option 1): I changed the socket and it works! --> FINISH
3. You (option 2): Still nothing --> GOTO 4
4. CSR: Are you charging from a USB port on a computer?
5. You (option 1): No --> GOTO 7
6. You (option 2): Yes --> GOTO 8
7. CSR: Contact your service provider. The phone may be damaged.
8. CSR: ensure that the computer is turned on. Some computers provide no USB power when they’re turned off.
9. You (option 1): I turned on my computer and it works! --> FINISH
10. You (option 2): Still nothing --> GOTO 7

The Guided Learning Way - Session Variables

Guided Learning allows you to ask the user a question inside a tooltip; It saves the user's answer and allows you to change the flow of the guide accordingly using *Branches*. Read more in our Branches article.

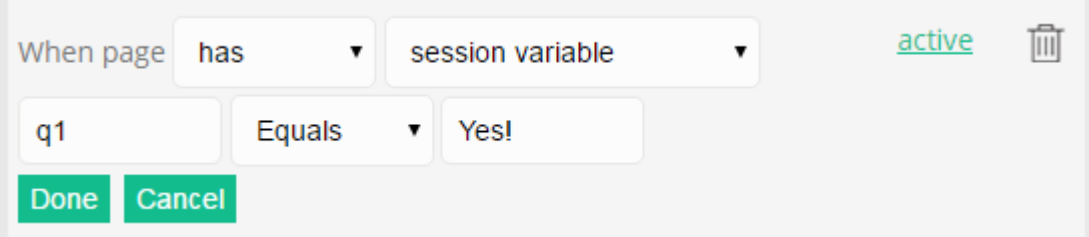


The tooltip above consolidates the 3 steps in our list into a single tooltip. To configure the possible answers we will need to click on the session variables button and configure each answer as can be seen in the image below.



Notice that all answers should have the same value (e.g. **q1**) set as the *Variable Name.*

What we need to do next is to alter the guide flow based on the answer that the user clicks on. To do this we will setup a branch to jump to the relevant step. In our example, if the user click ***Yes!*** we should end the guide; Guided Learning will default to going to the following question hence we do not need to creating any additional settings for the case that the user selected ***Still Nothing***



The above outlines the condition that needs to be added to the branch.

Cloning a Tooltip

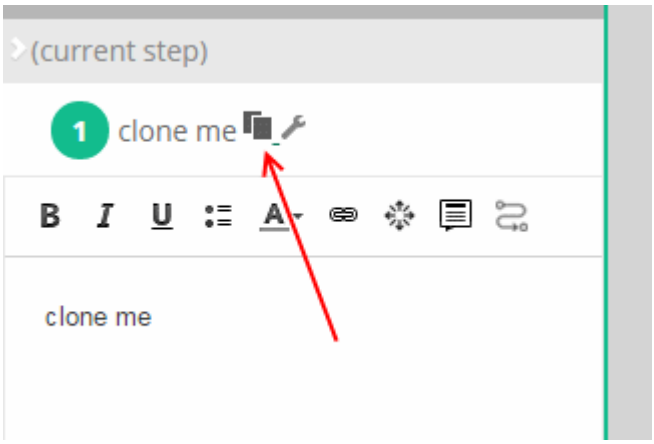
Cloning a tooltip is useful when:

- You have created a beautiful tooltip, with customized settings and styles. And you want to create another one just like it
- You have created a step with multiple tooltips and you want to move one of those tooltips to a new step
- You want to move a tooltip, you have created, into an existing step (e.g. to branch the guide)

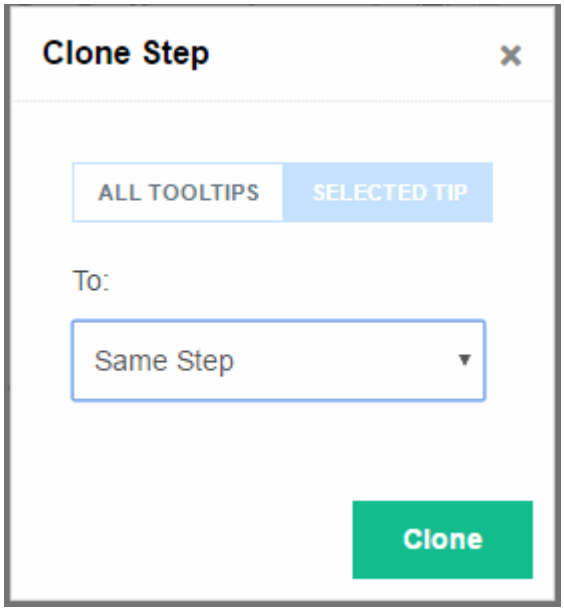
Cloning will create an identical copy with all the text, styling, conditions, branches and any other setting.

To clone a tooltip, in the editor, you will need to select it first.

Once selected, click on the clone icon:



The clone dialog will appear,



1. Choose whether you want to clone only the *selected tip* (default) or *all tooltips*
2. In the drop-down select where you want to clone to. This can be the *same step* (default), a new step (the new step will be created immediately after the selected step), or any other step in your guide.
3. Click the *Clone* button

Context Sensitive Help (CSH) using Oracle Guided Learning

"Context-sensitive help is a kind of online help that is obtained from a specific point in the state of the software, providing help for the situation that is associated with that state."

This definition is taken from this wikipedia article. Also from this article, "Context-sensitive help can be implemented using tooltips" which is exactly where Oracle Guided Learning comes in.

While you can use Guided Learning to create flows to guide users through complex procedures, sometimes, some extra information that appears when you put your mouse cursor over a button or a form field can do a world of good.

In Guided Learning we call these on-hover tooltips. To add them to any page in your application please do the following:

1. In the display settings of each tooltip, remember to:
   - check 'Hide Next button'
   - check 'Hide Close button'
   - uncheck 'Show Back button'
   - check 'Hide step count footer'
2. In the activation settings of each tooltip, remember to set 'Show on hover'
3. In the step settings section uncheck 'Highlight Target Element'

Contextual Page Level Help

Let's say that a new feature has been added to your application and you've written a detailed article about how to use this feature and have added this article to your knowledge-base/help-center/wiki.

How do I get the users who use this feature to read the article and to see the full power of the feature?

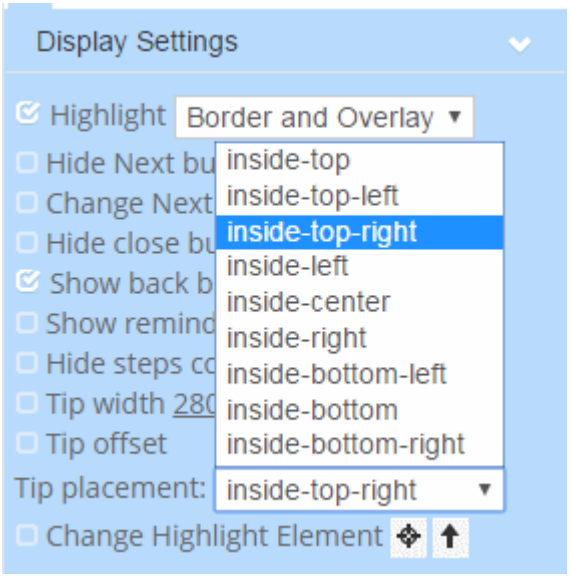
What if you could:

1. Open a splash window for the first 3 times the user lands on the feature page
2. Embed a 30-second overview video of this feature
3. Write the key benefits of using this feature
4. Link to your detailed article for more information

Actually, you can do this! We call this **Contextual Page Level Help**.

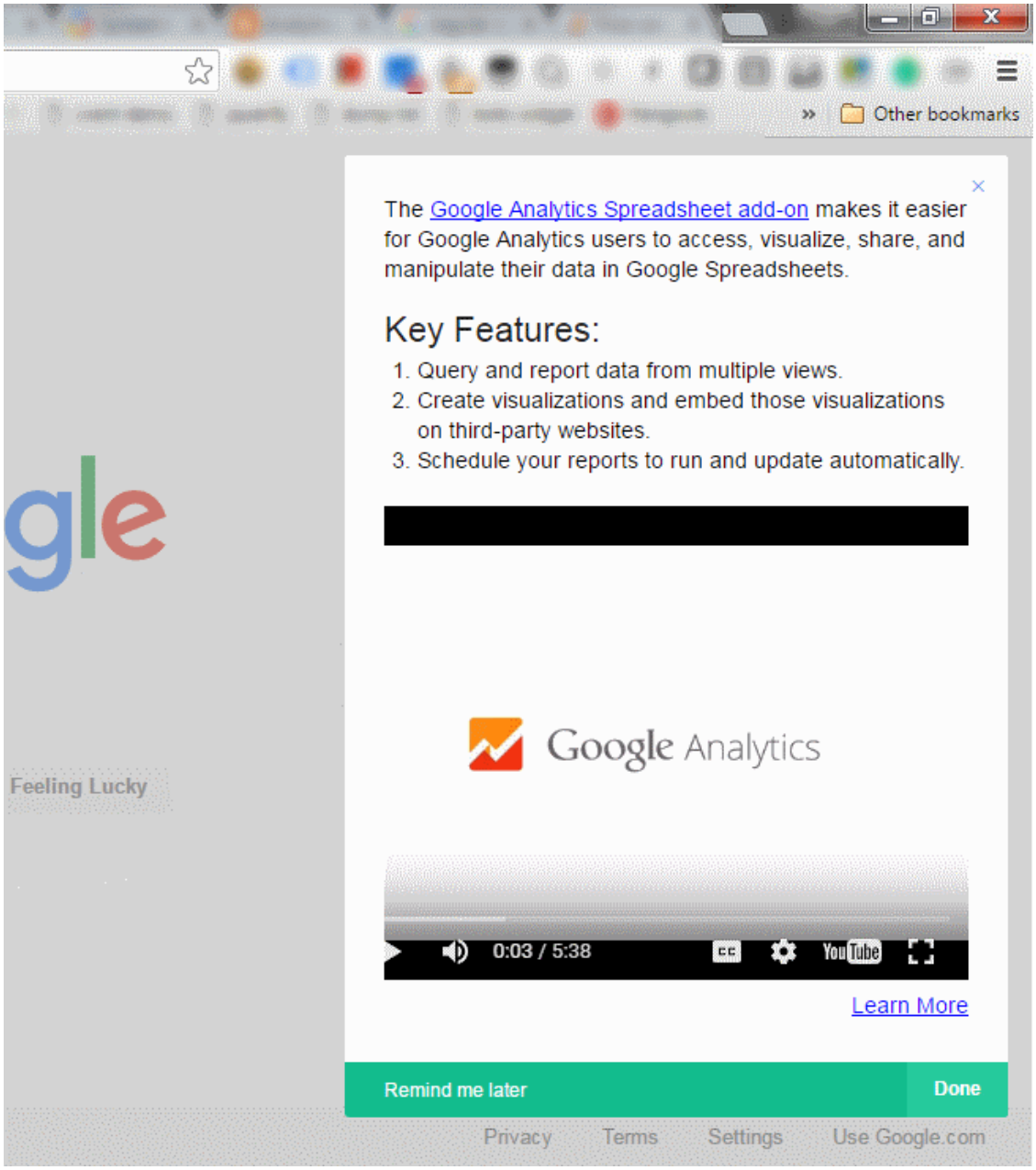
Here is how Guided Learning can help you look like a pro:

- Create a new walk-through guide on your guides page
- Make the first step a splash.
- Use the display settings to change the placement of the tip to be *inside-top-right*



- Now open the extended text editor and write/embed the desired content.
- Be sure to remove the *Back* button and consider using *remind me later*
- Remove the *Highlight*; this would allow the user to interact with the page as he sees fit while reading your content.
- Under the *Advanced Settings*, set *Fixed Position*; this will make the tip stay in view as the user scrolls the page up or down.

Here's the result:

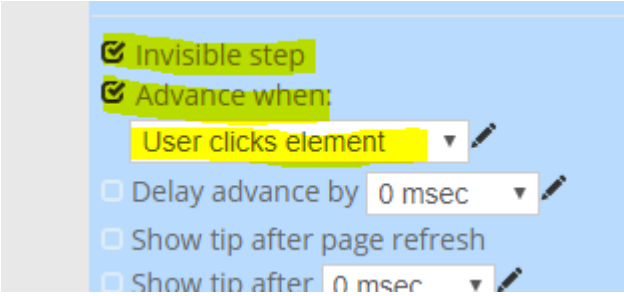


Create Popup Message on Disabled Element

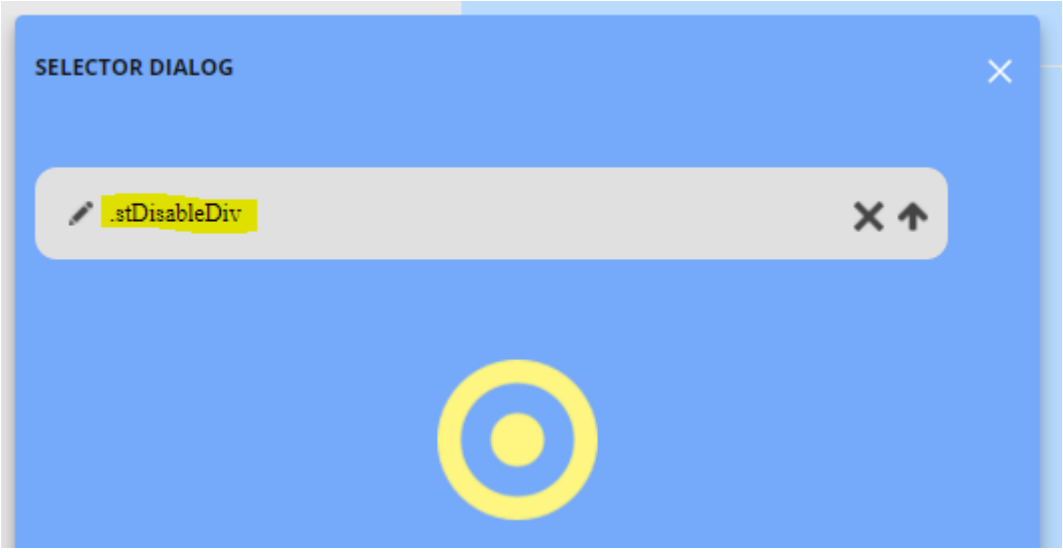
Instructions

Add the steps involved:

1. Create the disable layer as you normally would
2. Make it invisible and set it to advance on click



3. On the same step, create another sub-step that will 'show on hover'.
4. Set the target element for this tip to **.stDisableDiv**



The way OGL disables an element is by placing a "transparent" layer on top of it. If for example we disable a button when the end users click on the button they are in fact clicking on that transparent layer and the button never "sees" the click and is hence unusable.

When you set a tip to show on hover, OGL waits for the end-user to put their mouse over the relevant element and show the tip then; but if the element is also disabled, the mouse will never be over the element but only on that "transparent" layer mentioned above.

The solution is to have the tip show when the mouse goes over the layer instead of the element itself.

Embedding an IFrame in an Iridize Step

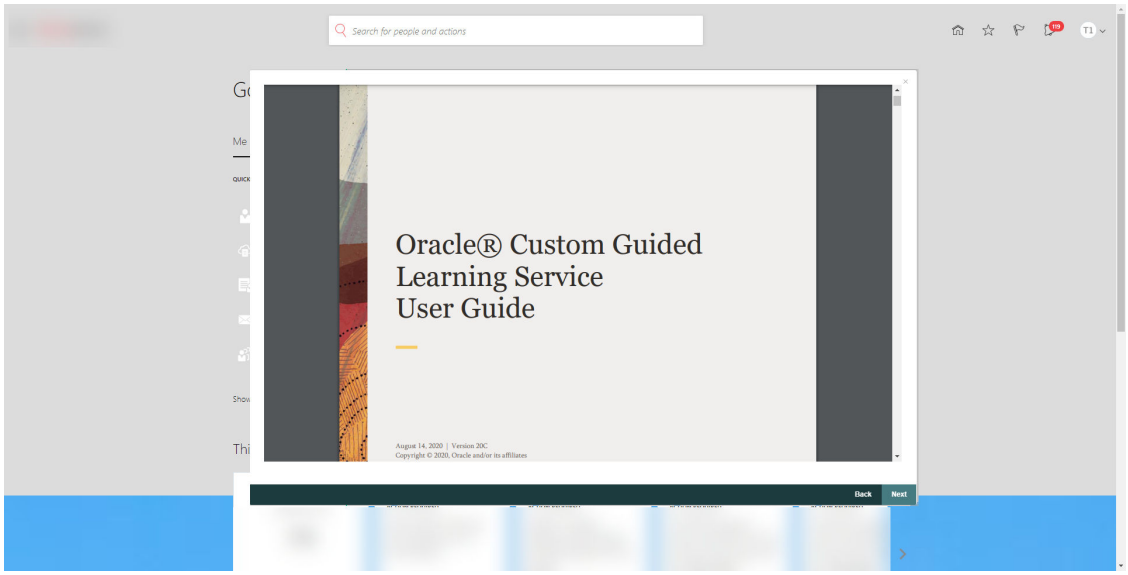
One of the cool ways of using Guided Learning is embedding external content into a Guided Learning tooltip.

You can use this feature to:

- Provide your users with a survey that you've created (in SurveyMonkey or in Google Sheets for example) directly over the interface of your product.
- Give your users direct access to relevant information in your FAQ or Knowledgebase while they are using the product
- Sharing information from an external source or website in a contextually relevant

This feature utilizes Guided Learning's ability to embed iframes (or *Inline Frame*) in our tooltip. This iframe-tooltip can appear as single-step guides or as part of a longer guide (one of the steps), based on your requirements and the process.

And, of course, these guides and tooltips can be made to appear according to pre-set conditions (only once, only on certain pages, only to certain people, etc.), just as any other guide. In addition, Guided Learning collects all usage data and lets you know that your users have actually seen this content (even though it is outside of your product).



793px

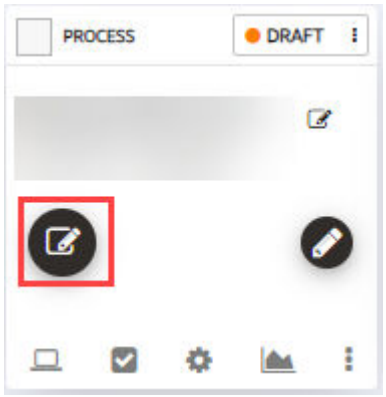
Embedding an iframe

The mechanism of embedding an iframe into a bubble is similar to embedding a video or an audio link and uses the same process. Guided Learning has a generic algorithm that recognizes the type of content that is provided in the URL and automatically deploys it correctly in the bubble.

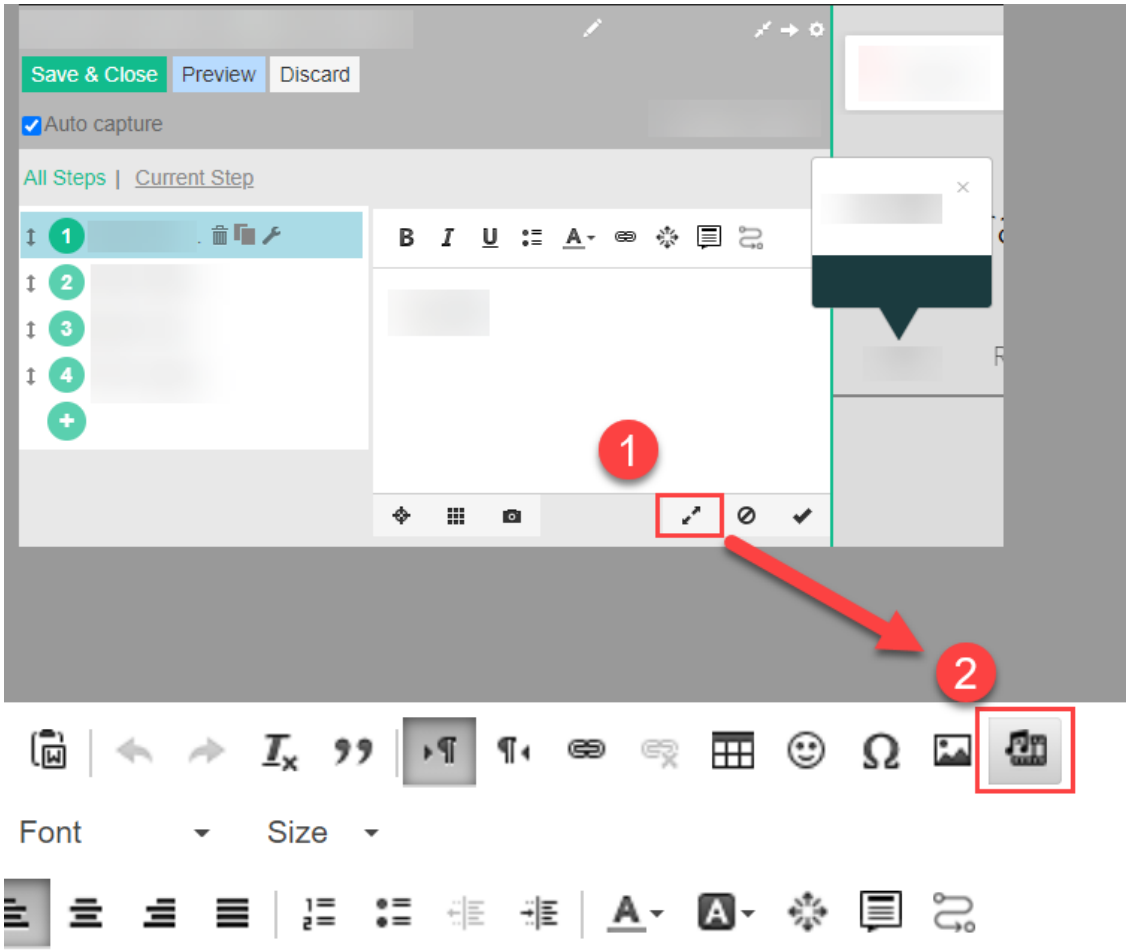
Just as with video and audio, there are two ways you can embed iframes into Guided Learning tooltip. In both cases, the process is easy and quick.

From the Editor

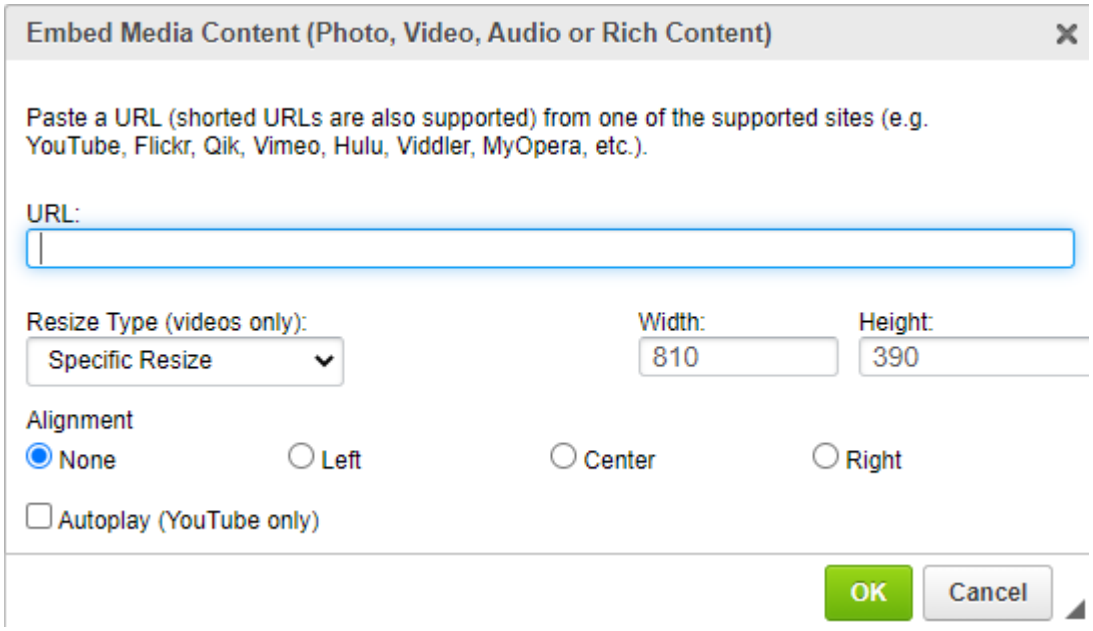
- Launch the Guided Learning Editor



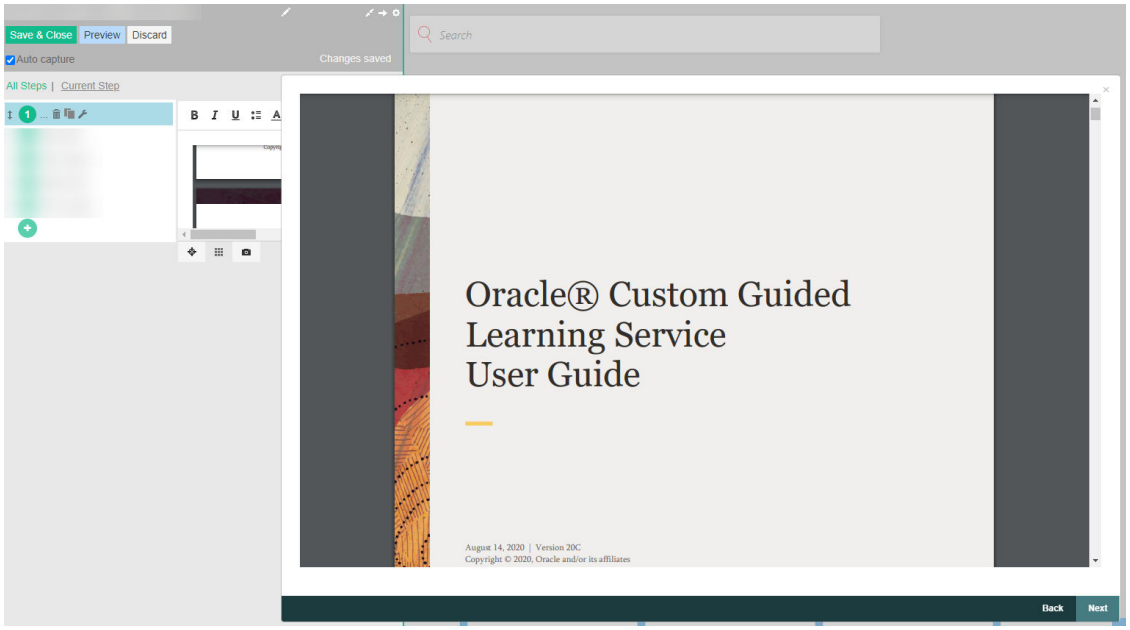
- Click **(1) Open Full Editor**, then select **(2) Embed Media From External Sites** (see Image)



- In the console box enter the URL for your iframe and click “ok”



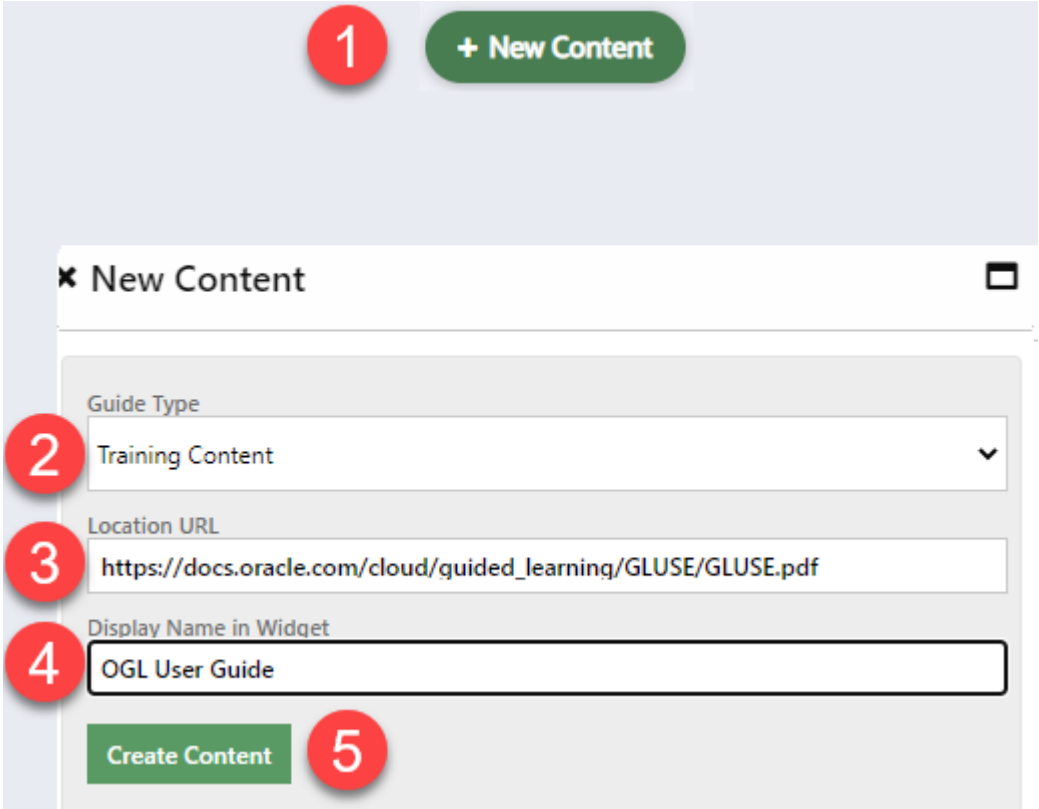
- The external website will now appear in your Guided Learning tooltip



- You can minimize the Expanded Editor and either continue to the next Step or click “Done” and continue to the Guide Activation Conditions.

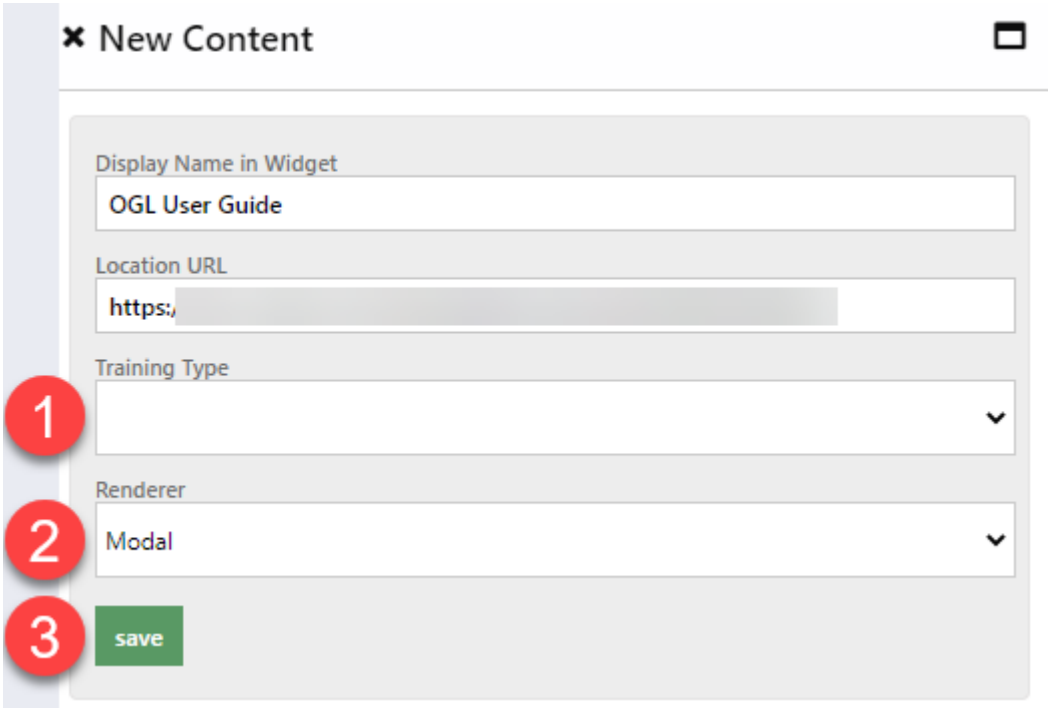
From the OGL Console

1. Click **New Content**.
2. Select **Training Content** from the **Gude Type** dropdown.
3. Enter the **Location URL** of the page you wish to embed.
4. Enter a **Display Name**.
5. **Click Create Content**.



Next:

1. Select a **Training Type** from the dropdown.
2. Select **Modal** from the **Renderer** dropdown.
3. Click **Save**.

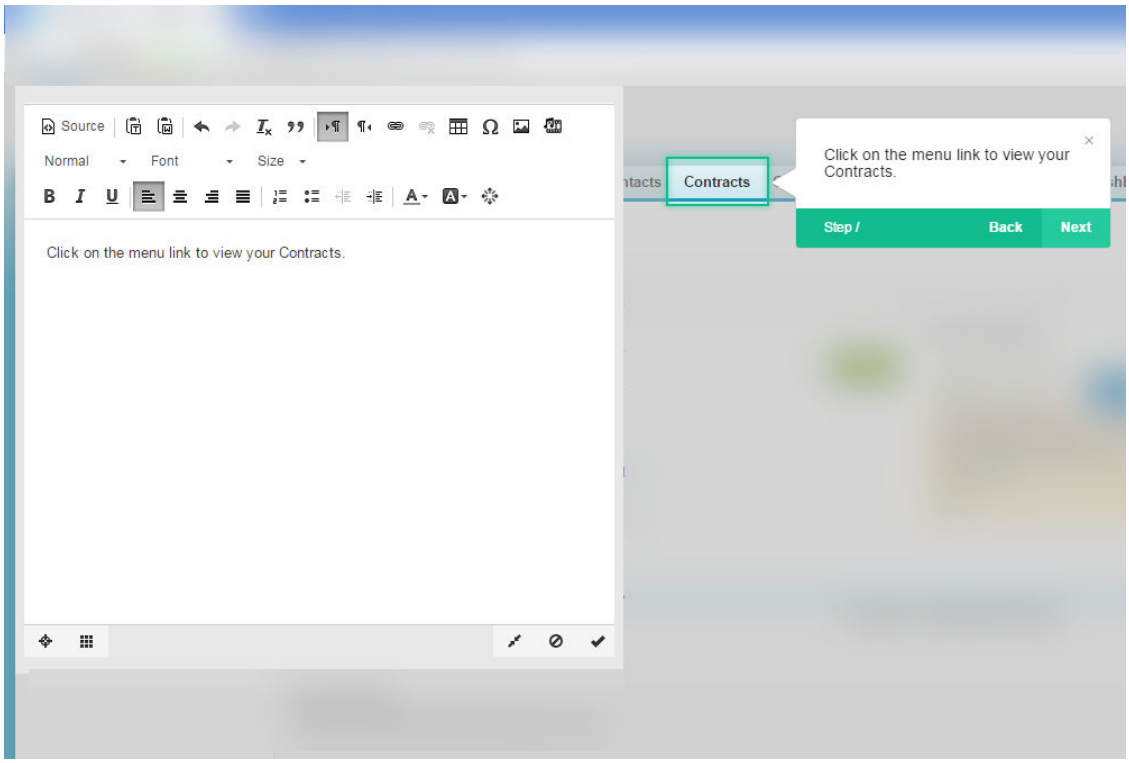


- You can now choose the URL you wish this guide to appear on.

IMPORTANT: Some products do not allow you to embed them inside of an iframe. We suggest you test each page to see that it works before launching it to your users.

Expanded Content Editor Reference

The expanded view of the editor includes many options for creating, styling, and embedding rich content into your Guided Learning guides and walkthroughs. Below is a quick reference for the most important options and their application.



Source -

View Source: view and edit the HTML source of the step's content directly.

Advanced Paste: when pasting rich text content into the editor you may want to use these buttons to paste as plain text (without styling) or to paste from Word and keep the content styles intact.

-

Undo/Redo: undo changes to the content or redo changes that were previously undone.

-

Clear Styling: remove all content styling, such as text color, background color, font size, and type, etc.

-

Quote: turn selected text into a Quote element with its custom styling.

 -

Text Direction: Set the direction of the currently selected paragraph to Right or Left.

 -

Link/Unlink: turn selected text into a link (anchor) or insert a link at the current cursor position. The Link Dialog allows you to set the type of the link, its URL, and its target window (open in same/new tab or window).

 -

Table: add or modify a Table element. Table Dialog lets you set table properties such as the number of rows and columns, table headers, border, width, and more.

NOTE: you can also right-click a table in the content to open the table properties as well as cell and row-level options.

 -

Special Symbols: add a special symbol to the current cursor position. This is especially useful for adding currency symbols, such as Euro or Pound or Yen.

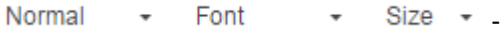
 -

Image: add or modify an Image element. The Image Dialog allows you to set image properties such as its URL, size, alignment, and whether it is a Captioned Image.

NOTE: you can change the Image's size both from the Image Dialog or by the in-content drag-to-resize controls. For Captioned Images, the caption can be edited directly in the content.

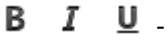
 -

Embed Video (and media): add or modify a Video or Slideshow from and external sources such as Youtube, Vimeo, and Slideshare. While the main use of this option is to embed Videos in guides and messages the full list of supported services also includes Audio, Photo and Rich Content sources such as Soundcloud, Bandcamp, Flickr, Instagram, and Wikipedia. Here you may find the full list of currently supported sources.

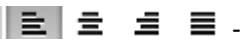


-

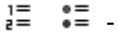
Text Style: set or modify the selected text's Style (Heading 1-6, Paragraph), Font type, and Font Size using these drop-downs.

 -

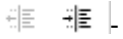
Emphasis: set or unset Bold, Italic, and Underline emphasis on the currently selected text.

 -

Text Alignment: Set the alignment for the currently selected paragraph to Left, Center, Right, or Justify.

 -

List: add a numbered or bullet list. Right-click the list elements to open up the list properties dialog, where you may choose numbered list type (Decimal, Latin numerals, Alpha, etc.) or bullet list type (Circle, Disc, or Square).

 -

Indentation: increase and decrease text indentation level. Increasing the indentation level of a list item creates a nested.

 -

Text Colors: set the text and background colors of the currently selected text. The color selection dialog lets you choose a color from a list of preset colors or enter your own custom colors.

 -

Session Variable Button: add a Session Variable Button. Session Variable Buttons can be used to create rich guide flows based on user choices. Clicking on a Session Variable Button dynamically sets a Variable that can be used in Step and Branch conditions to control step activation and guide flow. The Variable can also be persisted as a User Field, which can be used in Guide Activation Conditions.

Table Border Removal

When creating a tip containing a table for formatting purposes (padding), you probably don't want the border to be visible.

To hide the border, add the following to the theme:

div.sttip table, div.sttip tr, div.sttip td {

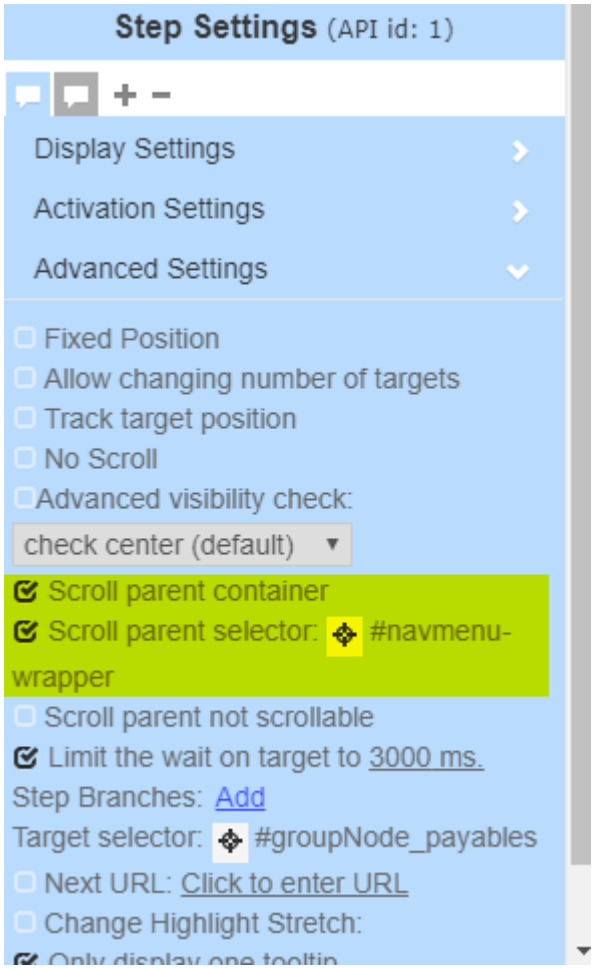
border: none;

padding: 6px 9px;

}

Horizontal Scrolling

When you are on the top home page of Fusion and you need to have the user select an element from the horizontal scroll bar at the top of the page that is not visible unless you scroll, and you want to have an additional tip in the same step for the next navigational step, follow the settings in the screenshot below:



Pay special attention to the Scroll parent selector and make sure it is set to "#navmenu-wrapper".

How can I embed a video in a guide?

In order to embed a video you must first upload the video to a video hosting service (e.g. Youtube or Vimeo). If your video is hosted elsewhere please work with your Oracle account team.

Video Guide

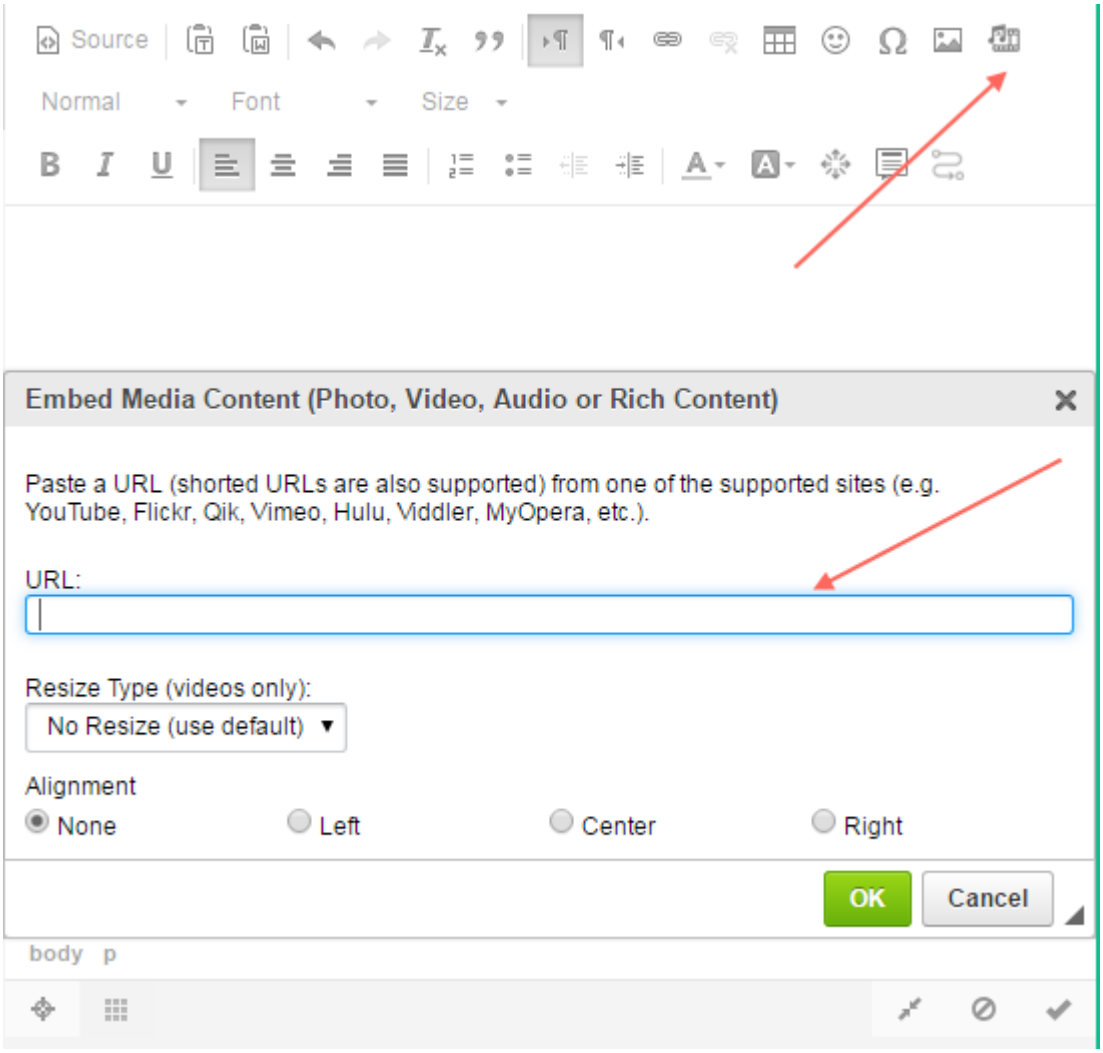
There are 2 options to embed a video an Guided Learning guide. If you simply want to share a video with your users Guided Learning allows you to quickly create a video; a video guide is a guide with a single tooltip that contains a video. It will look something like this:

To create such a video guide follow these instructions:

Guided Learning allows you to embed a video in any step of any walkthrough guide. You never need to create a special guide in order to embed a video.

Open the relevant guide for editing; select the step you would like to embed a video in or create a new step. Once the step is selected open the full text editor (see image below).

once clicked, click on the the 'embed media' button and paste the URL of page where the video is hosted.



Note: Try both the URL from the browser's address bar or the embed URL. For example, in YouTube those the URL looks like this

https://www.youtube.com/watch?v=L4VERUyBEVc&index=4&list=PLYZ9LfvwzQZJDMSz6KsPDfJQqYBmpa5-R

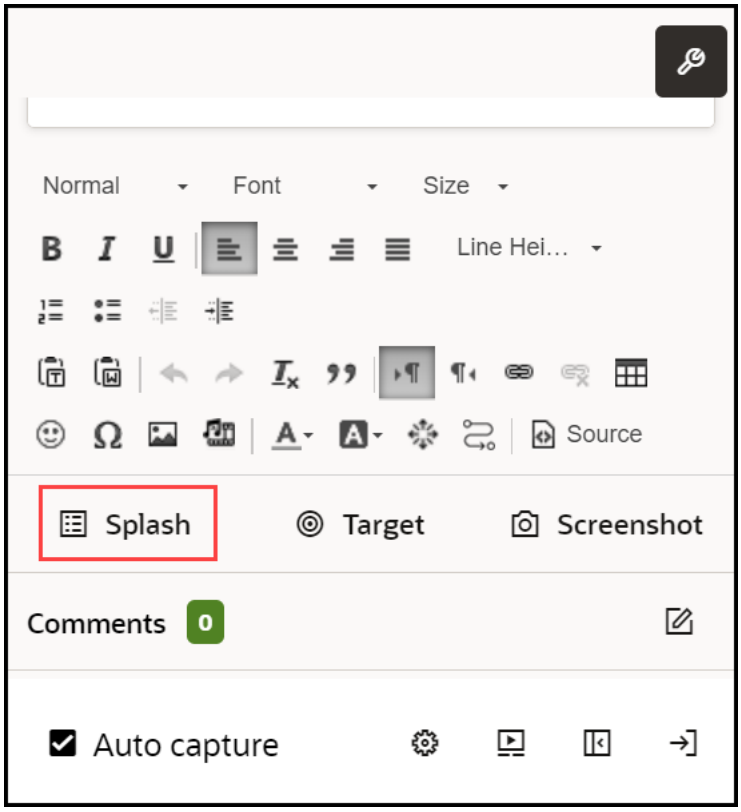
and the embed URL looks like this:

https://youtu.be/L4VERUyBEVc?list=PLYZ9LfvwzQZJDMSz6KsPDfJQqYBmpa5-R

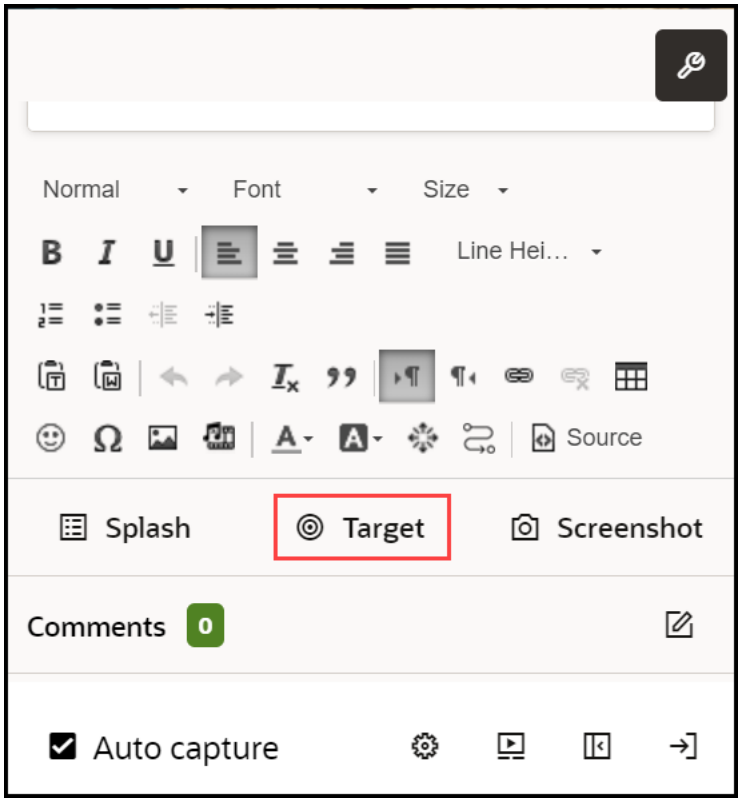
How can I point a Guide to the OGL Widget?

Follow these steps to Point a step in a guide to the OGL widget:

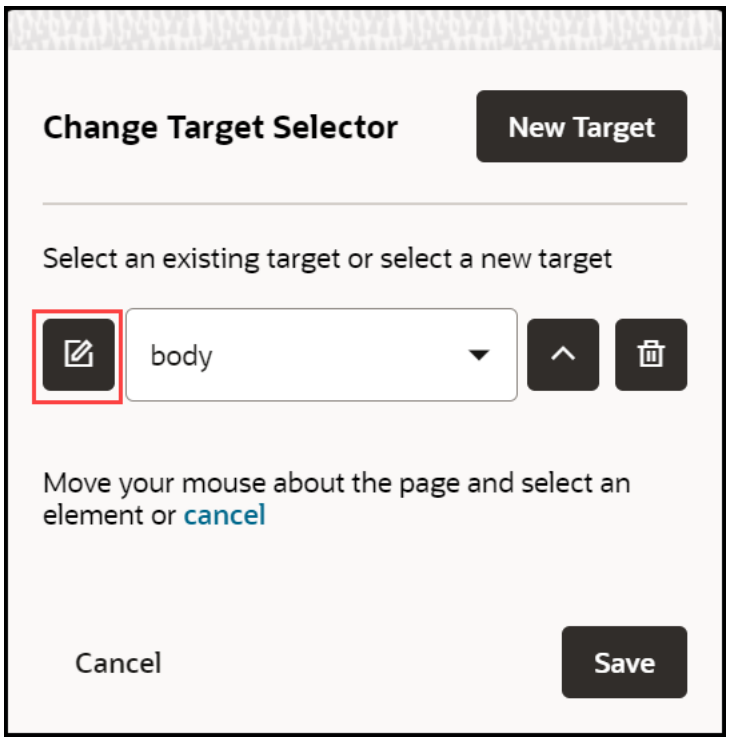
1. Select the **Splash** to temporarily give the step a target as **.body**



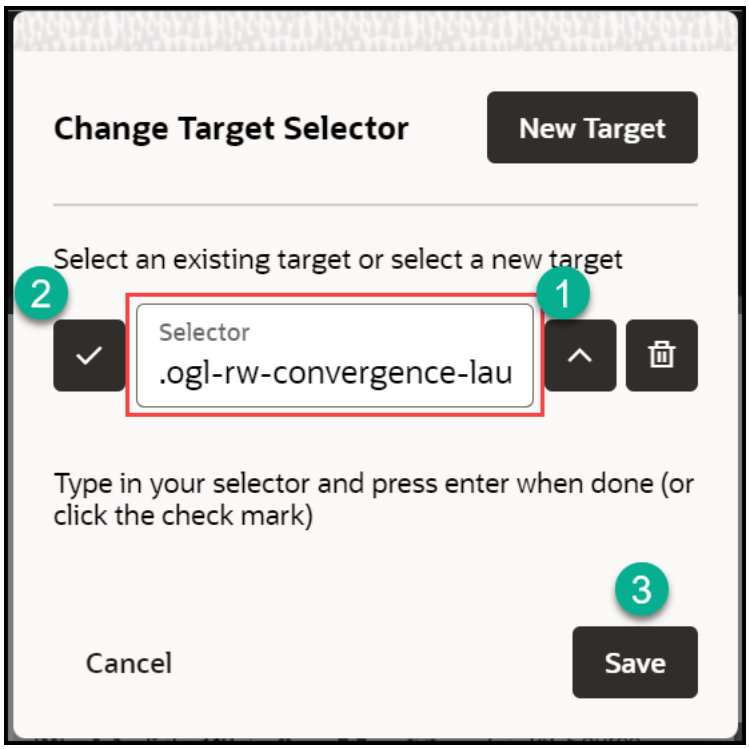
2. Select **Target** to enable the functionality to manually select the target selector.



3. Select **Edit Selector** to enable manual editing of the target selector with a text string.



4. Once the edit field is active, replace the current selector with the following: **.ou-convergence-launcher**
5. Select the check mark (√) to save your changes.



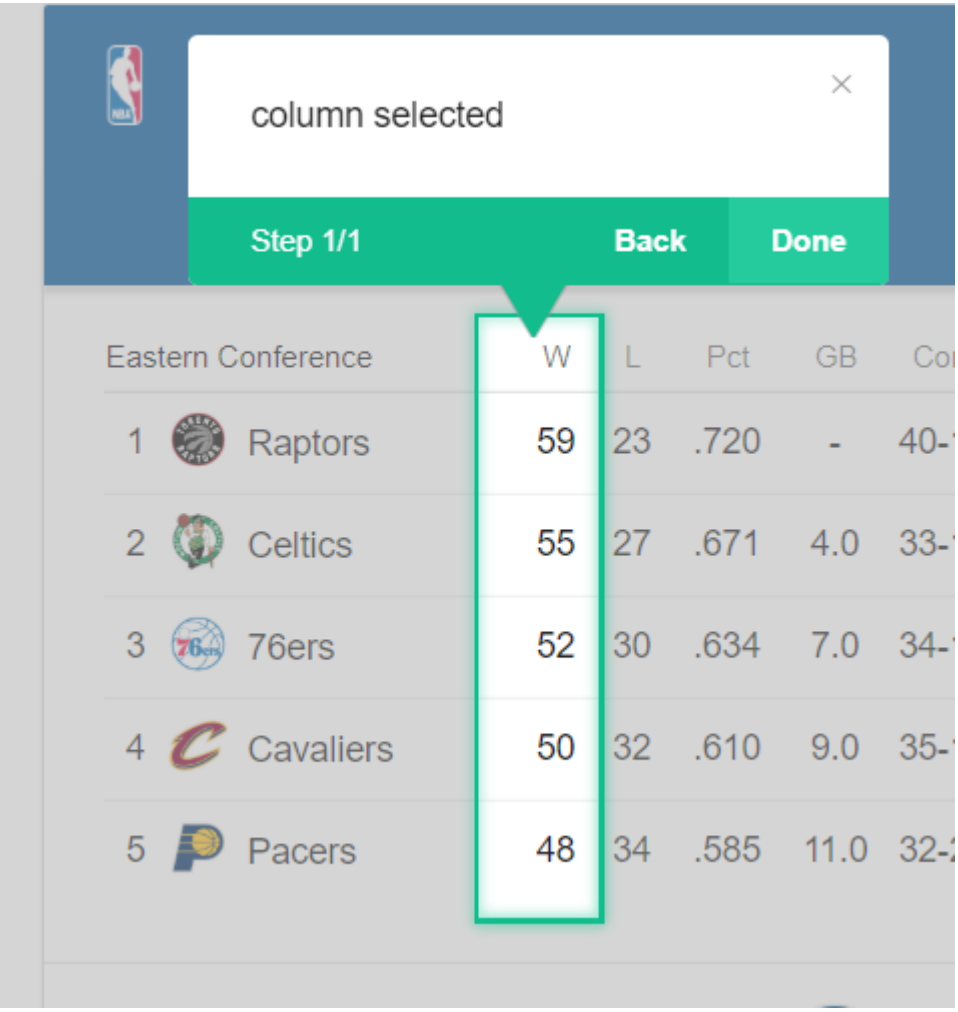
Note: The step will not display in the editor, as the OGL widget is not present whilst editing a guide. Test your changes by selecting 'Save and Close' and then by running the guide from the OGL widget.

How do I select a column?

Selecting a column is not straight forward since a column is not an HTML element. An HTML table is modeled as a set of records displayed one after the other. The columns are only a matter of display.

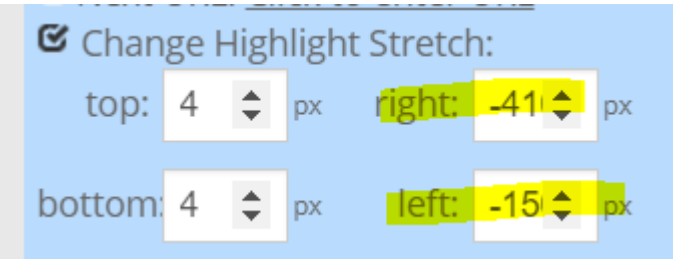
Instead of selecting a column it is recommended to point a tip at the column's header.

If you want to create a border around the column and mask the rest of the page you will need to select the entire table and then use the ***highlight stretch*** to make it smaller.

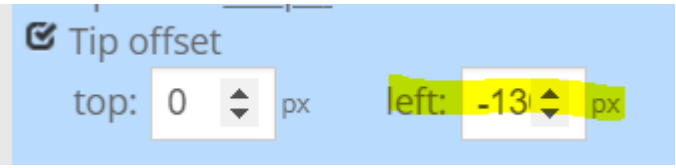


Use negative numbers for the left and right to make the highlight "smaller".

IMPORTANT: using the highlight stretch option is not responsive. the pixels you choose are hard coded no matter what the size of the screen is.



You may also need to check the offset of the tip itself because by default the tip will position at the center of the table. To do that use the ***tip offset*** setting



How do I skip the navigation steps if I'm already on the right page?

Consider the following use case:

You have created a guide that instructs the user on how to update the contact person for an account.

Your guide is divided into two logical parts:

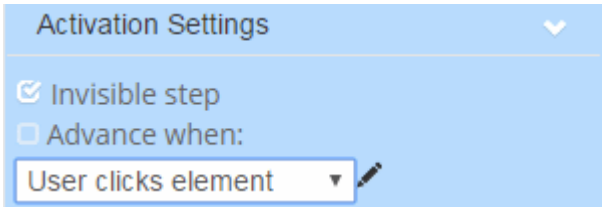
1. Searching and navigating to the relevant account
2. Updating the contact person

Now what if the user is already on the page for the account, you would want the Guided Learning guide to skip the first part of the guide and jump over the first part directly to the second (i.e. updating the contact person).

Let's see how this can be achieved with Oracle Guided Learning

Essentially we need to make our guide identify that it is on the right page and if so jump (branch in Guided Learning speak) to the first step of the second part.

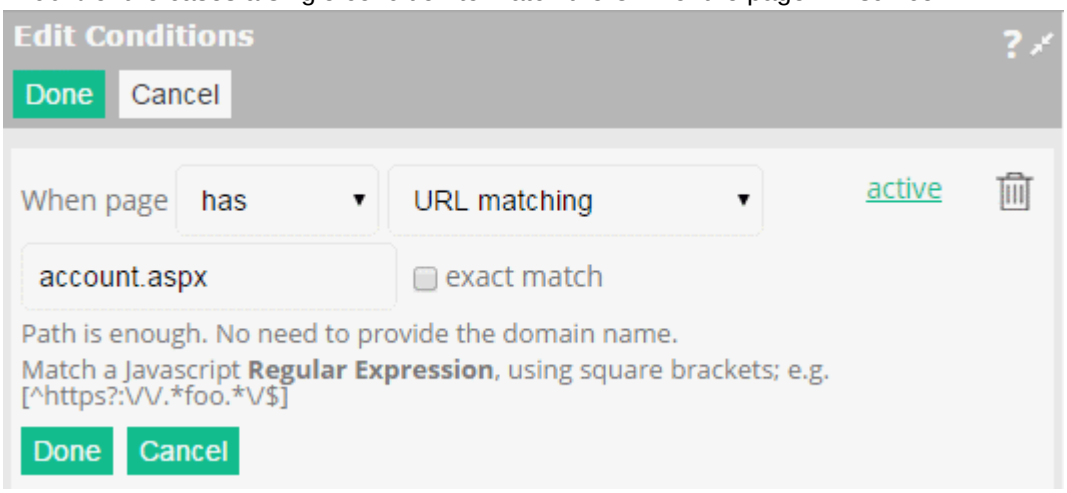
1. To do that we will add another **splash** tip on the first step of the guide
2. Make this tip invisible



   Make sure the 'Advance when' option is left unchecked.
3. Add a branch to jump to the relevant step (first step in the second part)

If we stop now, the guide will always jump to the second part. So we need to add step conditions to limit this behavior.

4. In 90% of the cases a single condition to match the URL of the page will suffice



Note that you don't have to provide the full URL

How to Capture New Screenshots in OGL

Base guides include screenshots captured in a demo Vision environment. You can change the screenshot from Vision to your own Cloud application screenshot.

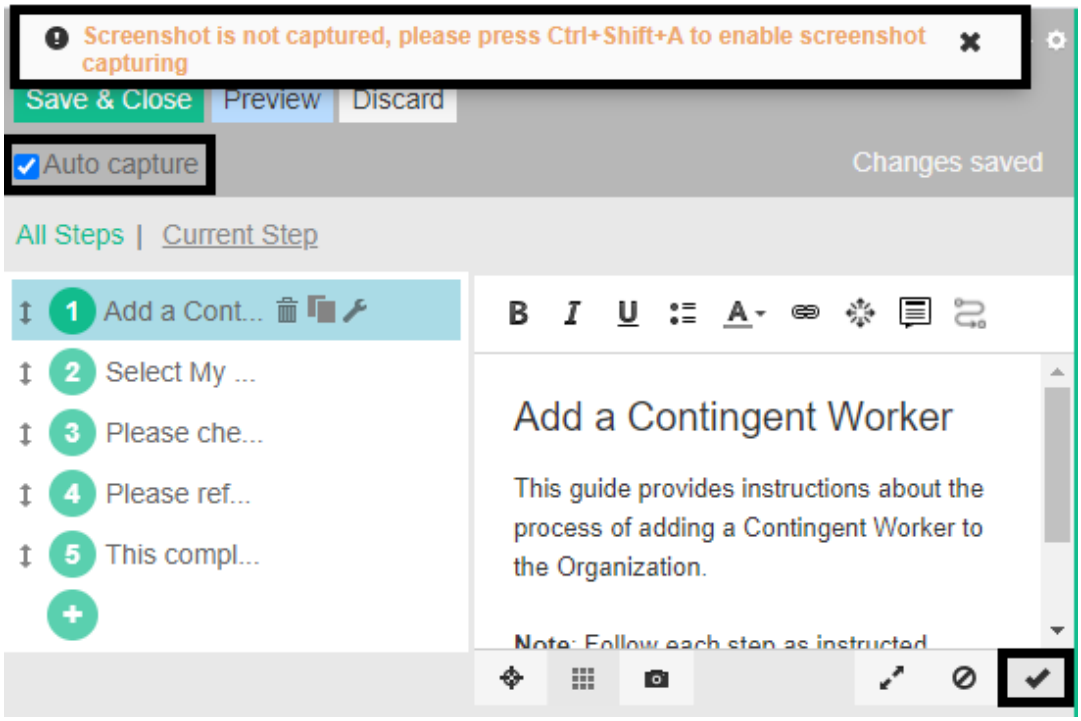
There are two methods to generate screenshots in OGL: auto capture and manual capture.

Method 1: Using Auto Capture

1. Open the guide for editing.
2. Press Ctrl + Shift + A to authorize screen captures.
3. Select the Auto capture check box if it is not already enabled.
4. Click the **Save and Next** button (bottom right option in the screenshot below).

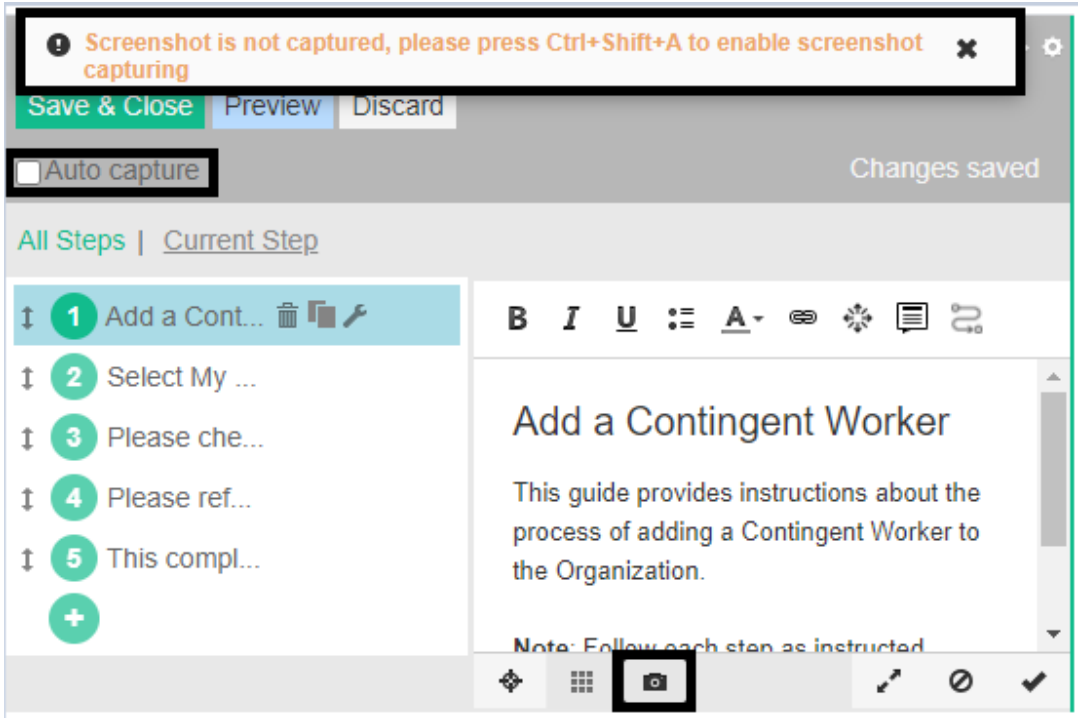
The screenshots are now automatically captured.

- To continue capturing screenshots automatically while reselecting the guide elements, use the **Save and Next** button.



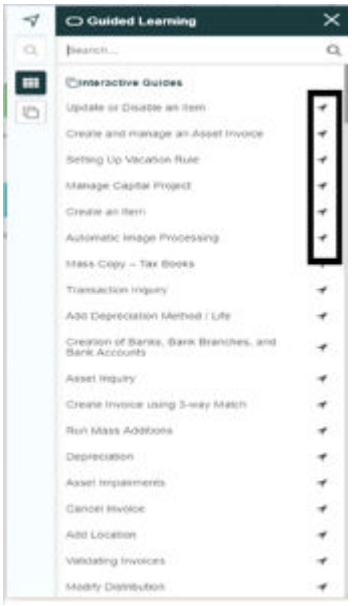
Method 2: Using Manual Capture

1. Open the guide for editing.
2. Press Ctrl + Shift + A to authorize screen captures.
3. Deselect the Auto Capture check box.
4. Click the **Camera** button (see screenshot below). Once this is clicked, the screenshot is manually captured.

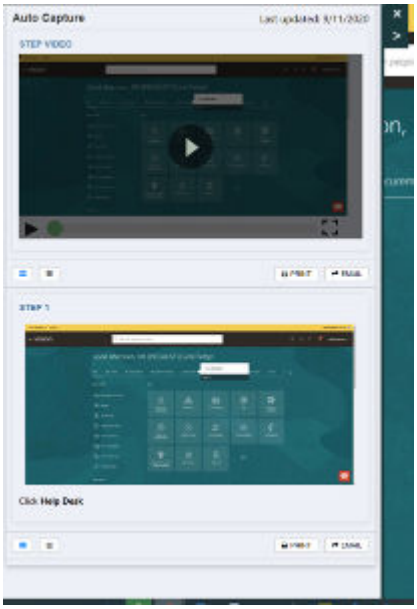


Changing Default “Vision” Screenshot

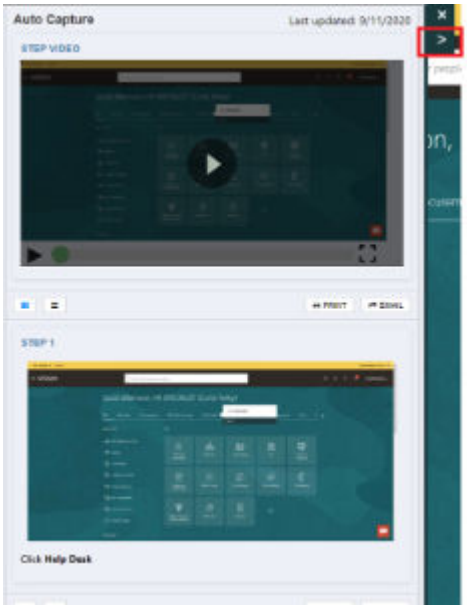
1. Log in to your cloud application.
2. Open the OGL Help Panel.
3. Click on the Step guide icon as shown below.



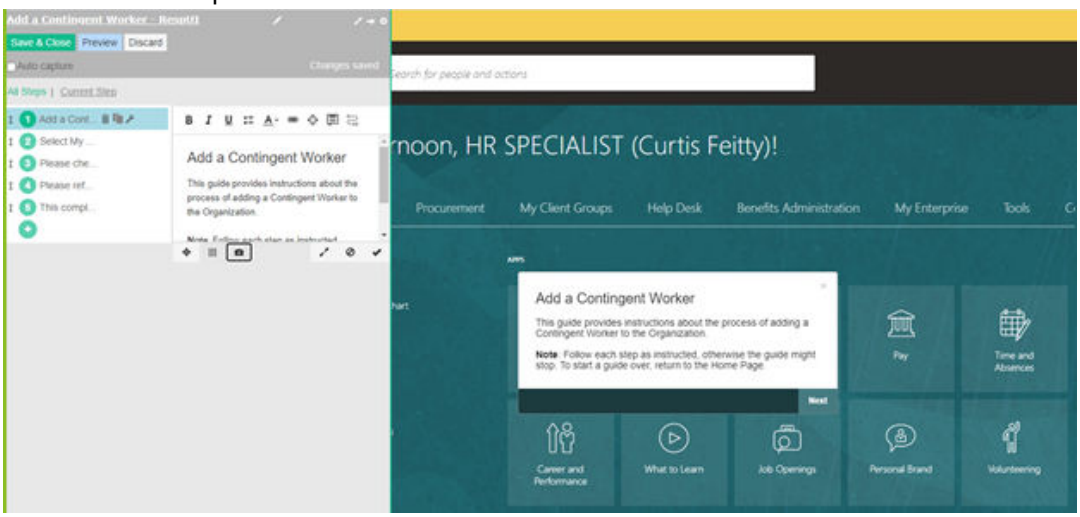
The Step guide appears as shown below.



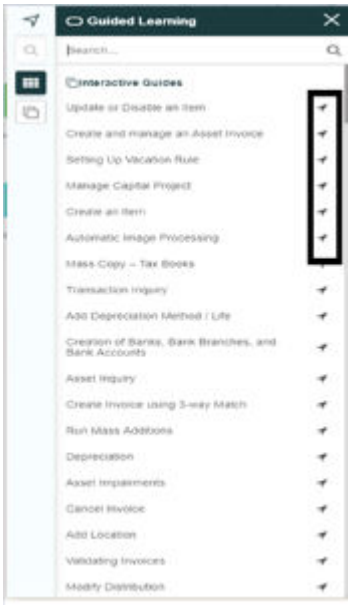
4. Click the unpin arrow (see the red box in the screenshot below) to detach the step guide and open it in a new tab. Having the base step guide open and available will make it easier to determine if you need to reselect target elements in your bae guide while recapturing screenshots.



5. a. Open the OGL Console (https://guidedlearning.oracle.com/account/#/).
   b. Open the guide for editing.
6. The OGL Editor opens as shown below:



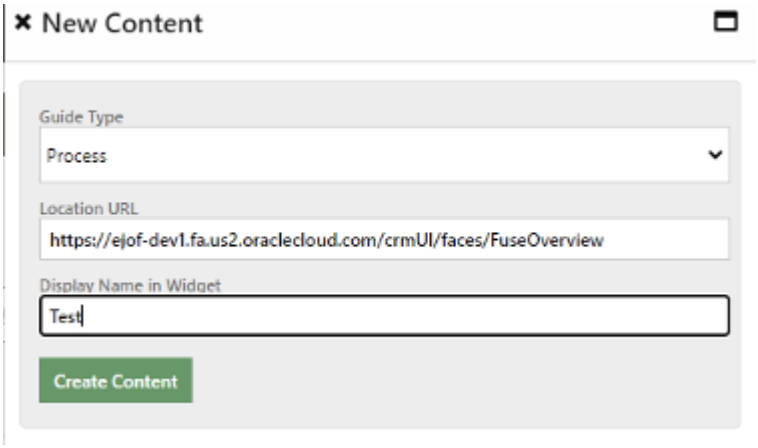
7. Press Shift+Control+A to enable screenshots
8. Choose your step where you want to change the screenshot
9. Using Method 1 or Method 2, you may select your element and click on the **Save and Next** or the **Camera** button
10. Repeat for other steps or tips where you want to change the screenshots
11. Click **Save and Close** to save the guide



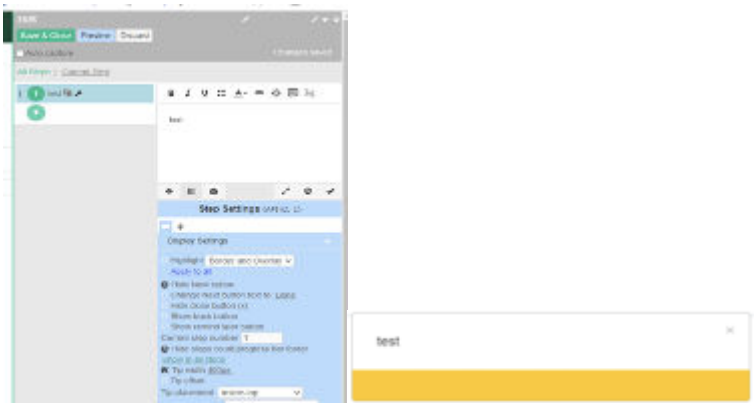
How to Increase the Tip Width of the Message in OGL

Workaround

1. Create a **Process** from the New content

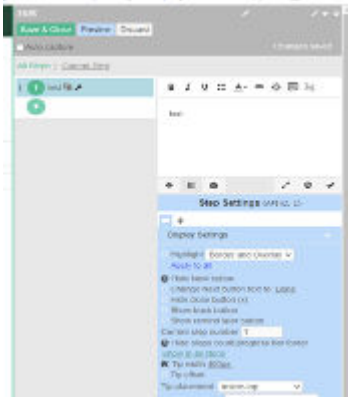


2. Create a **splash tip** and set the tip width as per requirement
Note: Update the step conditions as required. The page conditions can also be added in the guide activation setting.



3. Change the type from **Process** to **message** in the guide settings

4. Select the guide to **autoload** in the guide activation setting



Note: To change the tip width further, change the message type to process, update the tip width and revert the type to message.

How to Launch a Guide Based on User Interaction - Create a "Launcher Guide"

Don't wait for the users to *pull* the guide, *push* the guide to them:

- The best way to train a user (on a process) is when s/he uses the process for the first time.

- You want to share more information with the user about the feature he/she has just started using.

Guided Learning allows you to launch a guide based on a user's interaction with your application. Common use cases would be:

- Launch the 'create lead' guide as soon as the user clicked on the 'create new lead' button. Or more generally, launch a walkthrough guide as soon as the user clicked on a button.
- Launch a guide as soon as the user starts filling a form
- Launch a guide when the page contains something. For example, launch a guide if a common error message appears on the screen.

We call these guides *Guided Learning Launcher Guides*.

Normally, a launcher guide would have a single, invisible step that would wait for the desired user interaction. And a branch to launch the desired guide as soon as the user makes this desired interaction.

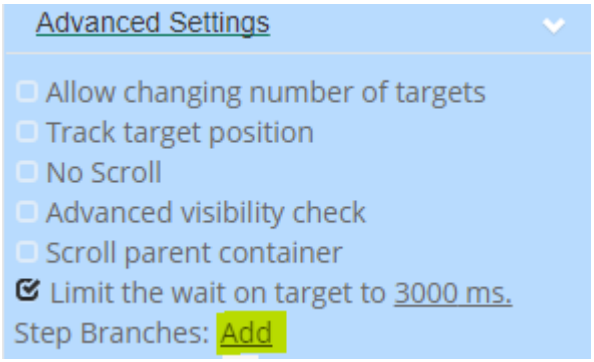
How to Launch a Guide from another Guide

There are several ways to link two guides together with Guided Learning.

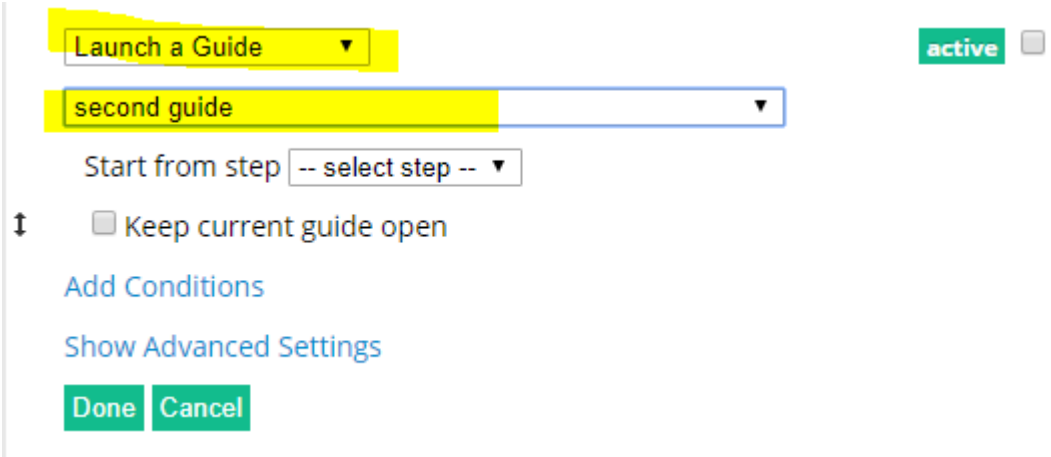
Launch a guide when the user completes another guide

To launch a guide when the user completes another guide we need to create a branch.

Select the last tip in your "first" guide. Open the ***Advanced Settings*** panel and click to add a new branch.



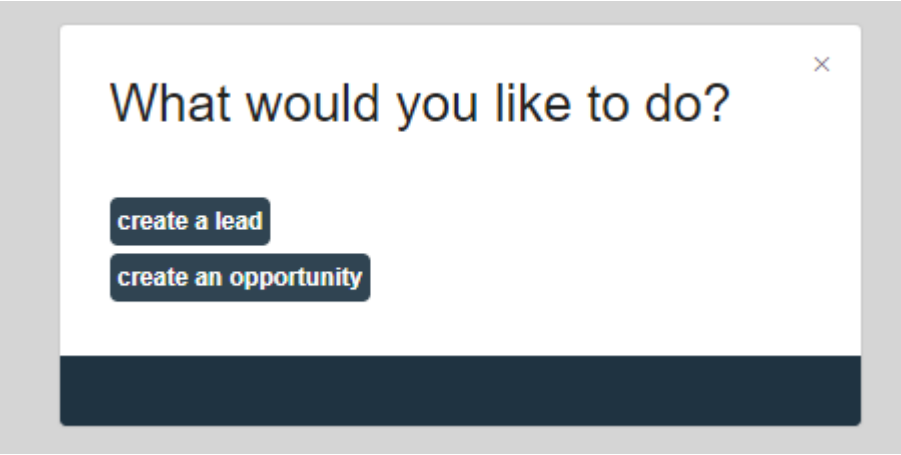
Add a 'Launch a Guide' branch and select your "second" guide from the drop down list.



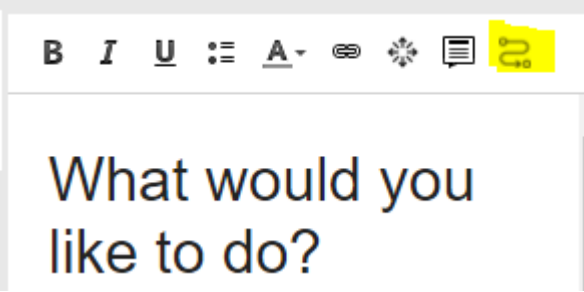
NOTE: You do not need to select a step. By default Guided Learning will start the guide from the beginning.

Launch a guide when the user clicks on a button in a tool-tip

Sometimes you would like the user to choose which guide they want to launch now. What if you would like to ask the user *"What would you like to do now?"*?



To create such an experience type the text of your tip (including the text of the options). Now select the text for the first option and click on the ***Launch a guide*** button.



In the popup, select the guide that you want to launch. You can choose whether you want the display to be a link or a button.

How to Make Tasklist Fixed at One Position

Issue: Tasklist was moving around the screen and difficult for users to find.

Instructions

Add the steps involved:

1. In design kit, search for the below code:

```
div.sttip.ir-todoList {

 display: inline-block;

 position: absolute;
```

2. Change the value for position from absolute to fixed as shown below:

```
div.sttip.ir-todoList {
```

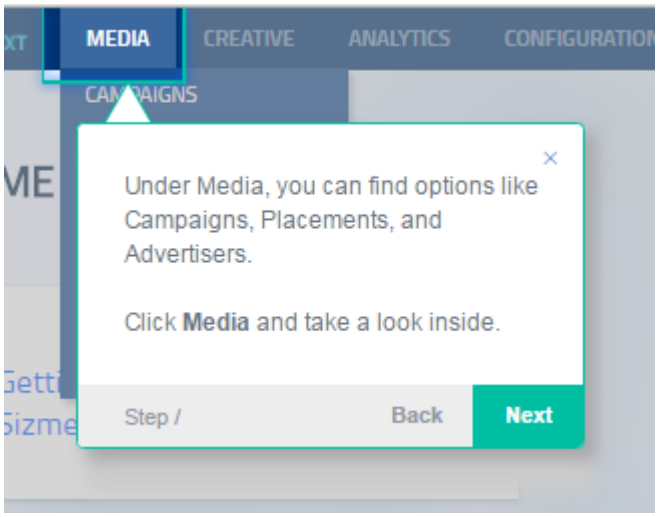
```
display: inline-block;

position: fixed;
```

How to Prevent a Tooltip from Hiding a Drop Down Menu

We often receive a request that sounds something like this:

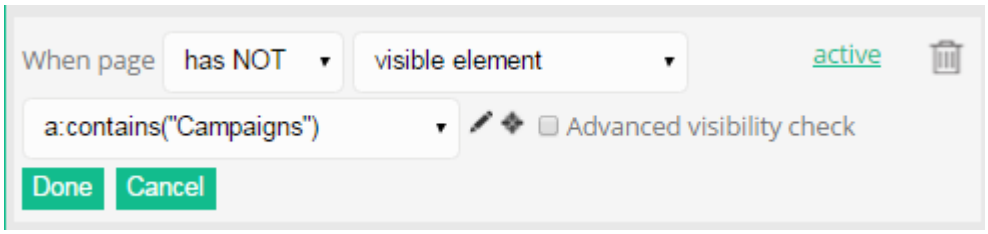
```
"The top navigation of my application uses hover menus.
When a user puts his mouse on a menu item a sub-menu is opened;
when the mouse moves out of the item the sub-menu disappears.
If I place a tooltip on the menu item, as soon as the mouse is over
the item, the tooltip covers the drop down and the user cannot see the sub-menu."
```



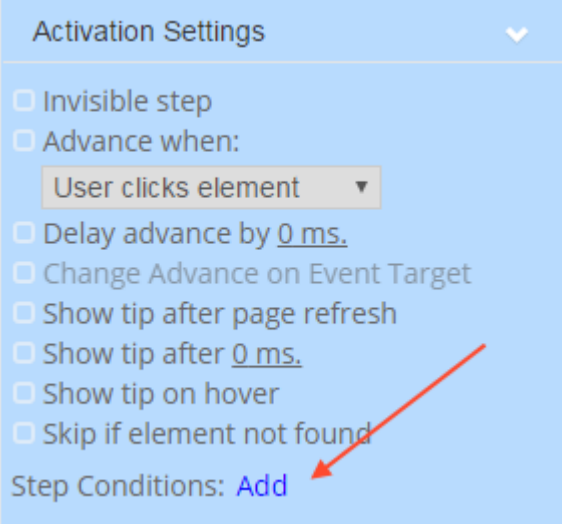
Notice how the tooltip hides the Media sub menu.

The best way to address this issue is using step conditions. Step conditions allow you to condition the display of any tooltip. In this example, we would like to tell Guided Learning to only show this tooltip if the sub-menu is not visible.

To add this condition, click the Add link under the tooltip's activation settings.



Once, clicked, add the following condition.



Notice how we are testing if the first item in the sub-menu (i.e. campaigns) is visible.

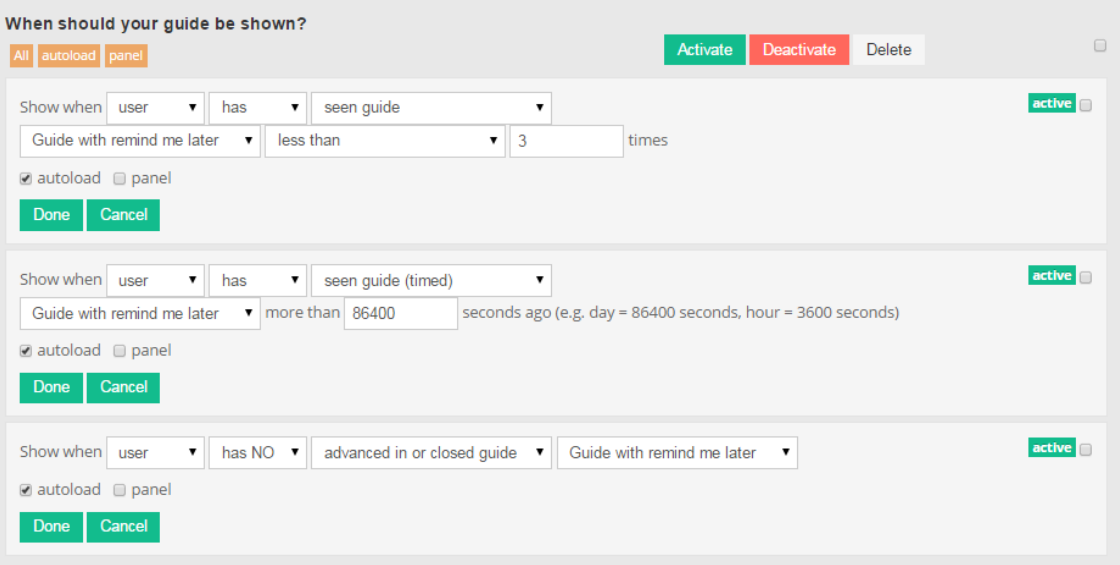
You can also amend the tip width or placement in the Display Settings. The tip width is measured in pixels or %. If just numbers are entered, it is pixels. For percentage, put a “%” at the end of the number.

How to Setup a "Remind me later" Button

Remind me later is very useful when launching a guide automatically.

There are 2 things you need to do:

1. Check the 'remind me later' setting in the first step of your guide.
2. Add the following activation conditions to the guide.



The first conditions determines how many times the user will be shown this guide. In the example above, if the user clicked 3 times on **remind me later** he will not see the guide again.

The second conditions determines what is the minimal time between 2 views of this guide. In our example the guide will not be displayed more than once per day (86400 seconds).

The third conditions makes sure that if the user has already engaged with the guide or closed it on purpose he will not see it again.

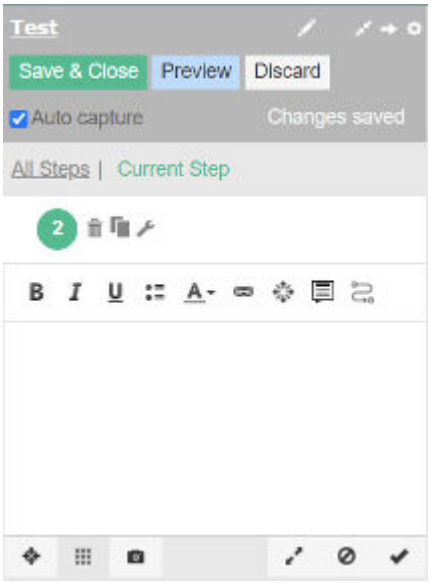
How to Work with Element Selection Settings

In OGL Editor, Element Selector Settings allow you to select Element Text, Element Title or Id Attributes of the page elements of the application you're working with.

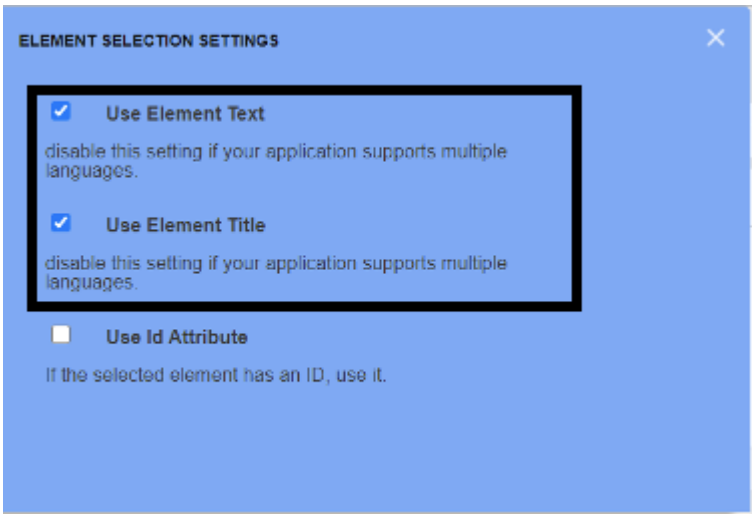
Managing Fusion Account for Only English (Default) Language

Element Selection Settings are unchecked by default in OGL Guide Editor for Fusion applications. When selecting an element in Guide Editor for a single language account, try to choose Fusion-specific selector. If there's no Fusion-specific selector available for the selected element:

1. Click the Settings gear in the top-right corner of the Editor

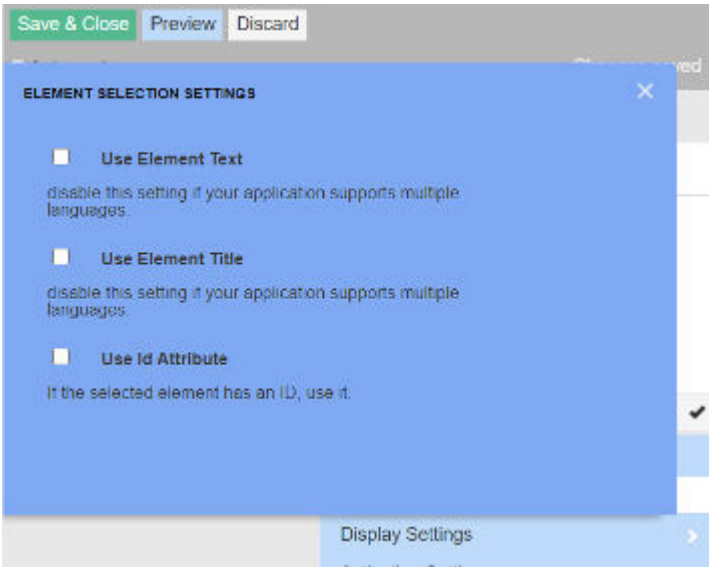


2. Enable 'Use Element Text' and 'Use Element Title' options.
This will ensure selection of Title and Attribute of the object and will result in fewer element selection issues during guide upgrades.



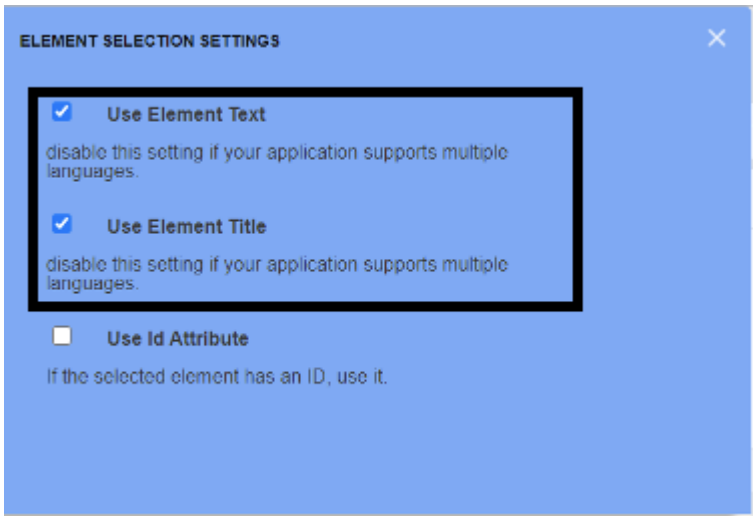
Managing Fusion Account for Multiple Languages

For multi-language accounts, it is recommended to disable Element Selection Settings. The selection of Title and Attribute of the objects will result in more element selection issues during guide upgrades because of difference between elements text/titles in default language (English) and other languages.



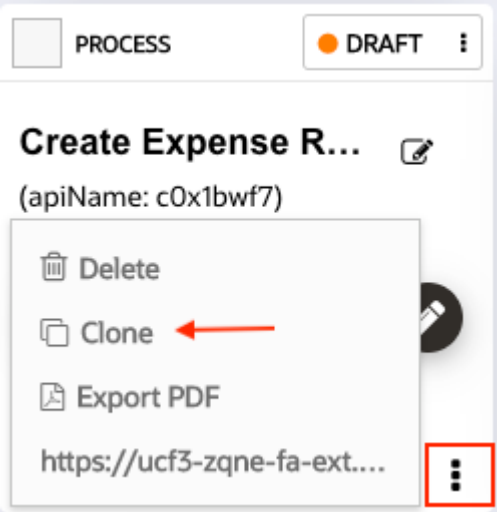
Managing Non-Fusion Accounts

For Non-Fusion applications, Element Selection Settings are enabled by default. When selecting an object, by default the Guide editor will look for the Title or Attribute of the Object. They are good for capturing the element because they tend to be constant and are unique in a page. This ideally is the best practice for choosing Selectors for Non-Fusion accounts.



I want to create a new guide that is based on another guide I have already created. Is this possible?

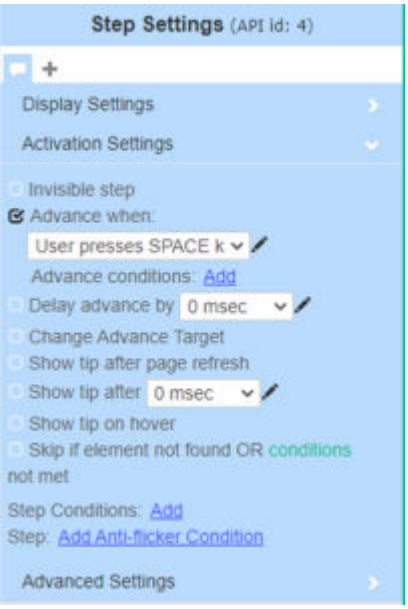
In the OGL Console, find the guide you want to clone, click on the ⋮ bottom right, then click on the Clone button, a copy of your guide will be created for you. Open it and make your desired changes.



I want to enable user to proceed upon clicking the spacebar

Guided Learning has a special event to capture clicking on the space bar as well as other major keyboard keys like ENTER and BACKSPACE.

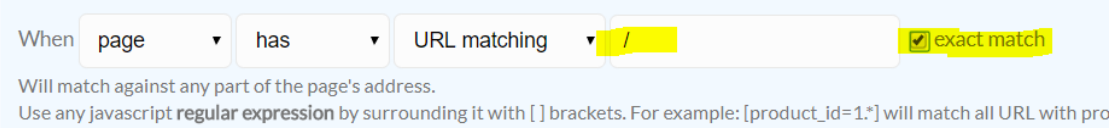
Under the '*Activation Settings*', you will see a drop down next to the 'Advance when:' option; select 'User presses SPACE key' and you're done.



I would like guide to launch only on the root "/" path of my website

To do this you will need to create the following URL condition.

'when page has URL matching /' and check the **exact match** setting.



Set up a Cross Domain/Cross App Guide

Cross-Domain

An application can have IFRAMES which are can be serviced from the same domain as the main page (also known as the top) or from a different domain. For security reasons, if an IFRAME is serviced from a different domain, the browser does not allow the OGL player to access it from the top window.

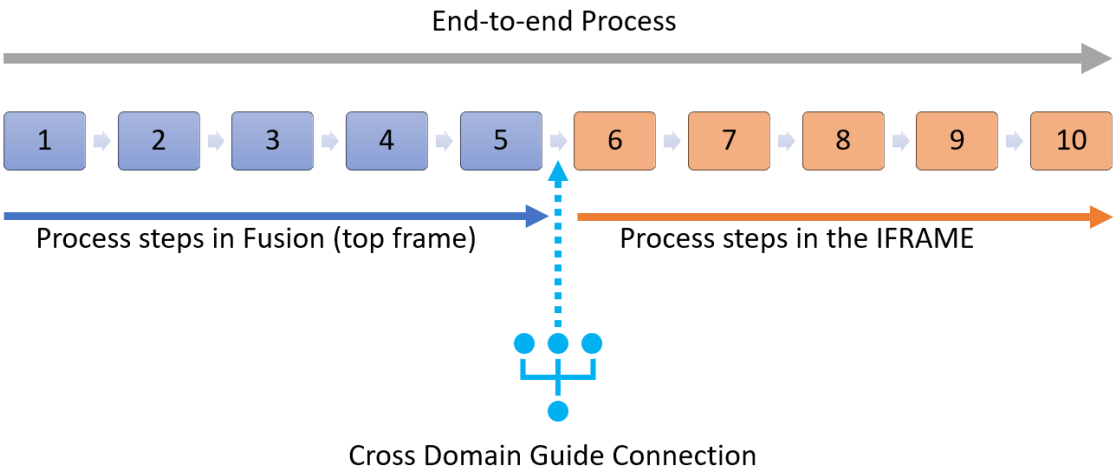
To create OGL solutions that can start from the top window and continue in the IFRAME, we need to create a communication channel for OGL between the top frame and the IFRAME.

To make this happen, the following prerequisite steps have to be carried out:

1. Request the correct script for the main application (if applicable)
2. Request the OGL script for the IFRAME (Ensure **iridize.stateOnly=window != top ? true : undefined;** is included in the JavaScript for the IFRAME and the main application)
3. Embed/enable OGL in the top frame (the main application i.e. Fusion)
4. Embed OGL in the IFRAME (Note: if there are several IFRAMEs being serviced from different domains, each of these IFRAMES need to have the OGL script embedded)

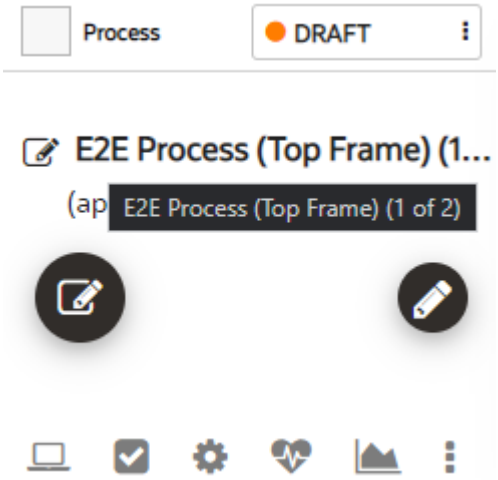
How to Set-up a Cross-Domain Guide

Let's say we have a process guide with 10 steps as an end-to-end process, with the first five steps being executed in the top frame, and the last five being executed in the IFRAME as illustrated below:



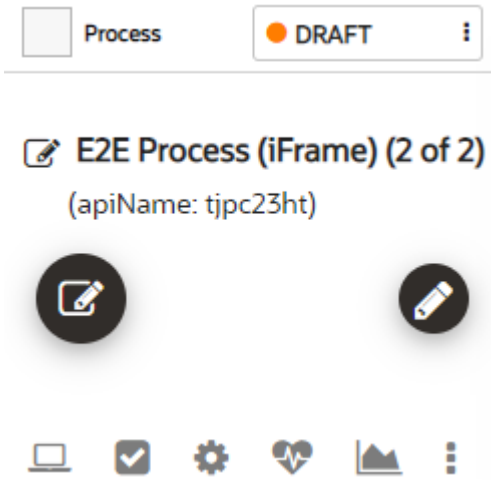
From the illustration, we see that we need to set up a **Cross-Domain Guide Connection** between steps 5 and 6 in the E2E Process. To enable this, we build the E2E process guide in parts:

1. First, build part of the process to run on the 1st application as one guide (Steps 1-5).

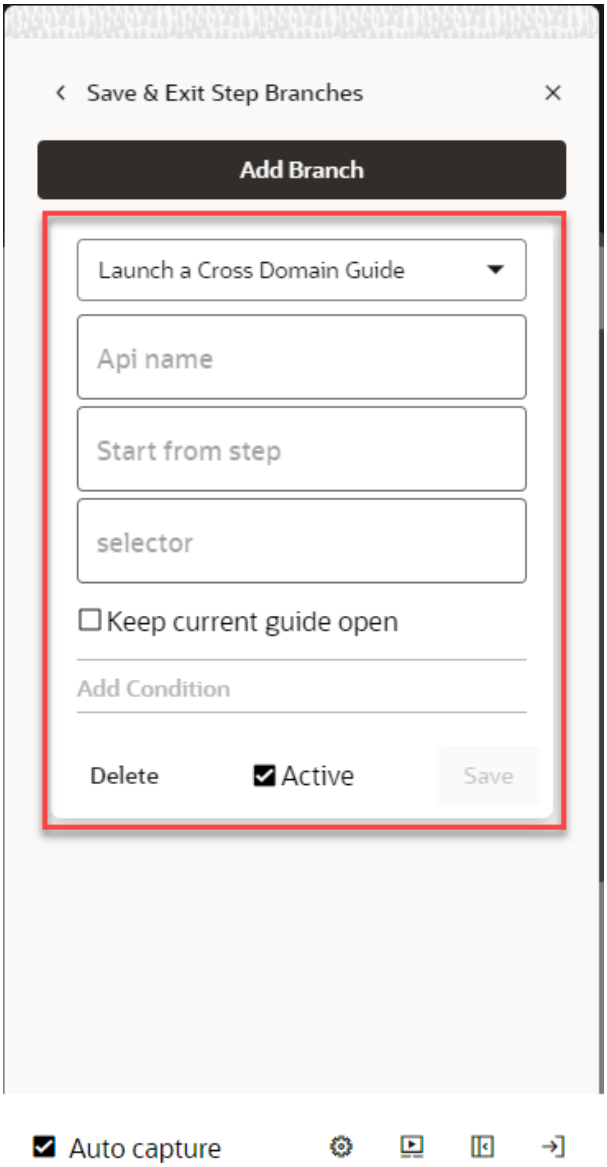


2. Then build the part of the process to run on the 2nd application (Steps 6-10).

> **Note:**
>
> Take a note of the apiName.



3. Open the top frame part of the guide in the OGL Full Editor
4. Select the last step in the guide (i.e. step 5 in our example)
5. Select **Step Settings**
6. Expand the **Advanced Settings** section
7. Select **Add** next to **Step Branches**
8. Select **Add Branch**



9. Select **Launch a Cross-Domain Guide** from the dropdown.
10. Enter the **apiName** of the iframe part of the guide
11. Enter the **Start from** step
12. Enter the selector for the IFRAME (*See IFRAME Selector Guideline below*)
13. Select **Done** to save the branch.
14. Save and close the guide

    Once these steps have been completed, the user executes the guide (top frame part of the guide), the iframe part of the guide will be executed when the user moves past the step with the branch.

    Key considerations

    - Determine if the iframe part of the guide should be in the OGL Help Widget or not)

- Determine when and how the step with the branch advances (i.e. advance on click etc.)
- Determine how to track these guides in analytics
- Determine a clear documentation strategy for maintenance

IFRAME Selector Guideline for Cross-Domain BranchesBasic Idea on Selectors

- attr^=val : gets an attribute starting (^) with val.
- attr*=val : gets an attribute containing (*) val.
- attr$=val : gets an attribute ending ($) with val.

For example:

- iframe[src*="/servlet"] returns an iframe with src containing “/servlet” in any part of the string.
- iframe[src^="/5001"] returns an iframe with src starting with “/5001”

How to get the proper selector
1. Do an inspect element on the iframe.

2. Find the iframe src attribute.

3. Use a subset of its value. For instance: The iframe “File Attachment” has the src“ /5001 F000002jdbq ?isdtp =vw&isWsVw=true&nonce=2014e000ccc4c41384944ea28a38fe591b2b936d9b48d5d99f2 bdffda916bfd5&sfdcIFrameOrigin=https%3A%2F%2Fkphr--trn1.cs90.my. salesforce.com”

- The following three selectors can be use for the iframe above.
  a. iframe[src*="?isdtp"]
  b. iframe[src^="/5001"]
  c. iframe[src$="my.salesforce.com"]

Always pick the selector that looks more likely to identify a single iframe. For instance, by choosing 2 or 3 there is a higher chance of referring to more than one iframe, therefore the cross-domain call could fail.

- 

In case of nested IFRAMES (iframe inside another iframe)

use '>>' as a separator eg: iframe[src*="AdfLoader"] >> iframe[src*="PageLoader"]

Cross-Application
An application can start in one application (i.e. Fusion) and continue in another application (i.e. CPQ). Normally, this means a new browser tab will be opened when moving from the first

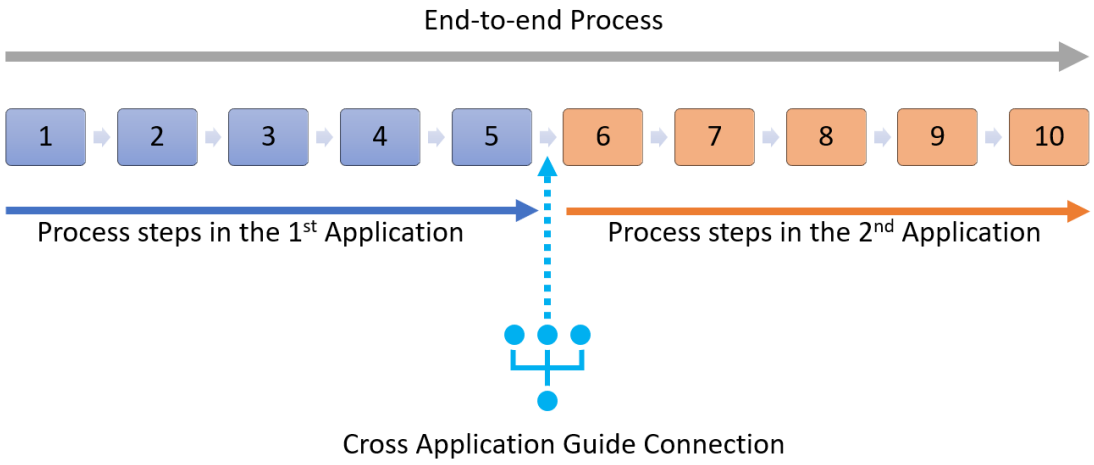
application to the second. Similarly, we need to create a communication channel for OGL, we achieve this by embedding OGL in both applications.

The following prerequisite steps have to be carried out:

1. Request the correct script for the main application (if applicable)
2. Request a new appID and the OGL JavaScript for the 2nd application (Ensure **iridize.crossAppIframe=true** is included in the JavaScript)
3. Embed/enable OGL in the first application (i.e. Fusion)
4. Embed OGL in the 2nd application

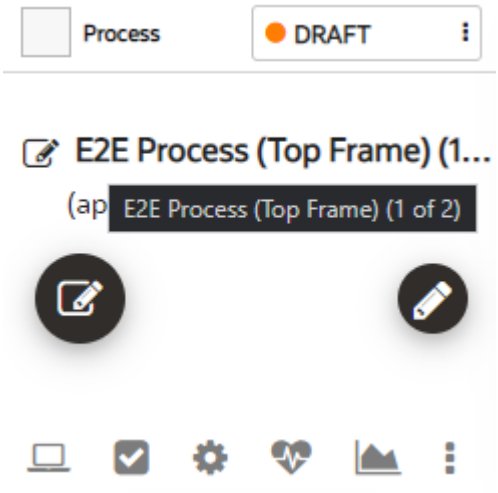
How to Set-up a Cross-Application Guide
Similar to a cross-domain guide, a cross-application guide has to be built in parts. the key difference is that for cross-application, the parts cannot be in the same OGL APPID.

Let's say we have a process guide with 10 steps as an end-to-end process, with the first five steps being executed in the main (1st) application, and the last five being executed in the 2nd application as illustrated below:



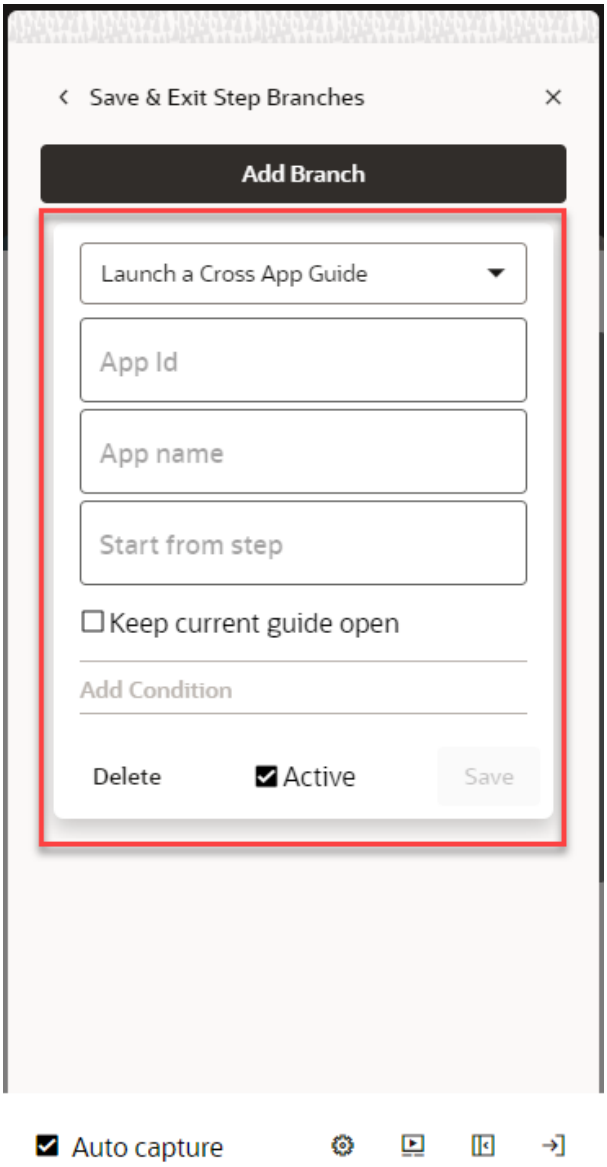
From the illustration, we see that we need to set up a **Cross-Application Guide Connection** between steps 5 and 6 in the E2E Process. To enable this, we build the E2E process guide in parts:

1. First, build the top-frame part of the process as one guide (Steps 1-5) in one appID.



2. Then build the iframe part of the process (Steps 6-10) in the second appID.

3. Go to the first part of the process guide
4. Open the guide in the OGL Full Editor
5. Select the last step in the guide (i.e. step 5 in our example)
6. Select **Step Settings**
7. Expand the **Advanced Settings**section.
8. Select **Add** next to **Step Branches**.
9. Select **Add Branch**



1. Select **Launch a Cross App Guide** from the dropdown
2. Enter the **appID** of the 2nd application
3. Enter the **apiName** of the guide in the 2nd application (this is the second part of the process guide)
4. Enter the Start from step
5. Select **Done** to save the branch
6. Save and close the guide

Once these steps have been completed, the user executes the guide (in the first application), the second part of the guide will be executed when the user moves past the step with the branch.

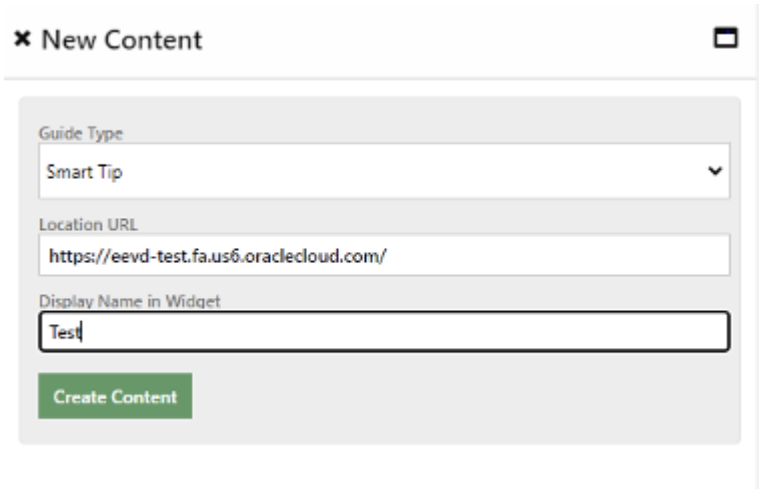
Key considerations

- Determine if the iframe part of the guide should be in the OGL Help Widget or not)
- Determine when and how the step with the branch advances (i.e. advance on click etc.)

- Determine how to track these guides in analytics
- Determine a clear documentation strategy for maintenance

Smart Tip and Beacon Best Practices

1. Create a **Smart tip or Beacon** guide from the New Content



Configuration:

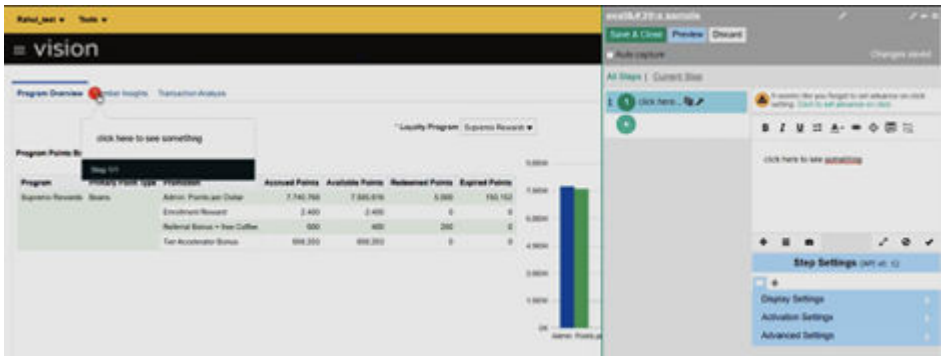
Guide Type → Select Smart Tip from the drop down

URL → Based on customer instance

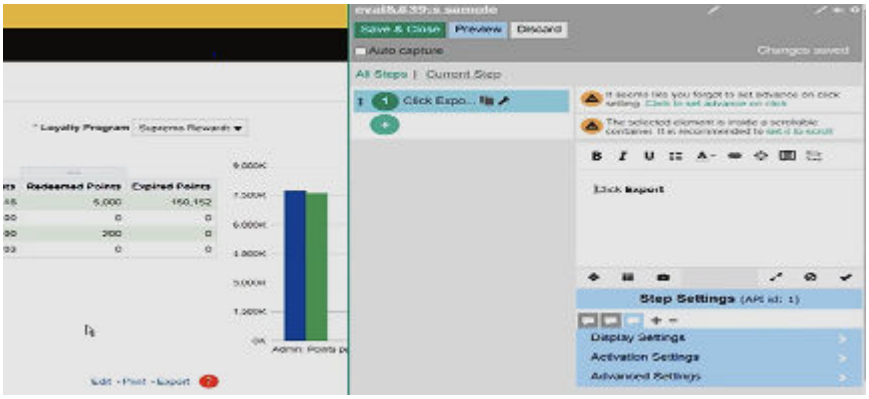
Display Name → According to requirement

2. **Smart tip:** A tip which displays when a user mouses over **?** to provide more information about the field.
Do: All the tool tips within the smart tip guide should be confined to a specific page.

Don't: Create tool tips which are related to two different pages in the single smart tip guide.

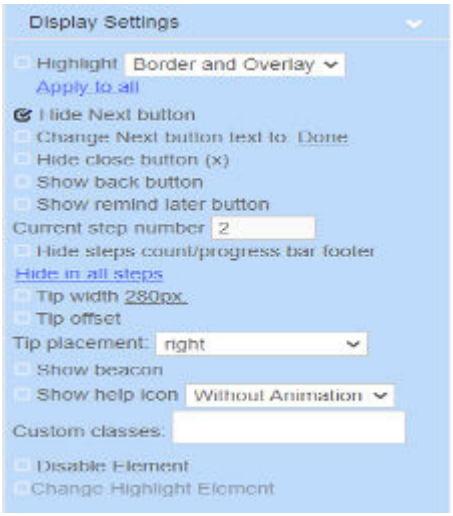


3. If two or more smart tips need to be created on one page, **create all the smart tips in one step as multi tip**

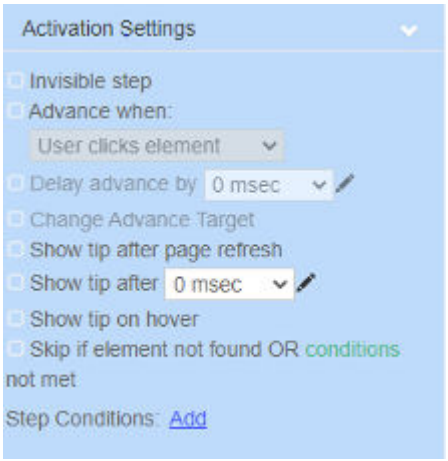


Display Settings for Smart tip or Beacon guides:

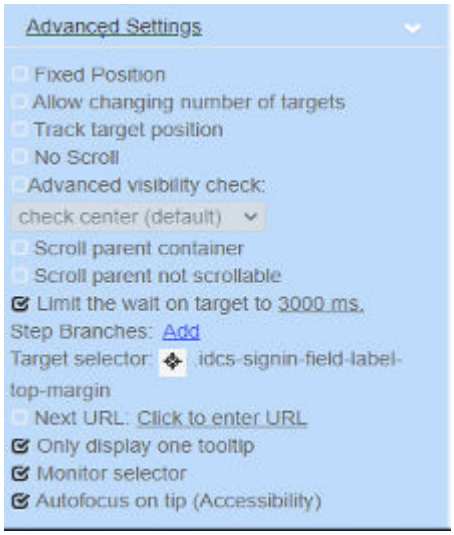
- **Highlight** → Disable
- **Hide Next button** → Enable
- **Show Back button** → Disable
- **Hide steps count/progress bar footer** → Disable



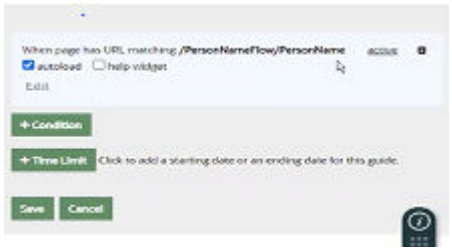
Activation Settings: No Changes to the default.



Advanced Settings: No Changes to the default.

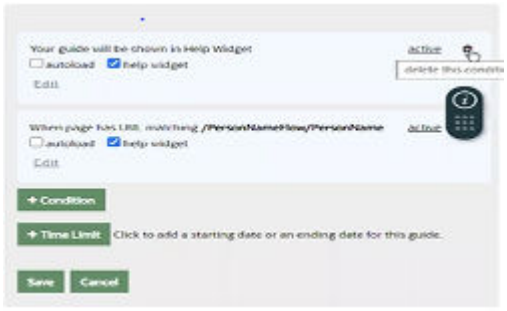


4. The page should be defined in the **activation condition** with **Autoload** turned on



Mandatory: Always have a page URL condition mapped to the smart tip where it should appear.

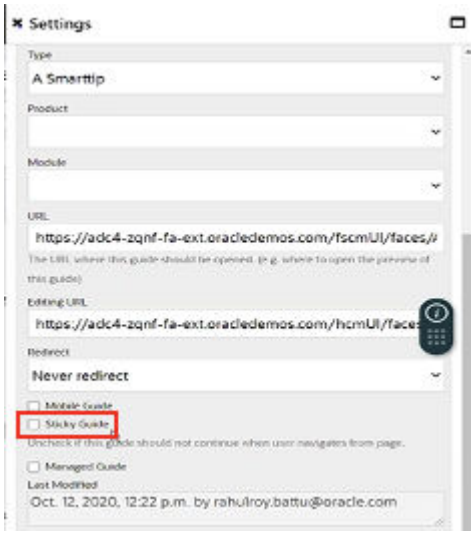
5. Remove any other condition under activation other than page URL Condition



Note: We can certainly add role segmentation for smart tips as desired.

For a **Non-Fusion application**, we need to manually take the URL and add it as a condition in Activation.

6. Ensure both **Smart tip and Beacon** are *not* set to **Sticky Guide** by unchecking the option



Note: Enabling Sticky Guide for an Auto Load smart tip causes a performance issue.

So if my application has a form and the user clicks on close instead of submit, I would like for the guide to end then. Is that possible?

Yes, it is most certainly possible:

1. On the step where you are expecting the user to click submit,
2. Create an invisible tip that would advance when the user clicks on the cancel button
3. Add a branch to this new tip that would close the guide. The branch is activated as soon as the tip is 'complete' which in our case would be when the user clicks on the cancel button.

Step Conditions and Page Conditions

Step Conditions control when a certain step will be activated. Normally this means when to display a certain tooltip. For an invisible step, they will normally control when the guide should advance to the next step. The types of step conditions are identical to Page type conditions available in the guide activation conditions.

This article describes all types of conditions that you can create using the Guided Learning editor.

For more details on the underlying concept, please refer to the next article, Step Conditions Explained.

URL Matching

URL Matching will match against any part of the page's address.

You can use regular expressions as well. Use any javascript regular expression by surrounding it with [ ] brackets. For example: [product_id=1.*] will match all URL with product_id's that starts with 1.

Use the "exact match" check box if you want the entire path to match exactly with the provided value. Note that the domain (e.g. http://www.iridize.com) will be ignored in this evaluation.

URL Parameter

Will match against a specific URL parameter. You can either test that the specific URL parameter exists or evaluate its value.

Variable

Will match against a javascript variable. You can either test the existence of a variable or test its actual value.

Cookie

Will match against a browser cookie. You can either test the existence of a cookie or test its actual value.

Session Variable

Will match against a special Guided Learning variable. The Guided Learning javascript API allows setting session variables that exist only throughout the duration of the currently logged in user. You can then condition your guides to display based on these variables.

Element

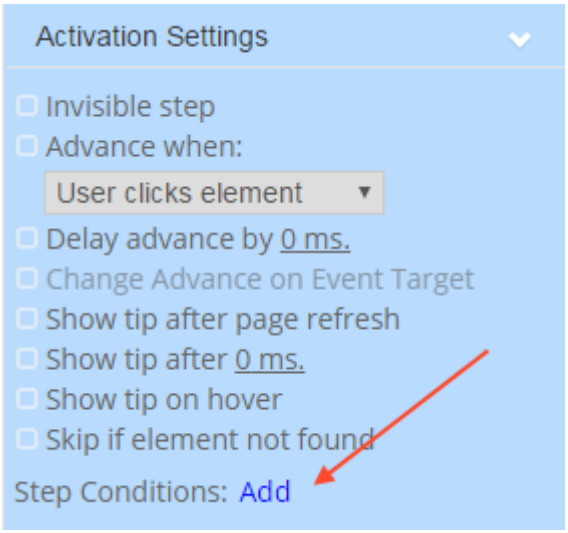
Will match against any element on the page's Document Object Model (DOM). You can also extend this condition by telling Guided Learning to wait for multiple objects matching the provided selector.

Visible Element

Will match against a visible element on the page's Document Object Model (DOM). This is more restrictive than an element condition because in HTML an element can be invisible.

Step Conditions Explained

Step Conditions control when a certain tooltip will be displayed.



Before displaying a tooltip, Guided Learning will iterate over all of the step conditions; only if all conditions are met will the tooltip be displayed. **If any condition is not met Guided Learning will not show the tooltip.**

Being that web applications are dynamic in nature, Guided Learning is continually testing all conditions; as a result, your tooltip can be displayed and hidden multiple times based on how the user interacts with the page.

Example Use Case

Let's say you have a social share button in your application. To encourage your users to click on this button, you add a tooltip to point directly at it. But, when the user clicks on the share button a popup window appears; when it does you will probably want the tooltip to disappear. When the popup is closed you would like the tooltip to display again.

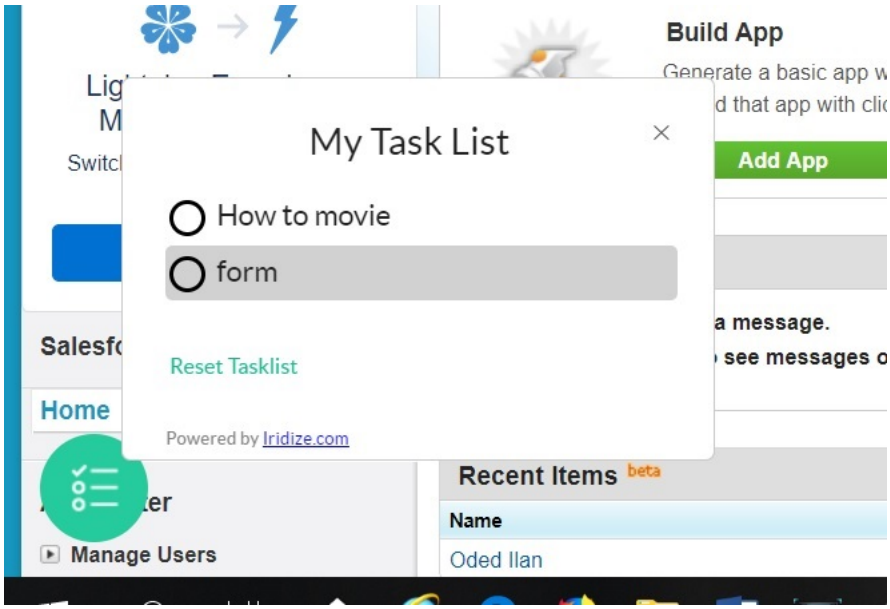
No problem!

What you need to do is to add a step condition to this tooltip. The step condition should say "Show this tooltip when the page does not have a **visible** share popup".

Adding Step Conditions

- **https://docs.oracle.com/en/education/oracle-university/guided-learning/user-guide/creating-ogl-content.html#GUID-1F27177C-4A85-42C8-ABDF-7E29FED9E577**

Task List / Onboarding Widget



The Improved Task List allows you to offer your users a *personalized* list of guides, videos, and links, which are presented as one group, and allow them to see how far they have advanced in completing/viewing the list.

Task Lists can be used for:

- Training flows that walk your users through a new software platform
- Onboarding sequences that train and certify new employees
- Pre-defined checklists the users need to complete while working on a certain process
- And more…

Guided Learning's Improved Task-List has several special features that make it a very powerful tool for onboarding, training, and assistance. With our Improved Task List, you can:

- Create unlimited Task-Lists per product

- Create a Task List for each type of user, based on role, past activity – or any other rules you wish to choose
- Allow your users to reset the task list progress

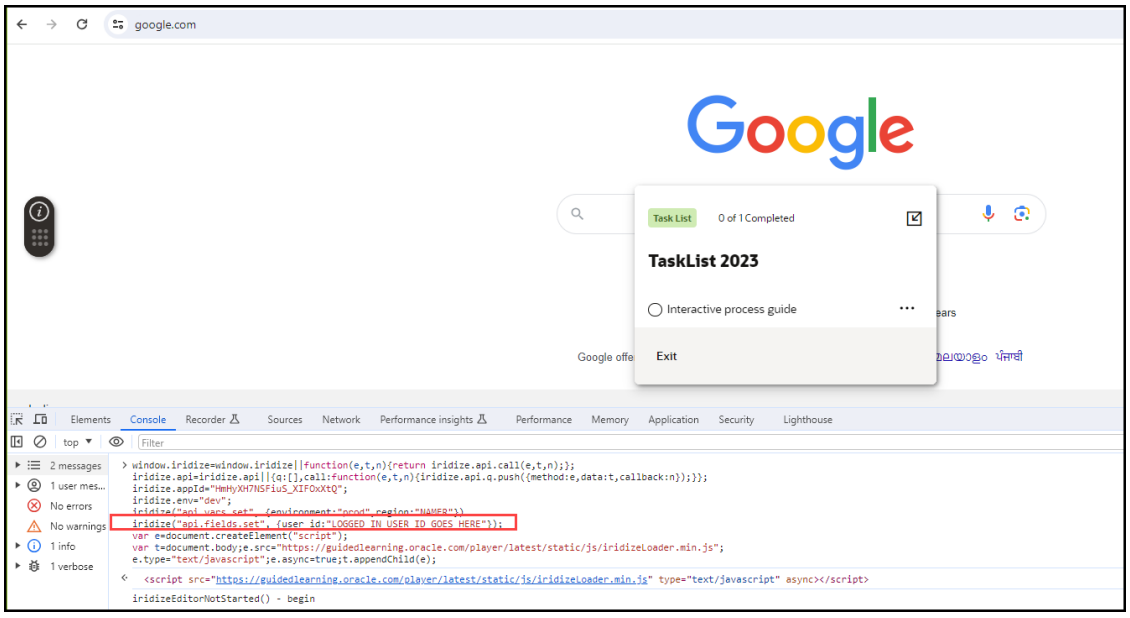
The Task List Experience

An end-user can access and launch tasks lists in two methods:

- Launch the List from the Task List icon which floats on the bottom right of your screen
- If the List is set to be available from the Help Widget you can search for it and launch it from there.

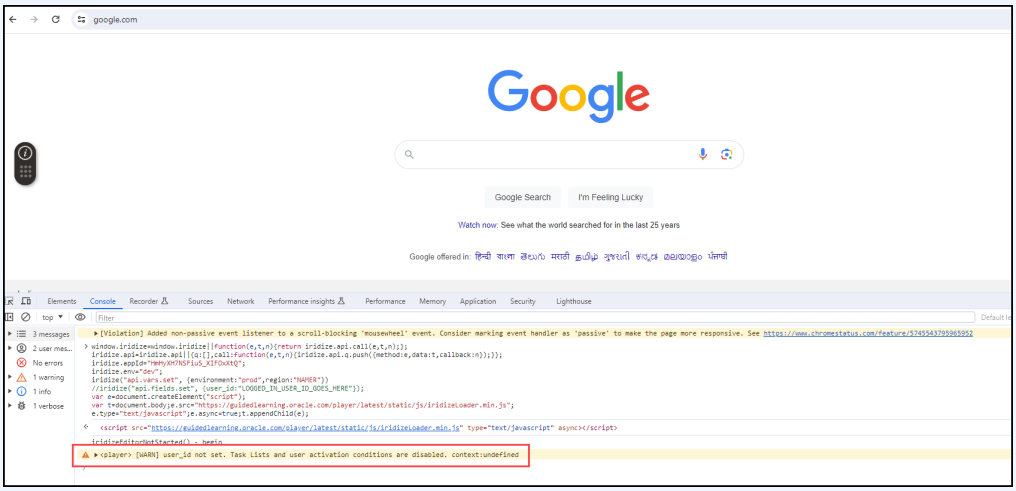
The Task List Experience

An end-user must set the User ID in the javascript to enable the Task List.



Note:

If the user ID is not set in the Javascript, it throws an error and the task list does not get displayed.



Creating a Task List

- https://docs.oracle.com/en/education/oracle-university/guided-learning/user-guide/creating-ogl-content.html#GUID-BA7681F2-BE0C-4A3E-B888-DC8FF03D217A

Guided Learning's new Task List is a very powerful mechanism for helping your users find their way around a new project.

Understanding Guide Options

- https://docs.oracle.com/en/education/oracle-university/guided-learning/user-guide/creating-ogl-content.html#GUID-6F20CFC7-79CE-4FC1-B1B0-86CF2253500C

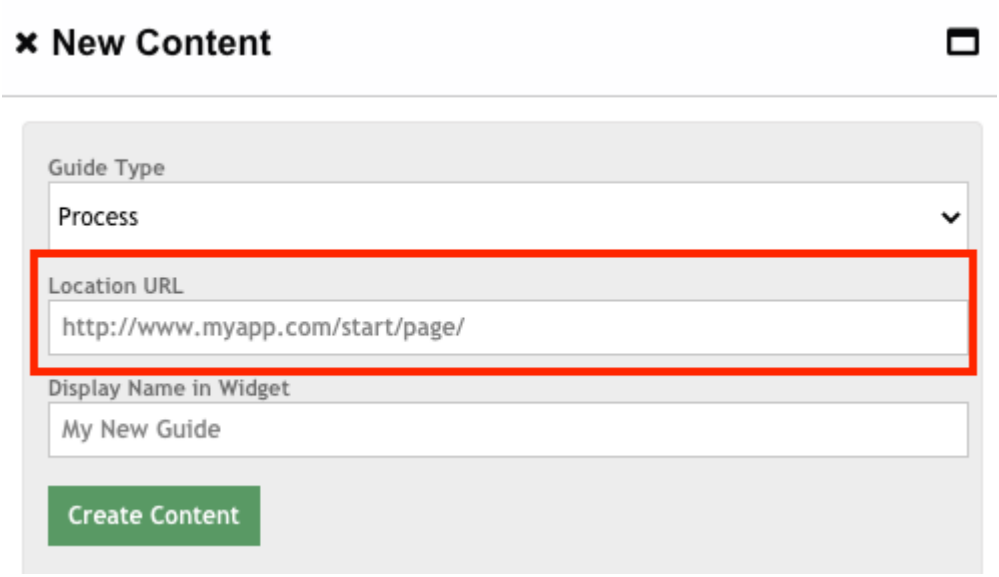
URL Configuration for a Guide

There are several URL parameters that you can configure to fine tune and customize the behavior of any walk-through guide.

In this article we will go over all of them in detail.

URL

You explicitly set this URL when you enter a URL and start creating a guide.



The URL controls the following:

- Unless you manually set the Editing URL (see below) this will be the where Guided Learning will take you to edit the guide when you click on the guide's name.
- It is the URL you will be redirected to when you **preview** the guide.
- If this guide appears in your help widget and you have configured the guide to *Redirect if on different page than URL*, than when the guide is clicked in the help widget Guided Learning will automatically redirect the end user to this URL.

Normally this will be a link to your staging environment where you create and test your guides before rolling them out to production.

You can change the URL on the guide options page under Advanced Options.

Editing URL

By default this will be identical to your the URL above. But if for any reason they should not be the same you can change it.

This URL has no bearing on where your guide will be seen by your end users

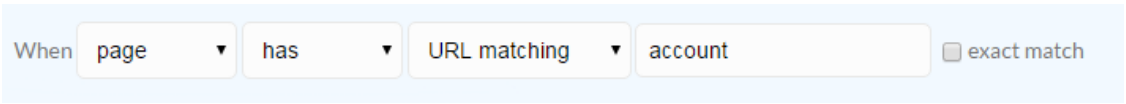
You can change the editing URL on the guide options page under Advanced Options.

Activation URL

This URL tells Guided Learning on which URLs to display the guides in the help widget and on which to launch the guide automatically (aka autoload).

Unlike the first two URL types there can be multiple Activation URLs. Furthermore you can use regular expressions to create URL wildcards.

Activation URLs are configured via the 'URL matching' condition type in the guide activation conditions.



Using Branches for Form Validation

Branches are a very powerful tool for changing the flow of a guide when filling out a form.

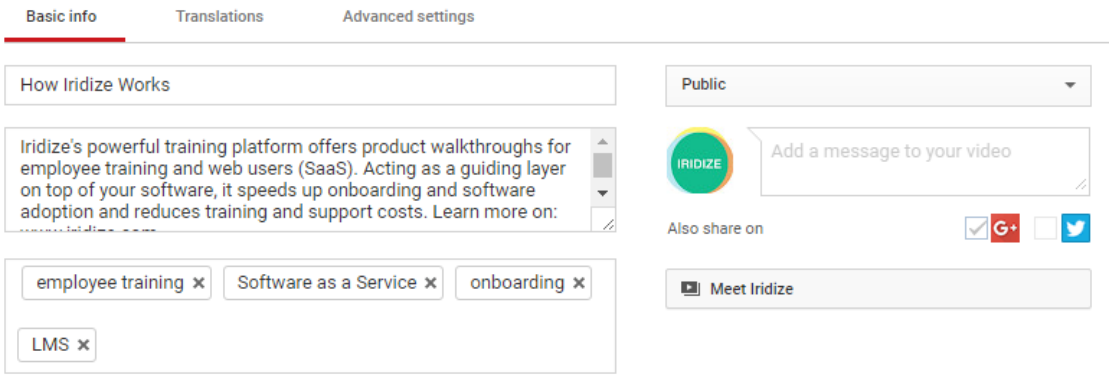
But there is much more to branches and forms. Guided Learning allows you to use branches to validate user input.

- What if the user did not make the expected selection in the drop-down menu?
- What if the user had left a field blank?
- What if the user had entered a password that is too short?

Combining **branches** with the Guided Learning **conditions framework** gives you the power to **verify user choices**.

Let's see how to do this with Guided Learning.

All of the examples are based on the YouTube form for editing a video's basic information.



Example 1: Enforcing a Selection

For this example, let's make sure that the user sets the video to be Public.

Our guide will look something like this:

1. Select Public from the Privacy Settings dropdown
2. Great, you've selected Public

In order to make the user select Public, we will create a loop so that if Public had not been selected we will remain on step #1 instead of advancing to step #2.

The natural flow of any walkthrough guide is to move from one step to the next (i.e. from step 1 to step 2, from 2 to 3, and so on). In order to break this flow, we will add a branch. Before naturally moving to step #2, Guided Learning will inspect all branches and if all of the conditions for a certain branch are met that branch will be taken.

In our example,

If the user did not select Public from the drop-down menu we want the guide to go to step #1. First, we will set up the branch (jump to the first step in the guide).

Then we will click to Add/Edit the condition:

Guided Learning is fully aware of what a drop-down menu is; when selected it will automatically choose the selected value and mimic the drow-down so that you could select the value of choice.

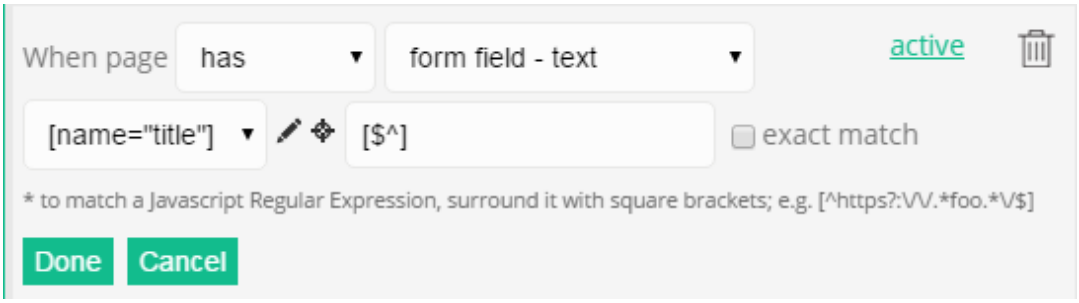
Note that if the user did select Public, our branch conditions will not be met and Guided Learning will default to the natural flow and advance to step #2.

Example 2: Disallow Empty Field

For our second example, on the same Youtube form, let's make sure the user selects a title for the video.

Much like in the previous example, our guide will have the following 2 steps:

1. Name your video.
2. Great! Name given.
   The branch is going to be very similar. In fact, the only difference is going to be in the condition. Here, since we are not testing against a drop-down menu we will create the following condition:



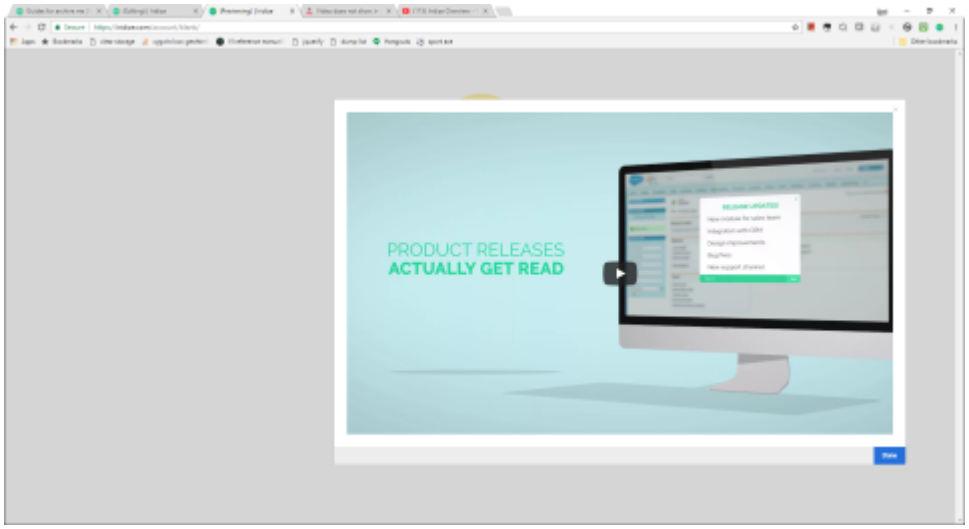
The strange-looking **[$^]** is the regular expression way of saying "empty". While this is somewhat complex, the fact that Guided Learning uses a regular expression engine allows use of very powerful text conditions like:

- has the word acme or corp - **[acme|corp]**
- at least 8 characters long - **[\w{8}]**
- only numbers and letters (e.g. no special symbols) - **[^\w+$]**
- capital first letter and only small letters after - **[^[A-Z][a-z]+$]**

The surrounding [] are for letting Guided Learning know there is a regular expression inside.

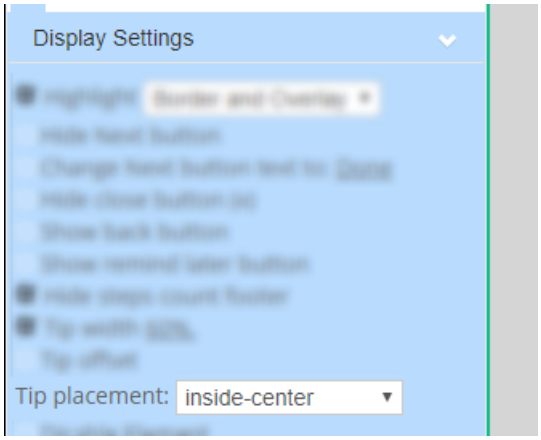
Video does not Align in Center

Does your tooltip load like this (misaligned toward the left or right of the screen)?



To resolve this, please open the guide in the editor.

Go to the display settings and under ***Tip placement*** choose ***'inside-center'***.



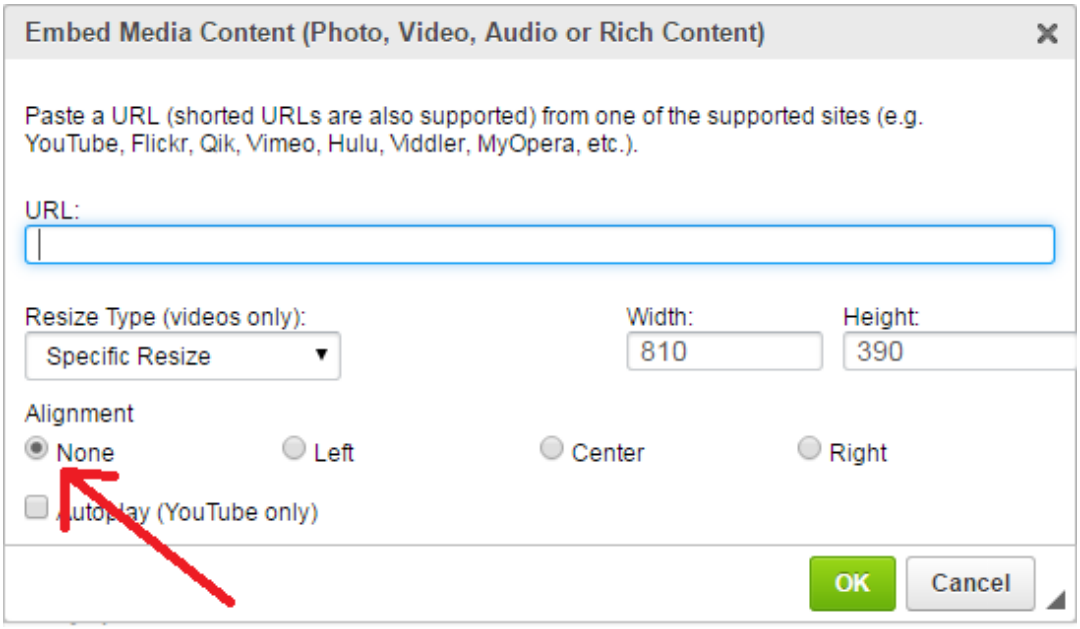
Video not Showing in Tip

By default, Guided Learning videos are set up to be responsive and fit on any screen.

When taking advantage of this responsiveness, the alignment option (see image below) is obsolete.

When embedding a video you have 2 options:

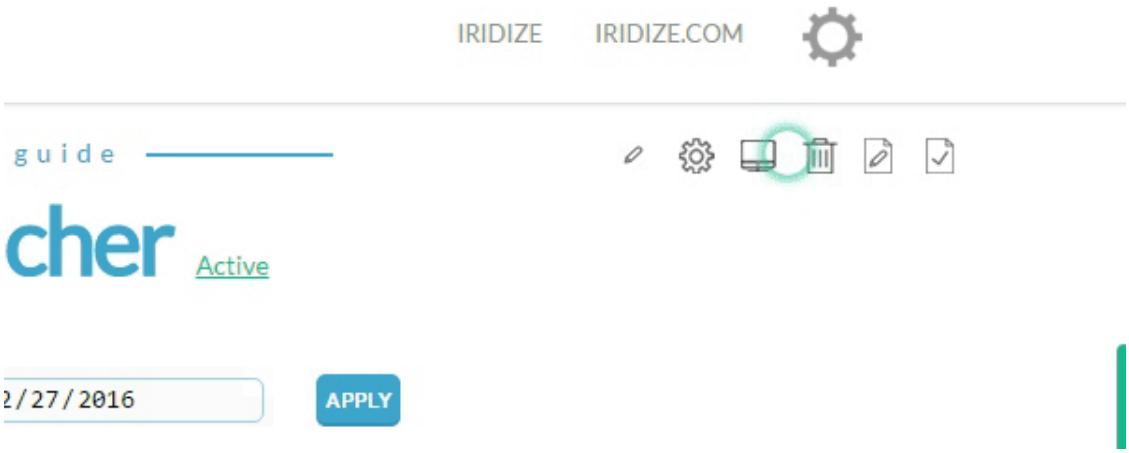
1. Use the responsive video and not use the align left/right option (set alignment to none. the width of the tip will be adjusted to fit the video and the screen automatically)



2. Disable the responsiveness by setting a width to the tip in pixels (under the display settings) and not in %

What is a beacon?

A beacon is a great way to attract your users and make them engage with a new feature or component that was added to your application.



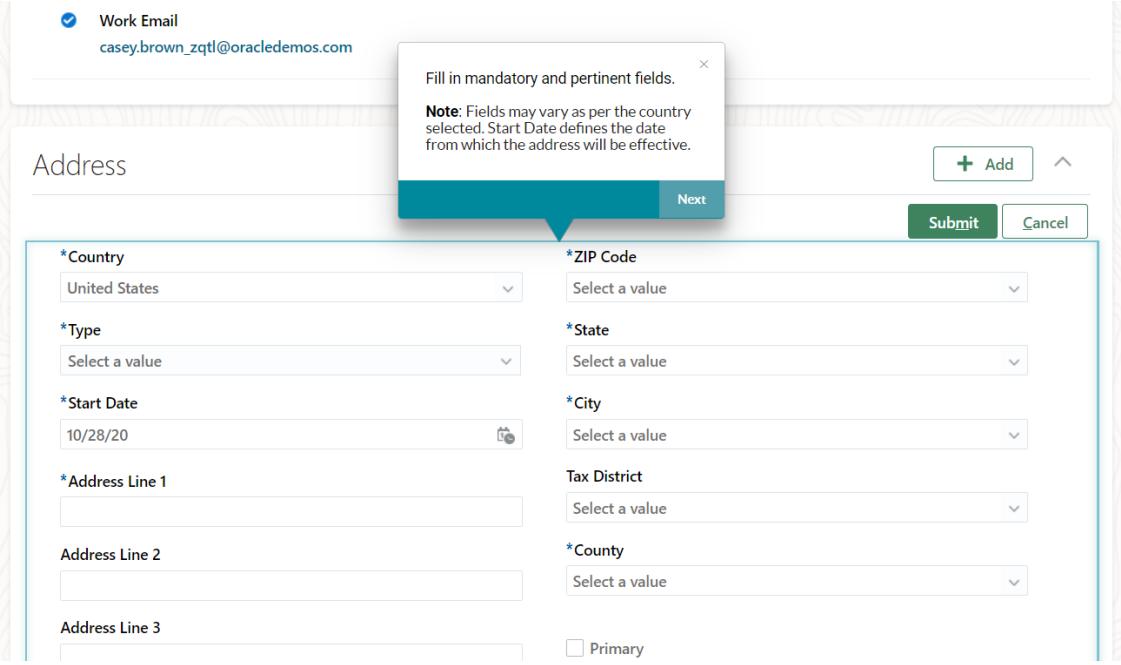
The beacon is the pulsating circle. You can place it on any element in your application by selecting it in the flow editor. The beacon setting is under the ***display settings***.

A great way to use a beacon is as a launcher to another guide. This will allow you to see separated analytics for users who engaged with the beacon (by hovering over it).

What is the best way to guide a user while filling out a form?

Generally speaking, users today know what a web form is and how it needs to be filled. You probably won't provide your end user with much added value by pointing at a password field and telling him/her to "enter your password in the text field".The best practice is to provide your

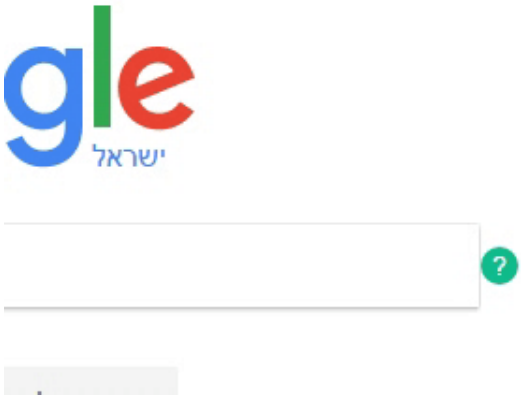
users with an overview of the purpose of the form. Provide additional information about fields that may be unfamiliar to your users. It is recommended not to disable the highlight as your users may want to go back and make changes to other form fields.



What is the help icon?

Smart Tips provide context-sensitive help on a field, button, or other UI element. There are three types of Smart Tips:

- **Icon**: Used to provide context-sensitive help, hover text, and supplemental information to buttons, labels, and other screen elements
- **Invisible**: Anchored to an element on the page, Invisible Smart Tips are used to provide context-sensitive help without using a help icon or beacon
- **Beacon**: Like an Icon Smart Tip, a Beacon adds a pulsing image drawing the reader's attention. Best practice is to use this type of Smart Tip for emergency and temporary notifications



Guided Learning places the help icon to the right of the element.

The help icon setting is available under the ***display settings*** in the Guided Learning editor.

You can style the help icon by downloading the design kit.

When I use two or more activation options in a guide, do they work like OR or like AND conditions?

They work like an AND.

That is, all conditions must be met for the guide to be launched.

Note that if you want to create an OR condition so that the guide could launch on more than one URL, you can do that using a regular expression.

5

Organizing Content

This chapter contains OGL knowledge articles on organizing your content.

Can I change the order of the guides in the start panel?

Yes.

In Display Groups you can simply drag-and-drop content to place it in the desired sequence.

This feature allows you to reorder your OGL content.

Can I hide the help widget?

Yes.

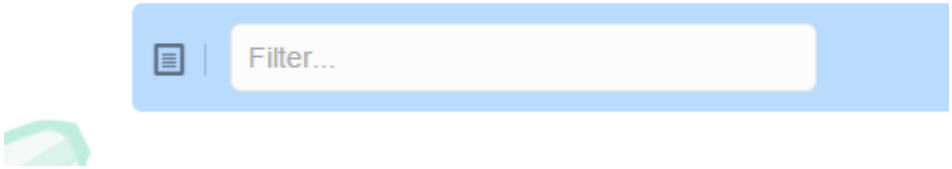
There are 2 ways:

- In Help Widget Settings you can uncheck "Display help widget even if there are no guides in the list". If the help widget has no content to display it will hide itself automatically.
- You can change your embed code to force the help widget to not load

```
iridize.showStartPanel = false;
```

Filtering a Guide in the Guides Page

You can use the filter box in the guides page to create very fancy filters.



If you just type in the box Guided Learning will automatically search that text in the **name** and **description** and **URL** of all your guides.

But you can also use **regular expressions**.

For example if you want to find all guides that have wordA or wordB in their name or description all you need to do is

/wordA|wordB/

Here's a full regular expression reference

On top of that, there are Guided Learning specific search terms:

- **:active** - returns all guides that are in production

- **:testing** - returns all guides that are in testing
- **:offline** - returns all guides that are offline (inactive)
- **:draft** - returns all guides that have an unpublished draft
- **:wal** - return all walkthrough guides
- **:lnk** - returns all link guides
- **:vid** - return all video guides
- **:widget** - returns all guides that appear in the help widget
- **:autoload** - returns all guides that have autoload conditions
- **:nocond** - returns all guides that have no activation conditions
- **create:[username]** - returns all guides that were created by user
- **edit:[username]**- returns all guides that were last updated by user

Note: You can use these in conjunction while using their order. For example, if you want to search all video guides that are autoloaded.

Getting a Link to Display Correctly

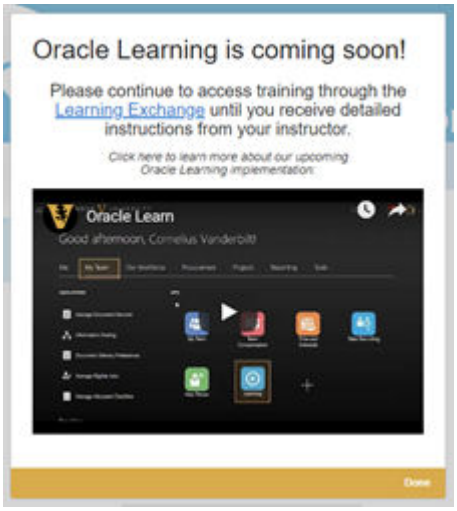
Link size does not change with rest of text

If you resize the body text of a tool tip, and that text includes a link, the link will not resize with the surrounding text. (see screenshot)

In order to get around that, you need to add custom code to the <a> element. Here is code example that results in the bottom screenshot:

```
<p style="text-align: center;"><span style="font-size:20px">Please continue to access training through the <a href="https://learningexchange.ea.vanderbilt.edu/" style="color:#2f78cb; font-size:20px" target="_blank">Learning Exchange</a> until you receive detailed instructions from your instructor.</span></p>
```

Example of link not resizing with surrounding text.



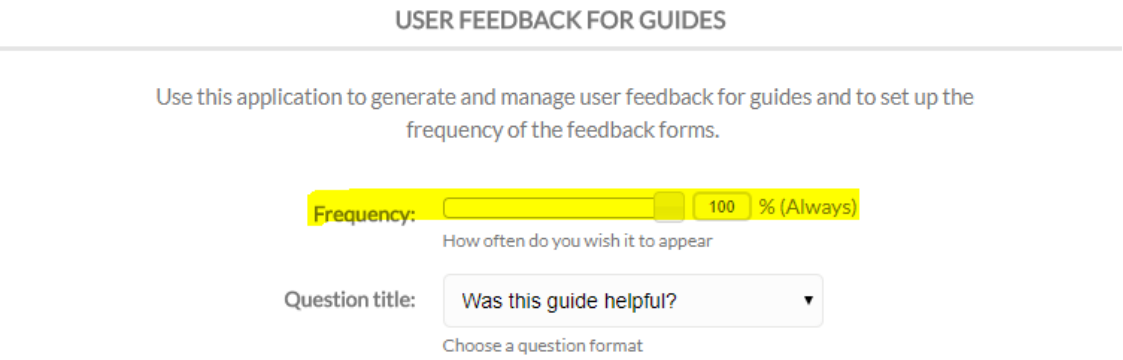
Example of link resizing with custom code applied.

Guide Feedback Widget does not Display

In order to avoid badly affecting the users' experience, Guided Learning runs several checks before displaying the feedback widget.

- Guided Learning will never display the widget twice for the same guide
- The widget will only display if more than 10 minutes passed since the last time the user saw the widget. That means that if the user provided feedback about a certain guide they will only see the widget again after more than 10 minutes.
- If the guide has just one tip, for example a 'Release Notes' notification, the widget will not display.

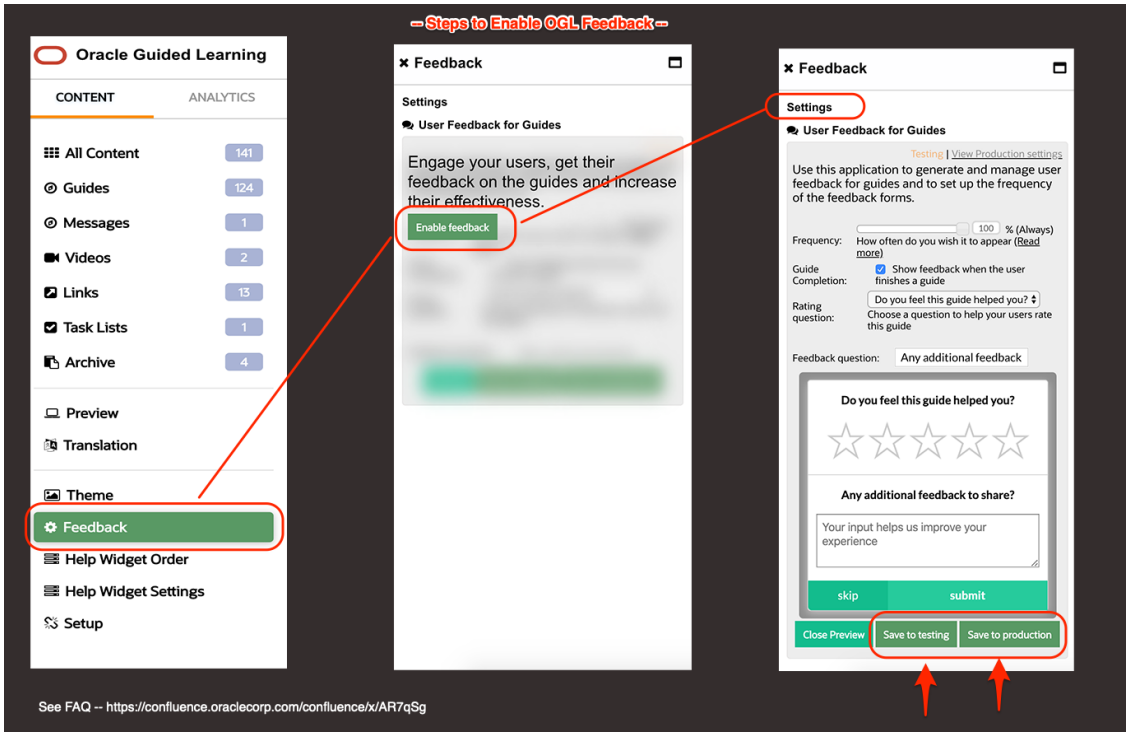
After all these conditions are met, Guided Learning will evaluate the frequency setting.



For example, if the frequency is 100% the feedback will display.

If the frequency is 50%, there is a 50% chance that the feedback will display; and so on.

How to enable OGL Feedback in your environment:

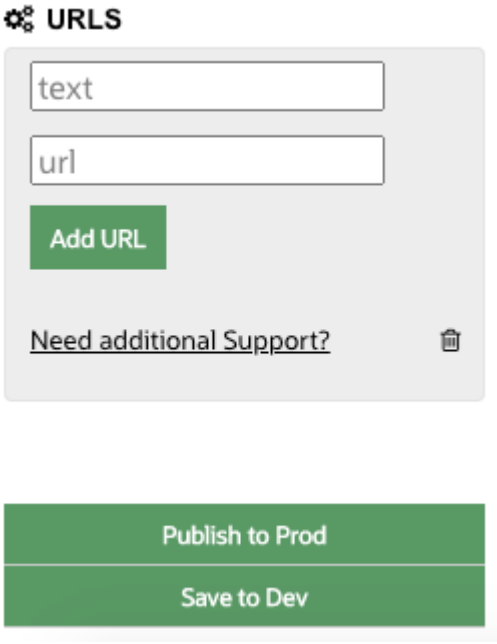


Help Widget Footer Customization

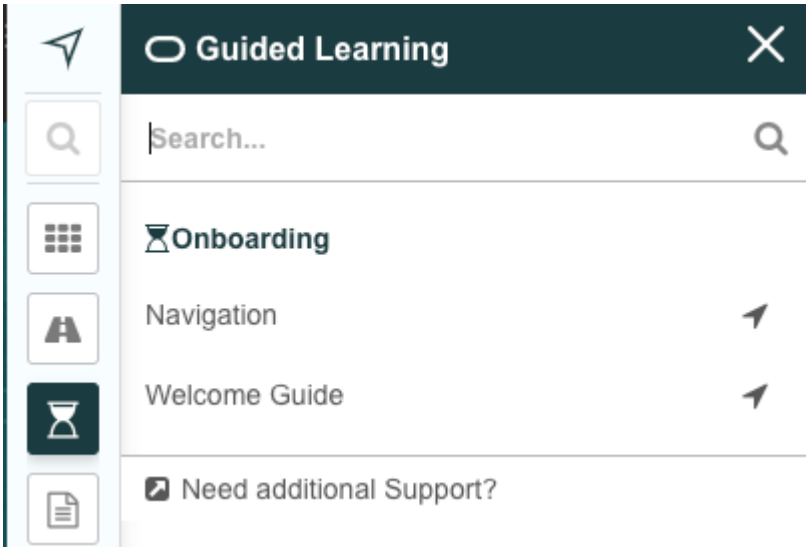
Instructions

Add the steps involved:

1. Go to Help Widget Settings on the left had side of your OGL Console.
2. In the URLs section, enter the text you want to appear for the link in the "text" field. In the "url" field, enter the url of the site you wish to link to.
   - Follow this url pattern: http://google.com
3. Select the "add URL" link
   - After you click the "add URL" link, an entry for your link should appear with a garbage can next to it: You should end up with something that looks like this:

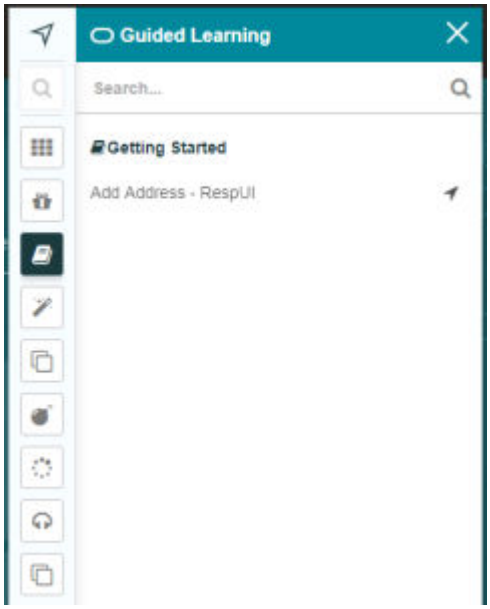


4. Select the Save Settings button.
   - The screen will refresh and you will be at the top of the Setup Help Widget page.
5. Open the application with Guided Learning and open the Help Widget.
   - You should now see a link at the bottom of the help widget like the one shown in the image below:



What is the start panel?

The start panel shows a contextual list of guides. The list can be based on the specific **user**, the specific **page** the user is on, or both.


Guided Learning
Search...
Getting Started
Add Address - RespUI

6

Guide Conditioning

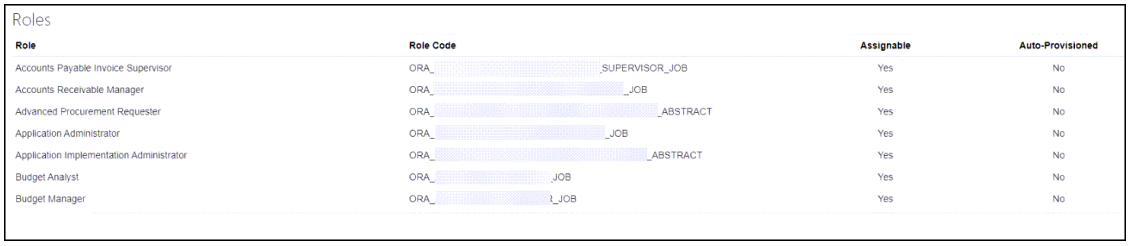
This chapter contains knowledge articles related to Guide Conditioning.

Advanced Guide Conditioning Using Shell Roles

Fusion Roles

In Fusion Applications, users are segmented into groups based on their roles. These roles help customize the Fusion features for different groups, such as employees, contractors, managers, etc. For instance, employees are restricted from accessing certain features, while managers have exclusive access to additional features that correspond with their roles. This is achieved by segmenting user access using Standard and Custom Roles.

Example of Standard and Custom Roles in Fusion:



Roles

Role	Role Code	Assignable	Auto-Provisioned
Accounts Payable Invoice Supervisor	ORA_ _SUPERVISOR_JOB	Yes	No
Accounts Receivable Manager	ORA_ _JOB	Yes	No
Advanced Procurement Requester	ORA_ _ABSTRACT	Yes	No
Application Administrator	ORA_ _JOB	Yes	No
Application Implementation Administrator	ORA_ _ABSTRACT	Yes	No
Budget Analyst	ORA_ _JOB	Yes	No
Budget Manager	ORA_ _JOB	Yes	No

OGL Guide Conditioning Using Standard and Custom Roles

OGL utilizes the same user segmentation roles as Fusion to ensure proper guide conditioning. This means that specific OGL guides will only be accessible to certain user roles in Fusion. For example, in the case of managers, OGL uses the role code for managers from Fusion to ensure that only designated guides are accessible to users identified as managers.

However, Fusion applications have a limitation in segmenting users into sub-cohorts using Standard Roles and Custom Roles since the Fusion Security Console team has chosen not to include many of these attributes in LDAP (Lightweight Directory Access Protocol). And therefore, Fusion doesn't have the capability to segment users according to their specific job designation, the business unit within the organization, or the departments within the same business unit. For example, Fusion segments users into employees and managers. But Fusion doesn't segment the employees into sub-cohorts, like Junior Analysts or Developers.

Therefore, OGL cannot use the current Fusion role codes for tailoring guides to specific cohorts, such as Developers or a particular business unit within the organization.

What is a Fusion Shell Role and How it is Useful for OGL Guide Conditioning

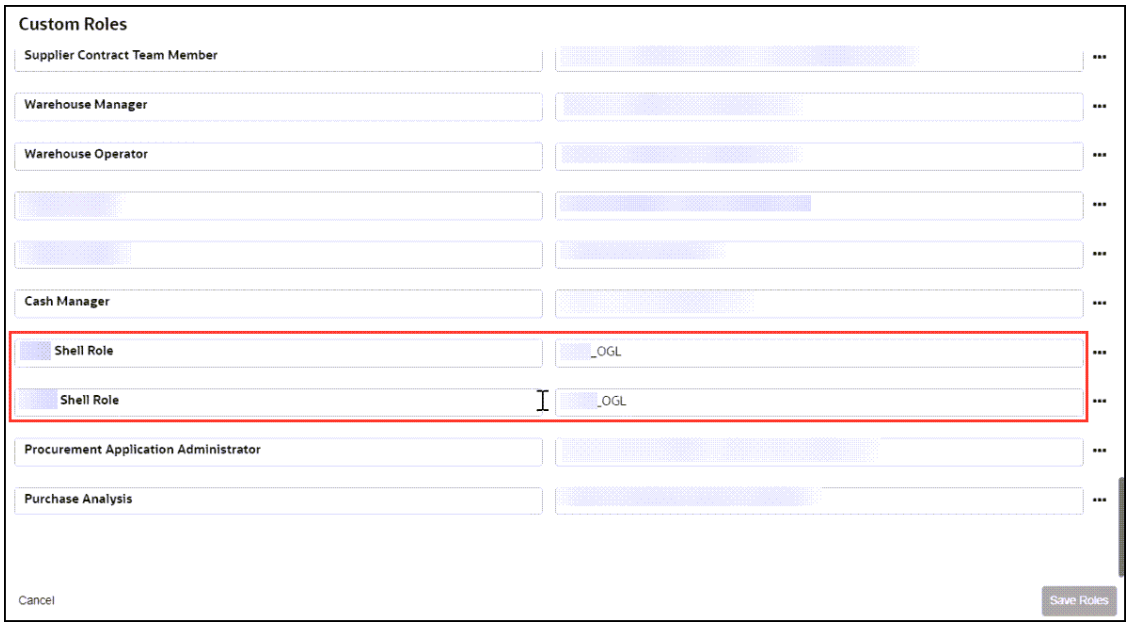
A **Shell Role**, as its name implies, is a dummy role that Fusion users can have in addition to their Standard and Custom Fusion roles. They are empty roles. Shell Roles are created by the client administrators, and they do not provide access to any of the Fusion features. These dummy user roles are assigned and restricted to, and only to, a specific group of employees that represents a cohort or a sub-cohort, like a specific employee group (Analysts, engineers, etc.), a business unit of an organization, or even a department within a business unit.

Once assigned to cohorts, the role code of the same Fusion Shell Role is added to the OGL console to customize the OGL guides to specific cohorts. So, this sub-level user segmentation in Fusion enables OGL to develop and deploy unique content for specific employee groups, a business unit of an organization, or even a department within a business unit.

Important:

- Customers/users are responsible for creating Shell Roles in Fusion Application, as per the requirements.
- As a Self-Service customer, you can create Shell Roles in Fusion and include them in the OGL Custom Roles list on your own.
- As a Managed Service customer, you can create Shell Roles on Fusion and provide them to your Project Manager. Oracle will then include these Shell Roles in the OGL Custom Roles list.

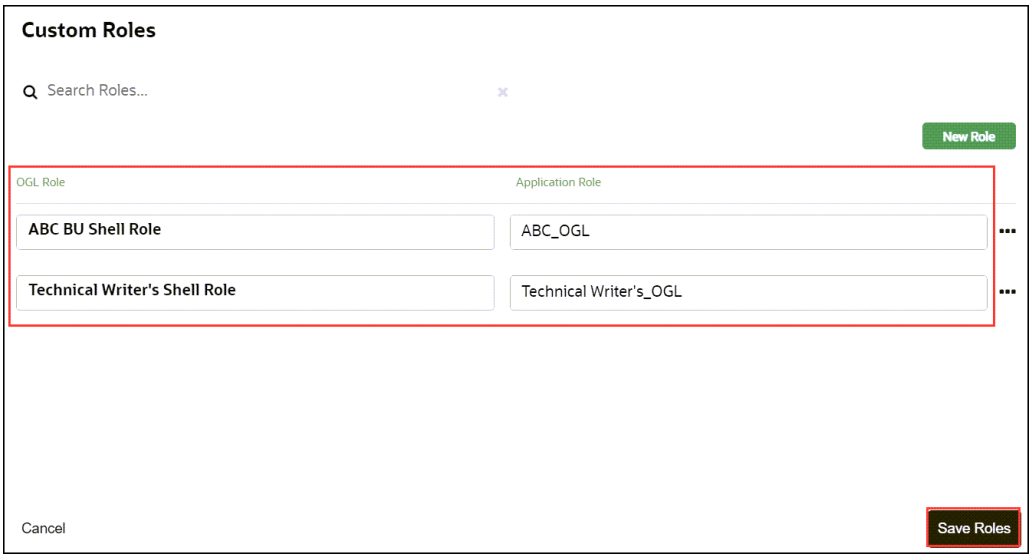
Example of Fusion Shell Roles used in OGL Custom Roles:



Process in a Nutshell

Here is an overview of the process:

1. The Fusion customer identifies the OGL guide conditioning requirements for each cohort.
2. As per the guide conditioning requirements, the Fusion administrator creates the Shell Role(s) in the Fusion Application.
3. The Fusion Administrator then adds the Shell Roles to the respective cohorts.
4. **For Self Service customers:**
   Add the Shell Role(s) to the OGL Custom Roles list.
   a. Go to the OGL console.
   b. Go to **Settings > Custom Roles**.
   c. Select **New Role**.
   d. Enter the **OGL Role Name** and the **Application Role Value**.
   e. Select **Save Roles**.



5. **For Managed Service customers:**
   Supply the Shell Role(s) details to your OGL Project Manager. Afterward, Oracle adds the Shell Roles to the list of OGL Custom Roles.

7

Statuses

This chapter contains OGL Statuses knowledge articles.

What do 'Active', 'Testing', and 'Offline' mean?

An Oracle Guided Learning guide can have multiple versions.

Each time you edit a guide, a new revision of that guide is stored in our database. A revision can either be a **published** or an **unpublished revision**.

To complement this, Guided Learning supports a dual deployment mode. While you can only see published revisions in your production environment, Guided Learning allows you to see draft revisions in your staging environment.

This can be very useful if you want to create a guide for a new feature that is not yet live in production.

- **'Published'** means that it is live in production. This means that the guide has a published revision. There should be a checkmark icon under the actions column, click to unpublish it.
- **'Draft'** means that it is only available in staging. This means that your guide has just a draft version. There should be a pencil icon under the actions column, click to publish it.
- **'Unpublished Revision'** is a copy of a Published guide that you are making some edits to. While in this status the production guide is still visible to the users
- **'Inactive'** means that the guide cannot be seen on either staging nor production environment. The guide can be re-activated if necessary

When a guide has multiple revisions, it can have an unpublished revision that will be displaying in your staging environment and a published revision that is live in your production environment. If this is the case you will have both a pencil and a checkmark icon under the actions column. This will allow you to either publish the draft revision or unpublish the published revision.

8
Theme

This chapter contains OGL Theme knowledge articles.

Can I remove the "Powered by" footer?

Yes.

Paying customers can completely white-label the Guided Learning layer.

In order to do that you need to download the Design Kit.

In the design kit you can use the following CSS rule to hide the 'Powered by' footer.

```
div.sttip .guide-footer {
    display: none;
}
```

How to Change the Help Icon

The help icon is part of the Guided Learning theme.

Paying customers can download and configure their theme by downloading the design kit.

To change the background color of the icon:

```
div.ir-marker {
    background-color: #12bc8d;
}
```

To change the ? to another letter:

Add the following CSS rule to the design kit (if it doesn't already exist).

```
div.ir-marker:after {
    content: 'i';
}
```

To change it to your own custom image:

```
div.ir-marker:after {
 content: none;
}

div.ir-marker {
background:
 url(data:[your image goes here])
 no-repeat
 left center;
}
```

You can use the following service to get the Data URI of your image - https://dopiaza.org/tools/datauri/index.php

Is it possible to make the overlay a little lighter?

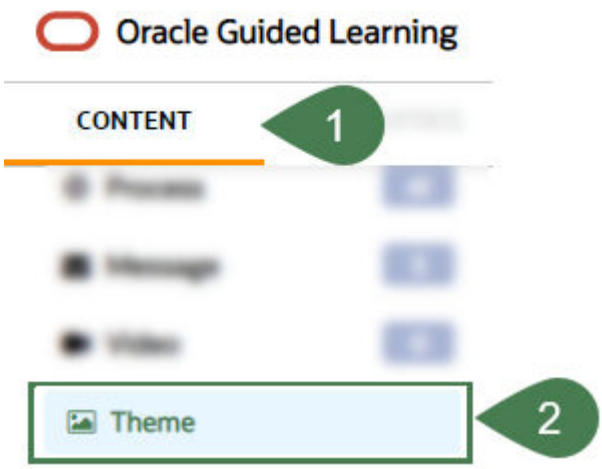
Yes, download the design kit and edit as desired.

NA Tenancy - https://guidedlearning.oracle.com/account/theme/

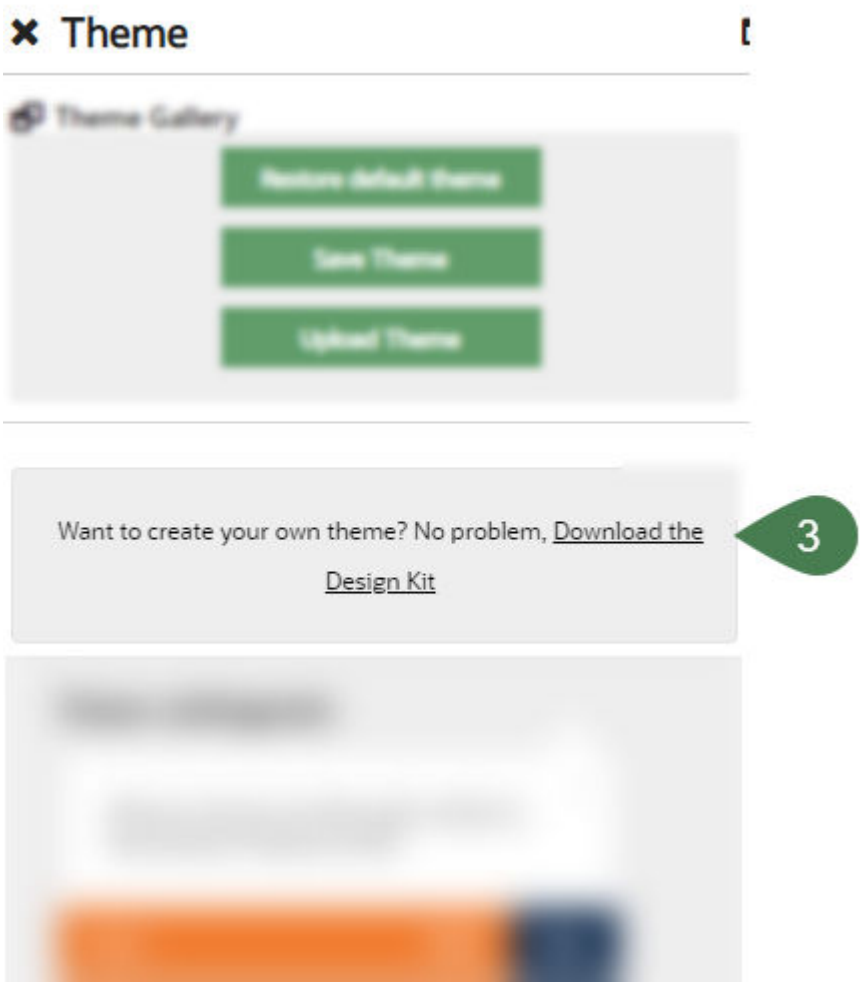
EMEA Tenancy - https://guidedlearning-emea.oracle.com/account/theme/

To Access the Design Kit from the OGL Console:

1. Select the **Content** 1 tab from the OGL Console
2. Select **Theme** 2 from the left panel



3. Select the link to **Download the Design Kit** 3



We offer multiple themes to our users. Is there any way to change the OGL theme based on user or, even better, to update individual styles on the fly?

Oracle Guided Learning themes are 100% CSS, so if you dynamically add CSS rules on the fly in such a way that they will 'win' over the default rules in your theme, you can easily change the look and feel at run time.

Enabling Multiple Styles in the same Guide

When there is a requirement to have different formatting for tips or help icons within the same guide (or even the same step), you will need to update the theme. An example of this would be having Smart Tip icons for policy documentation be one color, the associated tip a different style, and Smart Tip icons for how to use the tool in the default style.

To enable multiple styles:

1. Download the Design Kit and open the html doc in your preferred text editor
    - Best Practice: Search for "**ir-marker**" then organize your code in an orderly fashion

- It will help to have your custom code entered in proximity to the default code

2. The following code will result in the help icon containing an "i" when the POLICY class is active:

```
div.<your_class_name >.ir-marker:after {
content:'i';
}
```

so, for a class called "POLICY"...

```
/*This is the text for the POLICY Smart Tip*/
div.POLICY.ir-marker:after {
content:'i';
}
```

3. The following code will result in the help icon having a green background when the POLICY class is active:

```
div.<your_class_name >.ir-marker{
background-color: deeppink;
}
```

so, for a class called "POLICY"...

```
div.POLICY.ir-marker {
background-color: green;
}
```

4. The following code will result in the Next and Done buttons being green when the POLICY class is active:

```
div.sttip div.tooltip.<your_class_name> div.stFooter [data-iridize-role="nextBt"] {
background-color: darkorchid;
}
```

so, for a class called "POLICY"...

```
div.sttip div.tooltip.POLICY div.stFooter [data-iridize-role="nextBt"] {
background-color: green;
}
```

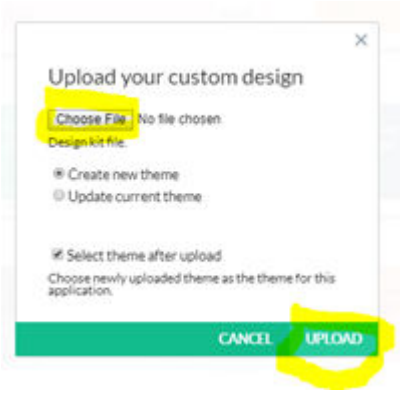
5. The following code will result in the back button being grey when the POLICY class is active:

```
div.sttip div.tooltip.<your_class_name> div.stFooter [data-iridize-role="prevBt"] {
background-color:grey
}
```

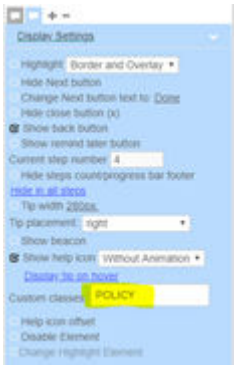
so, for a class called "POLICY"...

```
div.sttip div.tooltip.POLICY div.stFooter [data-iridize-role="prevBt"] {
background-color:grey
}
```

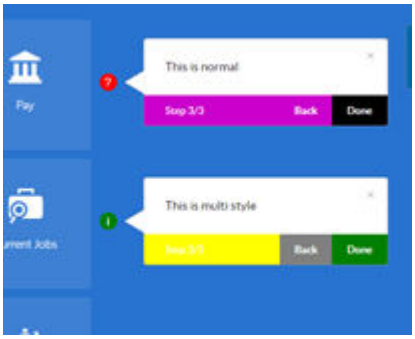
6. **Save** the edited HTML file. The above code will result in changes to the tip footer appearance outside the Next and Back buttons when the POLICY class is active.
7. Open **Settings** for the appId and select **Themes**
8. Scroll to bottom of Setup Theme configuration page and select **Upload Theme**.
9. Select **Choose File** button to select your edited HTML file then configure dialogue window as shown below then select **Upload**



10. In a new guide or existing guide, create a simple tip pointing to an element on the screen and enable the help icon *without* "show tip on hover-over" enabled
11. In the same step, create another tip just as above (point to a different element on the screen), but enter "POLICY" in the class field as shown below



12. Save the guide and launch the application
13. The step created above should look something like this



9

Tips & Tricks

This chapter contains OGL tips and tricks.

Multiple Widget Display Prevention

On a page containing an iFrame, the iFrame must have its own JS embedded. As such, it will usually display a launch widget, meaning the screen will display two launch widgets. To avoid this, place the following line of code in the embedded JS where you DON'T want the launch widget to appear:

```
iridize.stateOnly=window != top ? true : undefined;
```

Reference below snippet of the JS embedded into the iFrame application:

```
<script type="text/javascript">
window.iridize=window.iridize||function(e,t,n){return iridize.api.call(e,t,n);};iridize.api=iridize.api||{q:[],call:function(e,t,n){iridize.api.q.push({method:e,data:t,callback:n});}};
iridize.appId="nLyBo3uAR4ChwdcS8LHvRw";
/*FOR YOUR DEV OR STAGING ENVIRONMENTS PLEASE UNCOMMENT THE NEXT LINE*/
iridize.env="dev";
/* In a single page application, by default its wait for 5000 milli seconds (5 seconds) for route to be set. But due to some performance issue fusion may need more time */
//iridize.setRouteTimeoutMillis = 10000;
/*TO ENABLE USER CONDITIONS UNCOMMENT THE NEXT LINE. Add as many fields as you would like (e.g. user_role, user_name, etc.) */
iridize.crossAppIframe=true; /*To handle cross app function*/
iridize.stateOnly=window != top ? true : undefined; /*To handle cross domain function*/
/*iridize("api.fields.set", {user_id:"LOGGED_IN_USER_ID_GOES_HERE"}); */
window.addEventListener('load', function(){var e=document.createElement("script");var t=document.body;e.src={"https:"==document.location.protocol?"https:":"http:"}+"//guidedlearning.oracle.com/player/latest/static/js/iridizeLoader.min.js
</script>
```

Advance on Click Fails

Sometimes when you set advance on click on an element, Oracle Guided Learning will not advance to the following step as expected.

When this happens, normally, the application handles the click and blocks any other code from handling the event.

To work around this issue it is recommended to try using ***Advance when*** 'user presses mouse button'. The click event occurs when after the user presses and then releases the mouse button. The press event occurs before the click event, as soon as the user presses the mouse button.

As a result, using the press event you can catch the user's interaction before the click event is fired and handled by the application.

Autoload a Closed Guide

Guided Learning marks a guide as closed if the user manually closed it; normally by clicking on the 'x' at the top right corner of the tip.

To create a pleasant user experience Guided Learning will not autoload a guide that was manually closed by the user. To that end Guided Learning remembers the time that the user closed the guide.

By default, Guided Learning will not allow this guide to autoload before 1 hour has passed.

You can configure this timeout by using the following setting in your embed code:

```
iridize.stateTimeout = 3600; // in seconds.
```

Can a guide be launched automatically from an email campaign or a knowledge-base article?

Yes, regardless of the use case, you can always create a link that, when opened, automatically launches a guide.

Common use cases

- An e-mail campaign to your customers announcing a new feature. If clicked, you want the reader to be engaged with a Guided Learning tour of the new feature.
- Linking a knowledge-base article to the relevant guide. If the article is providing an overview of a certain procedure, the user can jump directly into doing that procedure with the guide activated.
- Integrating Guided Learning guides with your LMS

Prerequisites

- Guided Learning must be embedded in your application
- Sufficient knowledge of HTML or at least of creating an HTML link.
- Sufficient permissions to add the HTML link to the email campaign, knowledge-base, or LMS

Creating a Permalink

A permalink is composed of 2 parts:

- The link to the desired destination page. For example, if you application is hosted on *http://myapp.com*, the link to the user profile page could be *http://myapp.com/profile*.
- A parameter that signals Guided Learning to launch the guide. This is why we need Guided Learning to be embedded.

  ```
  stStart=[api name]
  ```

For example, to launch a Guided Learning guide with the API name *abcdefg* that gives an overview of the profile page:

1. Take the base page link **http://myapp.com/profile**
2. Suffix "**?**" to indicate a URL parameter to follow
3. Append**stStart=abcdefg**
4. End result is **http://myapp.com/profile?stStart=abcdefg**

IMPORTANT: Permalinks do not work with Oracle Fusion Applications.

Can a guide work across multiple domains?

Well, actually no, but your users will think it does.

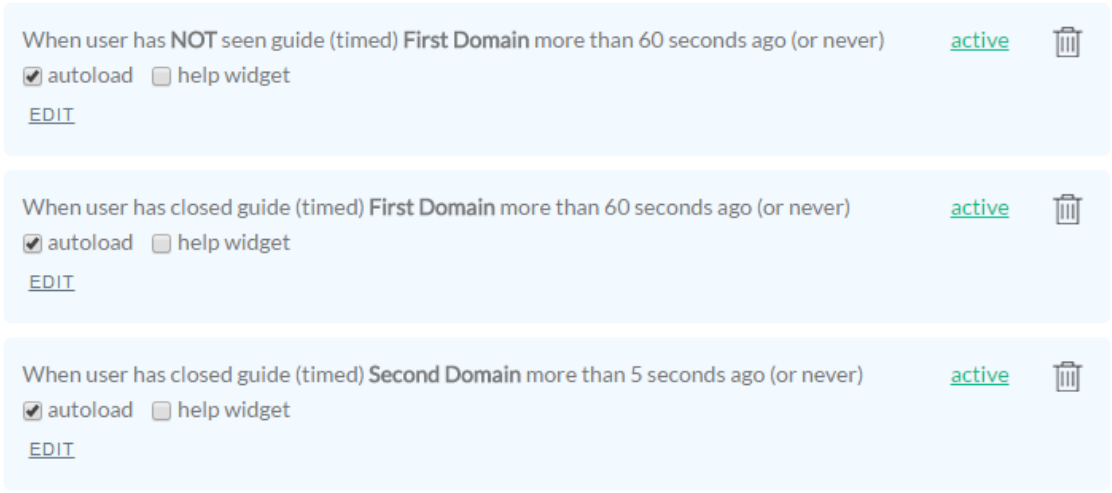
Let's explain, Guided Learning guides are running inside the browser. A key **security** policy that is always enforced by the browser is that no domain can access data from another domain; and hence the no. But what Guided Learning allows you to do is to create 2 guides, one for each domain, and link them with Guide Activation Conditions. Using this technique you can create, what would look to the end use, as a cross domain guide.

Prerequisites

1. Guided Learning is embedded in both domains via javascript or a browser extension
2. The same Guided Learning appId is used in both domains

Let's Do It

1. Split the guide into 2 guides one for each domain. If you already have a guide with all steps, no problem, clone it and delete the relevant steps from each one.
2. Add the following conditions to the **second half** of the guide:



To make the explanation easier, in the example above, the first half of the guide is called *First Domain* and the second is called *Second Domain*

Notice there are two guides referenced in these conditions. The first 2 conditions reference the guide on the first domain and the last condition is referencing the guide on the second domain.

Let's explain the conditions one by one.

1. Autoload this guide *second domain* if the user has seen the guide *first domain* less than 60 seconds ago (within the last minute).
2. Autoload this guide if the user has closed the guide more than a minute ago. Or in other words if the user has not just recently closed the *first domain*.
3. Autoload this guide if the user has closed the *second domain* guide more than 5 seconds ago **or never**. This condition is optional if you want the *second domain* guide to always launch no matter if it was recently closed or not.

Can I change the element selection algorithm?

Yes.

If needed you can hard-code the selection algorithm. By doing so you can force Guided Learning to identify the element based on your criteria.

To do so you will need to add the ***data-walkthrough*** attribute to the relevant HTML element and give it a meaningful value.

For example, if this is your HTML element:

```
<span data-walkthrough="foo">bar</span>
```

When you select it in the editor the selector will be:

```
[data-walkthrough="foo"]
```

Can I start a guide on a particular step using the public API?

Yes.

You will need to know the **apiName** of the guide and the id of the step.

To launch guide *i76g0pfx* from step id 2, you would call:

```
iridize('api.guide.start', {apiName:'i76g0pfx', step: '2'})
```

Can I use Guided Learning even if a user is not logged in?

Yes.

While you can embed Guided Learning on any web page, this issue is normally raised by our customers when they want to take advantage of the user activation conditions even before a user login. For example, to show an overview video to a user only once.

When you embed Guided Learning you can read/write a random user_id from a cookie or local storage.

To do so, please replace the following line in the embed code:

```
iridize("api.fields.set",
{user_id:"LOGGED_IN_USER_ID_GOES_HERE",joined_at:UNIXEPOCH_TIMESTAMP});
```

with ths:

```
    var randomUserId;
    var userId = ""; /* Replace the empty string "" with logged-in customer id as a
string if customer is logged-in*/
    if (userId === "") { /* If no userId is supplied, we manage a randomlly generated
user id, which is persisted in localStorage if it is available*/
        randomUserId = "Anon" + ((Math.random() + 1) * Math.E /
Math.PI).toString(36).slice(2)
        try {
            userId = localStorage.irruid || randomUserId;
            localStorage.irruid = userId;
        } catch (e) {
            userId = document.cookie.replace(/(?:(?:^|.*;\s*)irruid\s*\=\s*([^;]*).*$)|
^.*$/, "$1") || randomUserId;
            document.cookie = "irruid=" + userId+"; expires=Fri, 31 Dec 9999 23:59:59
GMT";
        }
    }
    /* Now set the user_id field */
    iridize("api.fields.set", {
        user_id: userId
    });
```

Can 'show tip on hover' and 'disable element' be used together?

Yes.

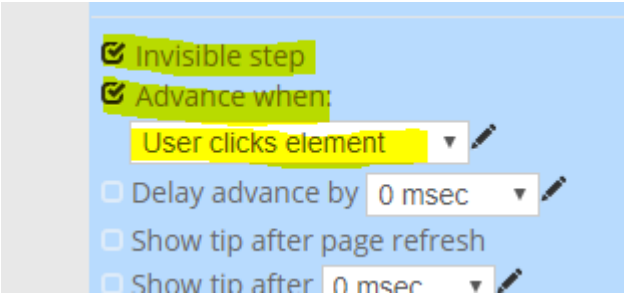
But it's not that easy.

The way Guided Learning disables an element is by placing a "transparent" layer on top of it. If for example we disable a button when the end users click on the button they are in fact clicking on that transparent layer and the button never "sees" the click and is hence unusable.

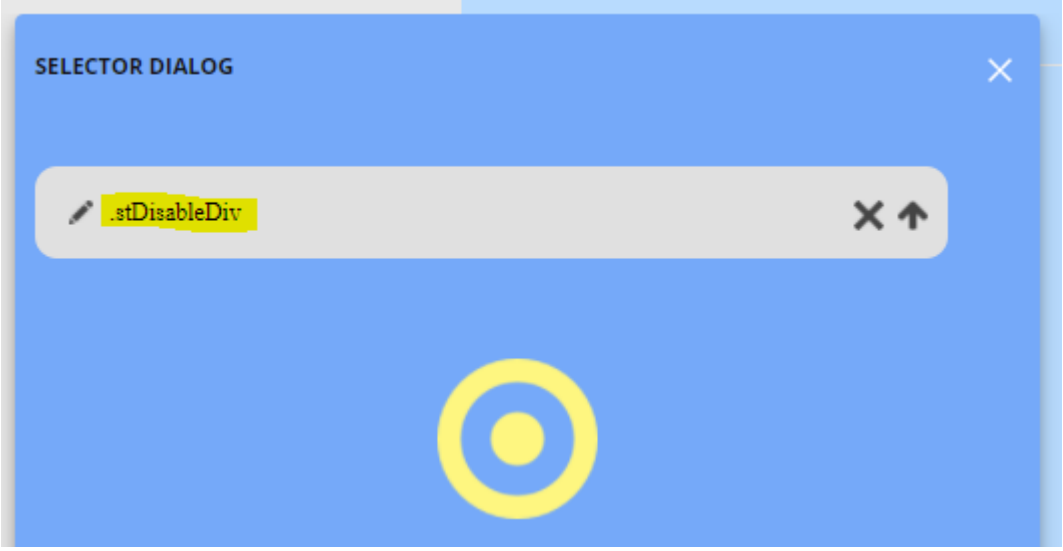
When you set a tip to show on hover, Guided Learning waits for the end-user to put their mouse over the relevant element and show the tip then; but if the element is also disabled, the mouse will never be over the element but only on that "transparent" layer mentioned above.

The solution is to have the tip show when the mouse goes over the layer instead of the element itself.

1. Create the disable layer as you normally would
2. Make it invisible and set it to advance on click



3. On the same step, create another sub-step that will 'show on hover'.
4. Set the target element for this tip to ***.stDisableDiv***



Conceptually, what is the best way to design a tour guide?

Each guide should have its own purpose. Normally a guide can either instruct a user to perform a specific task or show the user around a specific page or application. Doing both in the same guide may be confusing for your users.

Customization and Personalization Methods

With Guided Learning you can set complex activation rules for your guides, based on individual end-user fields. For example, you may wish to automatically start an introductory guide for every new user once. Or you may want to make a guide available for a user two days after she

have viewed a different guide. In order to be able to use such rules, you need to set some fields for your end-users, using the API.

api.fields.set

Set fields for the current end-user. This method expects a dictionary of one or more fields as the second argument. For example, the Guided Learning embed code includes the following call (commented out) by default:

```
iridize("api.fields.set", {
    user_id:"USER_ID_GOES_HERE",
    joined_at:UNIXEPOCH_TIMESTAMP
});
```

where we set the reserved-name field "user_id", which can be any string uniquely identifying your end-users, such as a username, user ID or an email address. Here the field joined_at is a date field, set as a numeric unix epoch timestamp (in seconds) giving the time the user signed-up to the application.

NOTE: You must set the user_id in the very first call to api.fields.set on a page. Subsequent calls may omit this field, but it is considered best practice to always include it.

Setting user group/role: you may want to user this method to customize the experience of users based on group participation or user roles. This information could then be used to set up activation rules, such that members of different groups will have different guides available. An example for such call could be:

```
iridize("api.fields.set", {
    user_id:"simpsons@springfield.us",
    joined_at:629848800,
    role:"admin"
});
```

Time fields: to set a time field, simply name the field with an "_at" suffix. The time value should be provided using a unix epoch timestamp (in seconds). For example, to set the date a user joined the application, we could call:

```
iridize("api.fields.set", {
    user_id:"simpsons@springfield.us",
    joined_at:629848800,
});
```

Personalization and Dynamic Content

api.vars.set

Dynamically set page variables which can be used in guides content.

For example, you may want to address your end-users in your guides using their names. In such a case the call to api.vars.set could be:

```
iridize("api.vars.set", {"new user":"F. Bar"});
```

The variable can then be used in any guide that will be running on the page by enclosing the variable name ("new user") within double curly brackets ({{new user}}). When the guide is run and the step is displayed, the part enclosed within curly brackets will be replaced with the value of the page variable if it is set. If it is not set, the variable name will be used instead (without the curly brackets). A default value different than the variable name can be set using the pipe, "|", symbol. For example, the text of a welcoming step could be set to:

Hi {{user_name|there friend}}, thanks for signing up to fooBar. Let us show you around.

With the variable was set as described above, the text of this step will read "Welcome F. Bar, thanks for signing up to fooBar, let us show you around.". If the variable was not set, the text will open with "Hi there friend,..."

Another use case for this API call is for including dynamically generated hyperlinks in your guides text. For example, you may have in your application paths such as "/foo/username/bar/". If you would like to add a hyperlink to such a path within a guide you could simply set the link target to "/foo/{{username}}/bar/", and run "api.vars.set" as follows:

```
iridize("api.vars.set", {"username":"007"});
```

This will yield a hyperlink with the target "/foo/007/bar/".

If the var is not set, but a user field with the same name was set in an api.fields.set call, the value from the user field will be used. If both a var and a field with the same name are set the value of the var will be used.

Display Groups not Correct in Prod

If you are seeing your Display Group settings properly in Dev but in Prod they do not look the same.

Instructions

When making changes to a published guide in the Display Group Manager that you want to see reflected in the Help Panel you must follow these steps:

1. Be sure to hit the “Save” button in the DG Manager
2. Publish the guides you updated (they will show “Unpublished Version” after a browser refresh)
3. Wait 5 minutes to see the updates in env=prod environment (prod uses caching to improve performance but updates take up to 5 minutes to populate)

Does Guided Learning provide functionality for selecting elements within an iFrame?

Yes.

But, as a javascript library Guided Learning must also adhere to the same origin policy enforced by the browser.

This means that if the iframe is loaded from a different domain than you will not be able to select elements inside it from the outside frame. An exception to that is if both the top frame and the iframe share the same root domain; If so, you can use the document.domain attribute to allow Guided Learning to selects element inside the iframe even if it is loaded from a different sub domain.

Editing Step Branches

One of the strong points of the Guided Learning solution is how simple it is to create and manage guides for the most complex processes. At the heart of this lies the Step Branches editing page the Guide Editor. The Step Branches editing page empowers you to control the flow of the guide based on various page conditions, setting multiple branches all from the a single place.

How Branches Work

You can set one or more branches for each step in the guide. For each branch you can optionally set one or more page conditions to limit when that branch could be chosen. When the guide advances from a step which has branches, the first branch that can be chosen (considering each branch's conditions) is selected and the guide continues along that branch. Thus, the order in which branches are set in the Edit Branches page is important. Additionally, adding a branch after a branch without conditions will have no effect (see Keep Current Guide Open below for an important exception). If no branch is chosen, the guide will continue to the default next step, just as if no branches were set.

NOTE: If you have multiple Tips in parallel within a Step, the branches are set for each Tip separately.

The Edit Branches Page

The Edit Branches page lists the branches for the current step, you can add and archive branches as well as edit branch settings. To open the page click the "**Step Branches: Add**" setting in the Advanced Settings Panel of the current step in the Guide Editor.

A step with two branches. The first branch is a Jump to Step branch with conditions and the second is a Launch Guide branch.

1. Done/Cancel

Click Done to save the step branches or click Cancel to discard your changes.

2. Action Buttons

The action buttons (Activate/Deactivate/Delete) allow you to perform an action on all selected branches. You can select all branches by checking “Select all branches” box (labeled 3 in the image above) or by manually selecting each branch using the selection checkbox at its top right corner.

- **Activate**: Click to activate all selected branches. Only active branches are evaluated.
- **Deactivate**: Click to deactivate all selected branches. Inactive branchesare not evaluated and will not affect the flow of the guide.
- **Archive**: Delete all selected branches. **This action cannot be reversed once Done is clicked**.

3. Check to select all Branches

Check this box to select all the branches in the Edit Branches page. Helpful if you would like to perform a mass action.

4. Branch Editing Area

See below for more details on editing branches . The image above displays 2 branches.

5. Edit Branch

Click to edit the branch.

6. Show/Hide Advanced Settings

Click to toggle whether advanced branch settings are visible. This option is visible for Launch a Guide type branches only.

7. Reorder Branches

Drag the up & down arrow to drag and drop a branch and change the order of branches.

8. Add Branch

Click to add a new branch.

Jump to Step Branch Type

Jump to another step within the same guide.

Editing a Jump To Step branch. This branch is set to Jump to Step with step API Id "3" when the set Conditions are fulfilled.

1. Select Branch Type

Branch type select box.

2. Step API id

The API id of the step to jump to. The step API id may be found in the Step Settings Panel of the Guide Editor.

3. Add/Edit Conditions

Click to open the conditions editor, where you can add/edit Conditions to the branch.

4. Done/Cancel

Click on Done to finish editing the branch and keep your changes. Click on Cancel to discard the changes made to the branch.

Launch a Guide Branch Type

Choose the Launch a Guide branch type to launch another guide from within the current guide. When used without branch conditions, this is often used to string together several guides into a unified guide. When branch conditions are used this is often used for forking a guide based on user choices or application state.

Editing a Launch a Guide branch with advanced settings. This branch is set to Launch a Guide with apiName "98j68vyj".

1. Select Branch Type

Branch type select box.

2. Guide apiName

The apiName of the guide to launch. The apiName of a guide can be found in the guide's listing on the Guides page of the Guided Learning dashboard.

3. Step API Id

By default the launched guide is started from its first step. When this textbox is not empty, the launched guide will start from the step with the API id in the textbox. The step API id may be found in the Step Settings Panel of the Guide Editor.

4. Keep Current Guide Open

The Launch a Guide type of branch normally launches the guide and then closes the current guide. When the Keep Current Guide Open checkbox is checked the current guide is kept open. Additionally, the branch choosing process continues on to the following branches (if any) and to the default next step if no other branch is chosen. This is often useful when launching a guide which is run in a popup window (see Popup Window Name setting below), where the current guide stays open waiting for the user to get back from following the launched guide on the popup window.

5. Add/Edit Conditions

Click to open the conditions editor, where you can add/edit Conditions to the branch.

6. Show/Hide Advanced Settings

Click to toggle whether advanced branch settings (items 7-11 below) are visible.

7. Wait for Page to Load

When this checkbox is checked the launched guide will not start until a page load takes place (either a full page load or a SPA route load on a properly integrated SPA). This is almost always necessary when launching a guide in a popup window (see Popup Window Name setting below). Another use for this setting is when combined with the Redirect to Page or Reload Page settings.

8. Redirect to Page

When the textbox is not empty, the user will be redirected to the URL specified in the textbox when launching the guide. Please note that it is almost always a good idea to also set the Wait for Page to Load setting when using Redirect to Page.

9. Reload Page

When this checkbox is checked, the page will be reloaded when launching the guide. Please note that it is almost always a good idea to also set the Wait for Page to Load setting when using Redirect to Page.

10. Popup Window Name

By default the launched guide starts in the current browser tab/window. When this textbox is not empty the guide will be launched in a tab/window with the Javascript window.name property matching the text in the textbox. This is useful for launching guides within a popup window or another tab. For use with a popup window, this setting almost always requires also setting the Wait for Page to Load setting to work properly.

If you don't know the name of the popup window, refer to this video

11. On Close Return-Here/Launch-a-Guide

This setting sets an action to take place when the launched guide is closed (either by the user or by finishing the guide).

- Return Here - with this setting the launcher guide is suspended upon launching the guide. When the launched guide is closed the launcher guide resumes from the same step it was when suspended.
- Launch a Guide - with this setting the launcher guide is closed as usual. When the launched guide is closed, the guide with the apiName set in the textbox is launched.

12. Done/Cancel

Click on Done to finish editing the branch and keep your changes. Click on Cancel to discard the changes made to the branch.

Embedding an iFrame in a Guided Learning Step

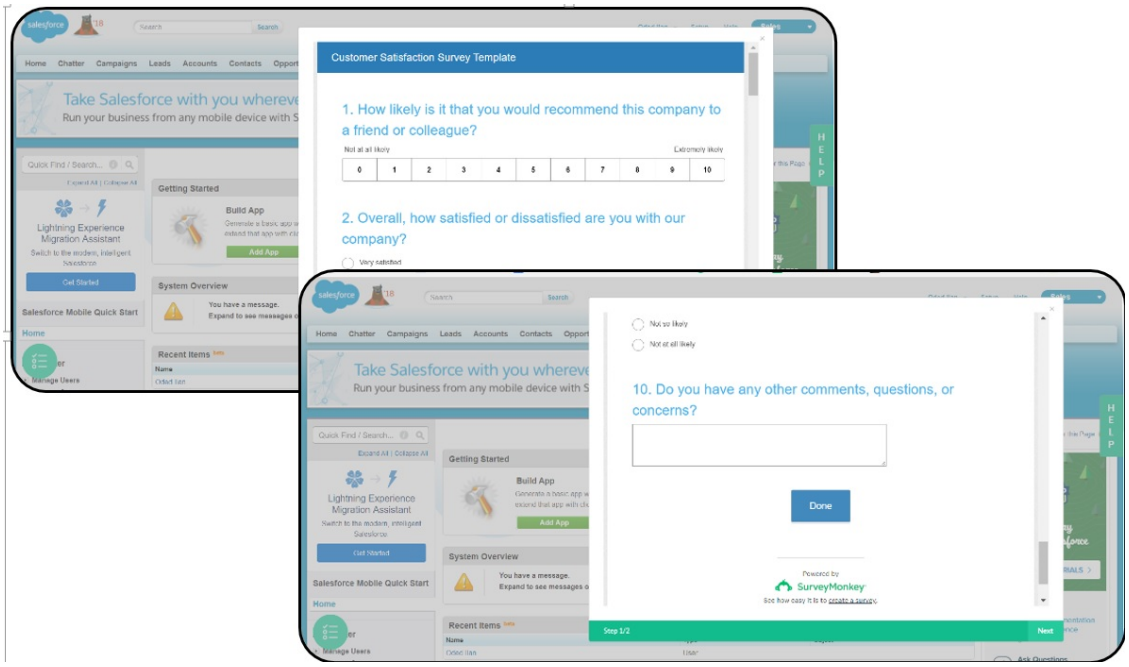
One of the cool ways of using Guided Learning is embedding external content into an Guided Learning “bubble.”

You can use this feature to, for example:

- Provide your users with a survey that you’ve created (in SurveyMonkey or in Google Sheets for example) directly over the interface of your product.
- Give you users direct access to relevant information in your FAQ or Knowledgebase while they are using the product
- Sharing information from an external source or website in a contextually relevant

This features utilizes Guided Learning’s ability to embed iframes (or *Inline Frame*) in our bubbles. These iframe-bubbles can appear as single-step guides or as part of a longer guide (one of the steps), based on your requirements and the process.

And, of course, these guides and bubbles can be made to appear according to pre-set conditions (Only once, only on certain pages, only to certain people, etc.), just as any other guide. In addition, Guided Learning collects all usage data and lets you know that your users have actually seen this content (even though it is outside of your product).



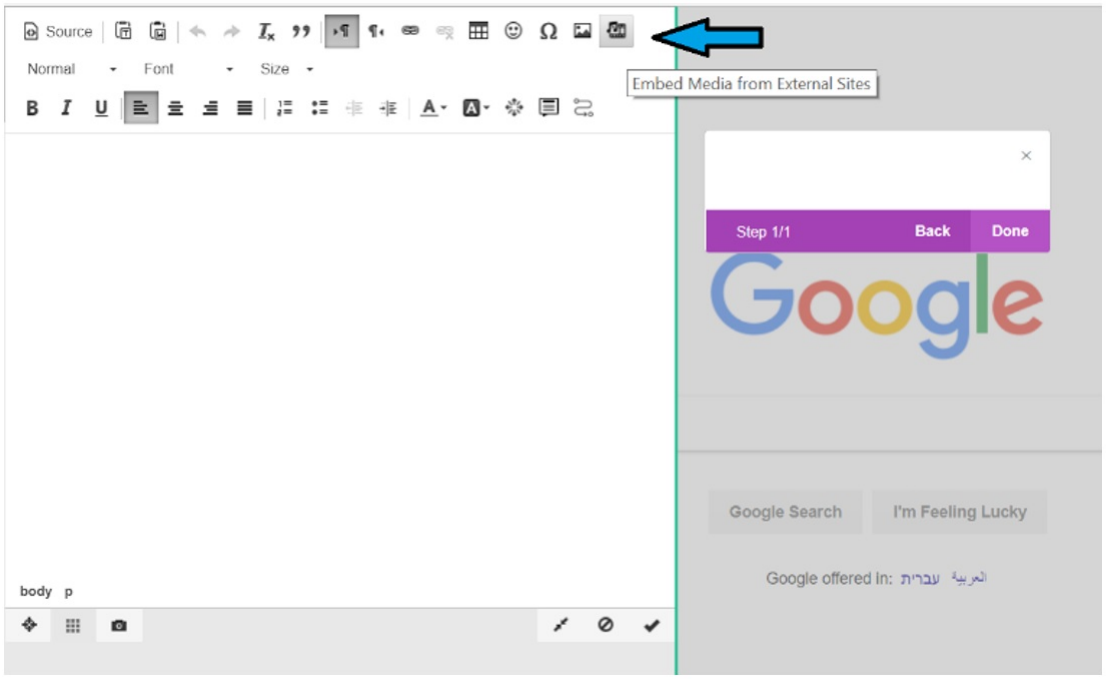
Embedding an iFrame

The mechanism of embedding an iframe into a bubble is similar to embedding a video or an audio link and uses the same process. Guided Learning has a generic algorithm that recognizes the type of content that is provided in the URL and automatically deploys it correctly in the bubble.

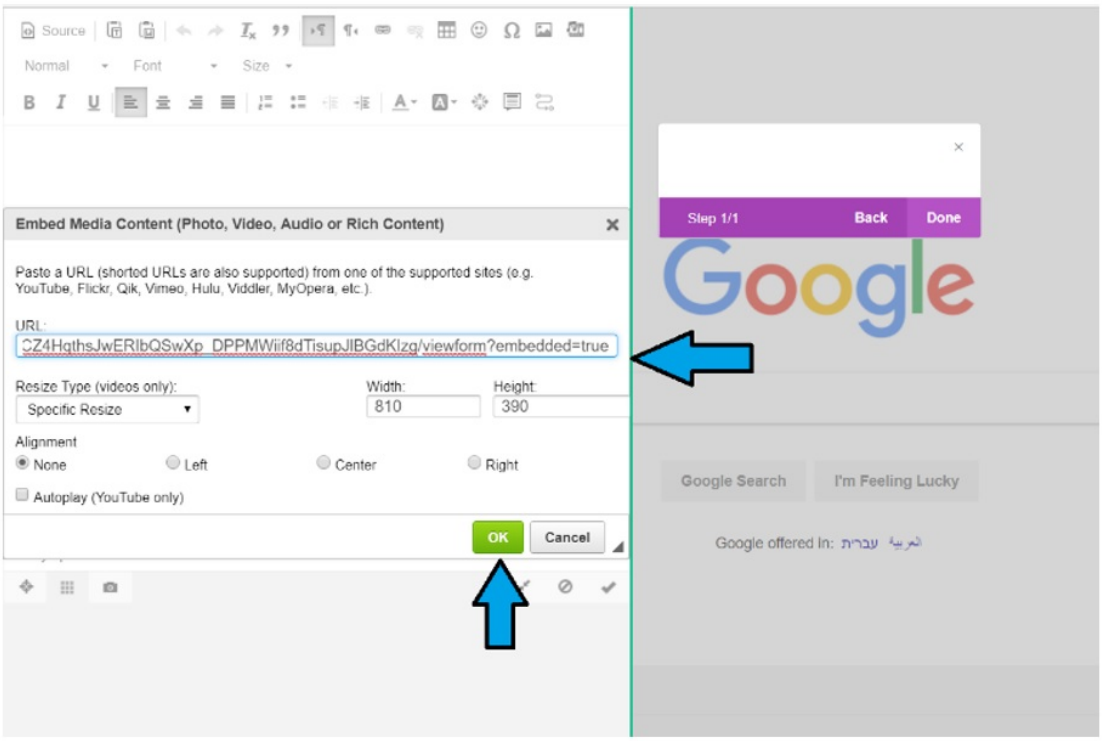
Just as with video and audio, there are two ways you can embed iframes into Guided Learning bubbles. In both cases the process is easy and quick.

From the Editor

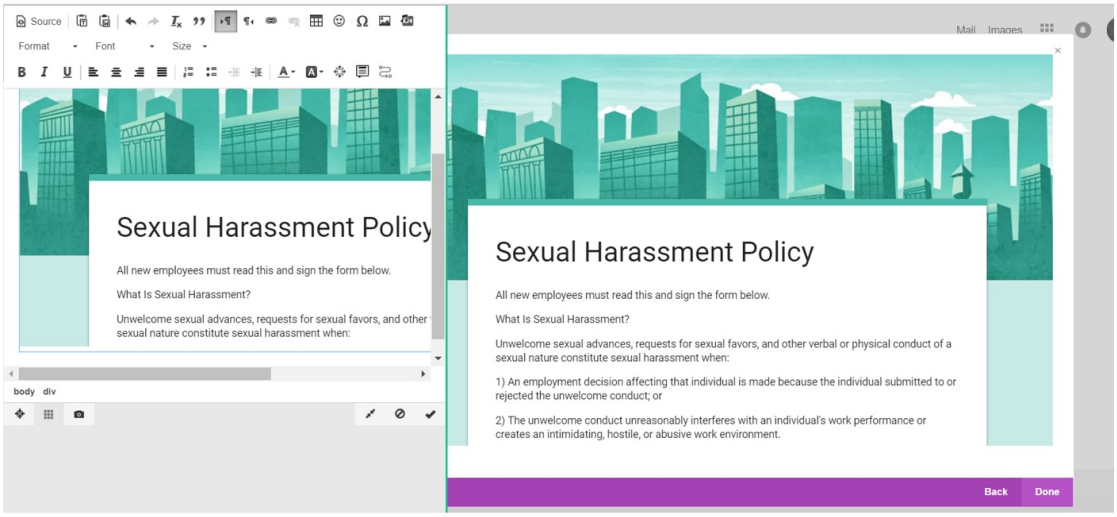
- Launch the Guided Learning Editor
- In the Expanded Content Editor choose the Embed Media Console Box (see Image)



- In the console box enter the URL for your iframe and click “ok”

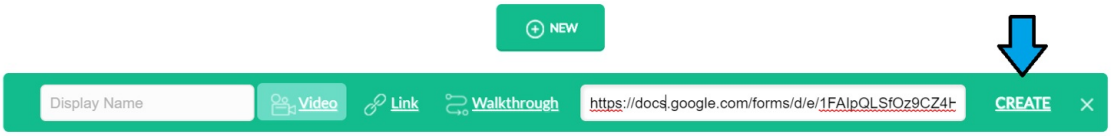


- The external website will now appear in your Guided Learning bubble

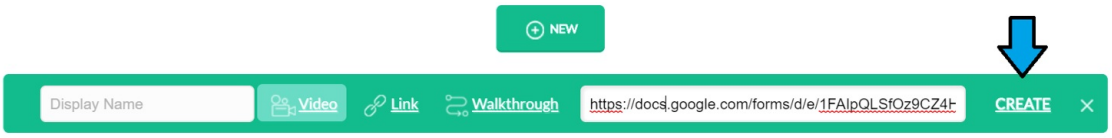


- You can minimize the Expanded Editor and either continue to the next Step or click “Done” and continue to the Guide Activation Conditions

From the Dashboard



- Click the New button for creating a guide and choose “Video” from the options in the green bar (1).
- In the URL box (2) enter the URL of the page you wish to embed
- Don’t forget to give the new guide a name (3)



- Now click **Create**
- A new guide has been added to your Video Guide list
- You can now choose the URL you wish this guide to appear on

Please note! Some products do not allow you to embed them inside of an iframe. We suggest you test each page to see that it works before launching it to your users.

Find the "Route" of a Fusion Apps Page

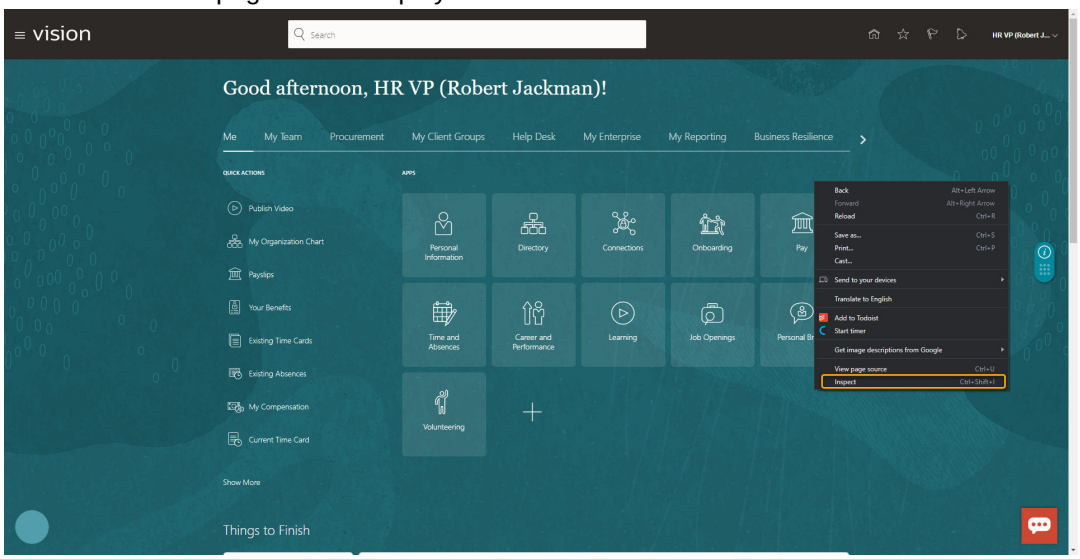
Describe when someone would need this information. For example "when connecting to wi-fi for the first time".

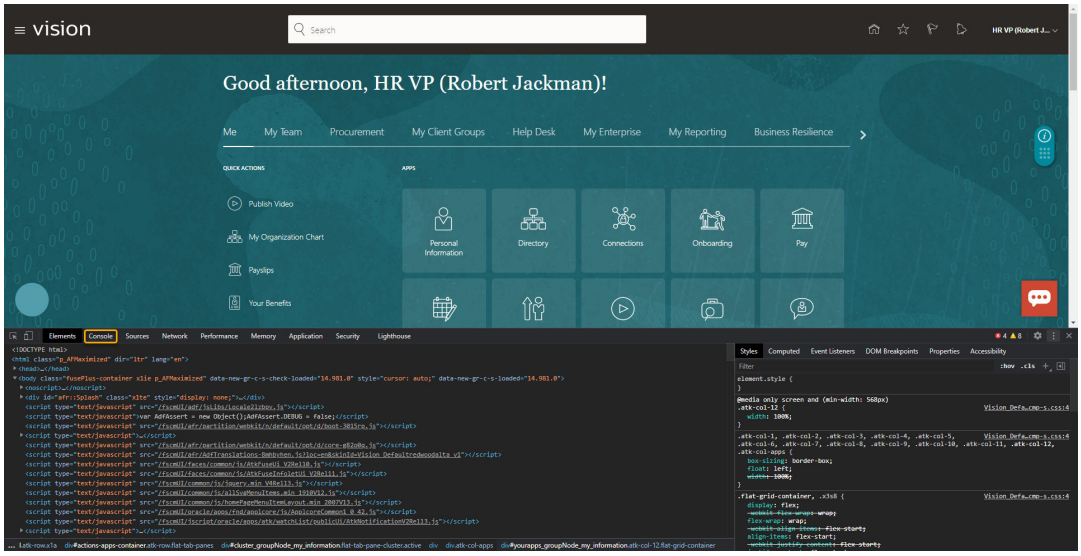
Instructions

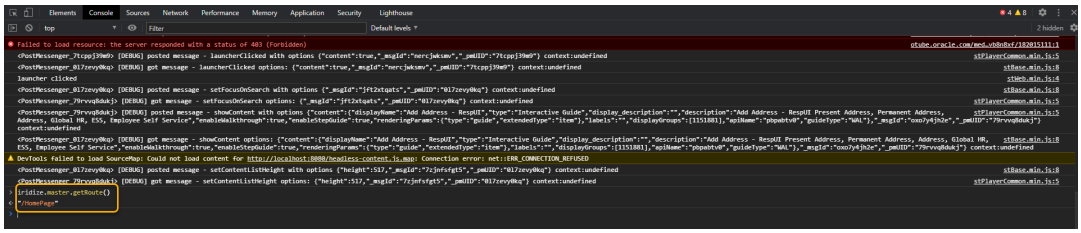
1. In a Chrome browser, right click anywhere on the page and select "Inspect"
2. In the Console tab of Dev Tools, enter

```
iridize.master.getRoute();
```

3. The route for that page will be displayed





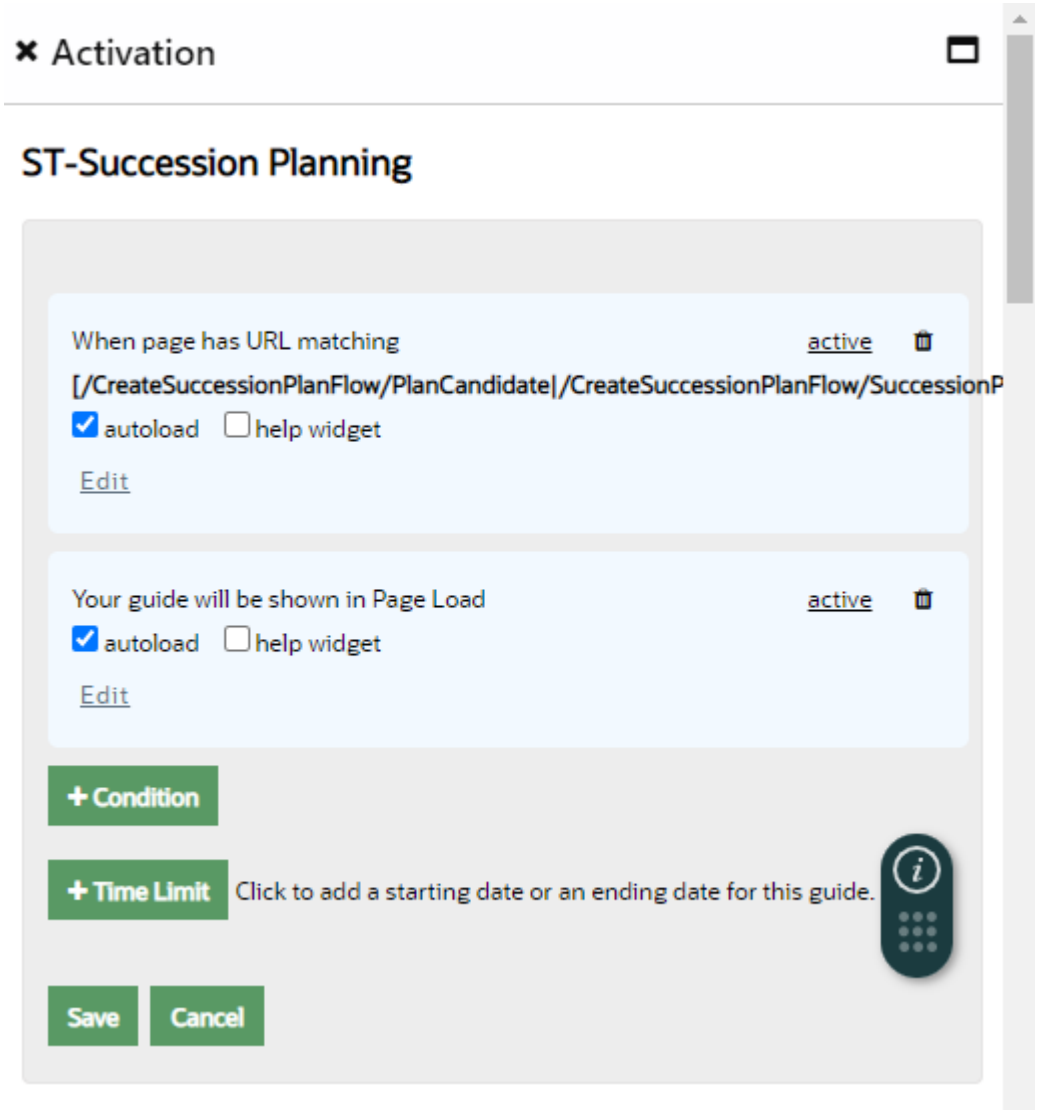


Guide Appears in Help Widget after Updates

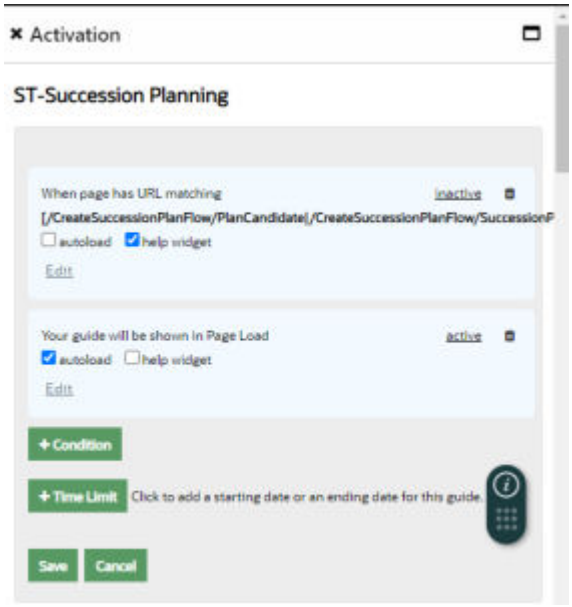
This is most commonly an issue when creating a Smart Tip guide designed to auto launch, but you not intended to show up in the Help Widget. If Activation Settings are done correctly, the guide will not appear in the Help Widget. However, if the guide is updated, the Activation Settings will automatically update to display the guide in the Help Widget. To avoid this issue with existing guides or new guides, follow the steps below.

New Guides

1. After you initially create the guide and select the "Done" button, you are brought to the Activation Settings window. Here you will see something similar to the following settings:



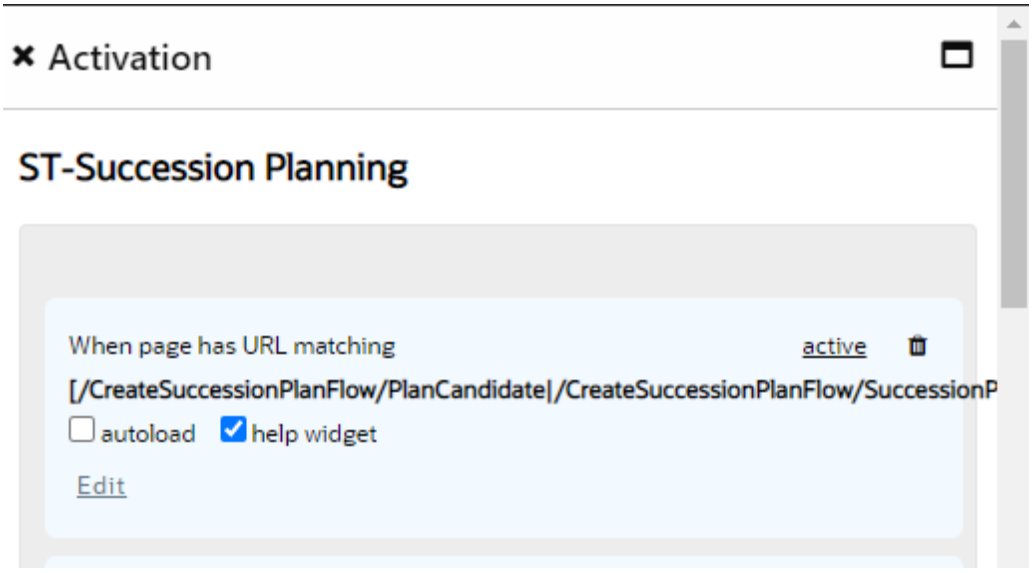
2. Assuming you want the smart tip to autoload and NOT show in the Help Widget, adjust the settings as shown below (change the "When page has URL matching..." condition to "Inactive" and the "Your guide will be shown..." Condition to autoload):



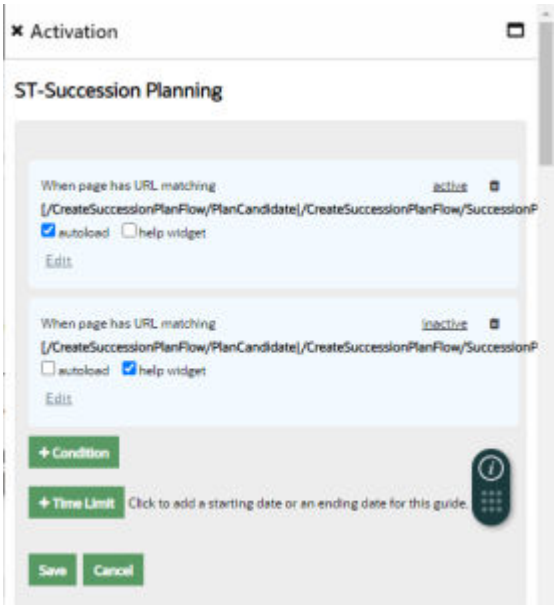
3. If you go in and make edits to this guide (add a new Smart Tip, for example, or edit the text in the existing Smart Tip), the Guide Activation Settings will not change.

Existing Guides

1. Go to guide that was previously created and examine the Activation Conditions.
   - There should be at least a segmentation condition with "URL matching" and help me checked. The segmentation condition will usually have a fairly long string defining the routes. If it is a small guide, there the URL matching condition may be relatively short.



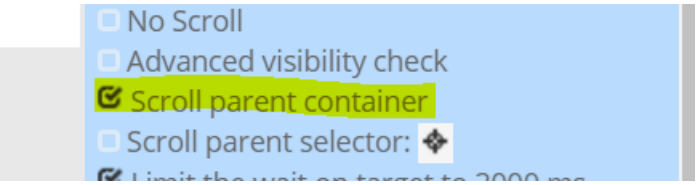
2. Set the segmentation condition to autoload
3. Open the guide, edit it ((add a new Smart Tip, for example, or edit the text in the existing Smart Tip), and select Done
4. Review Activation Conditions
   - You should now see an additional help widget activation condition, like this:



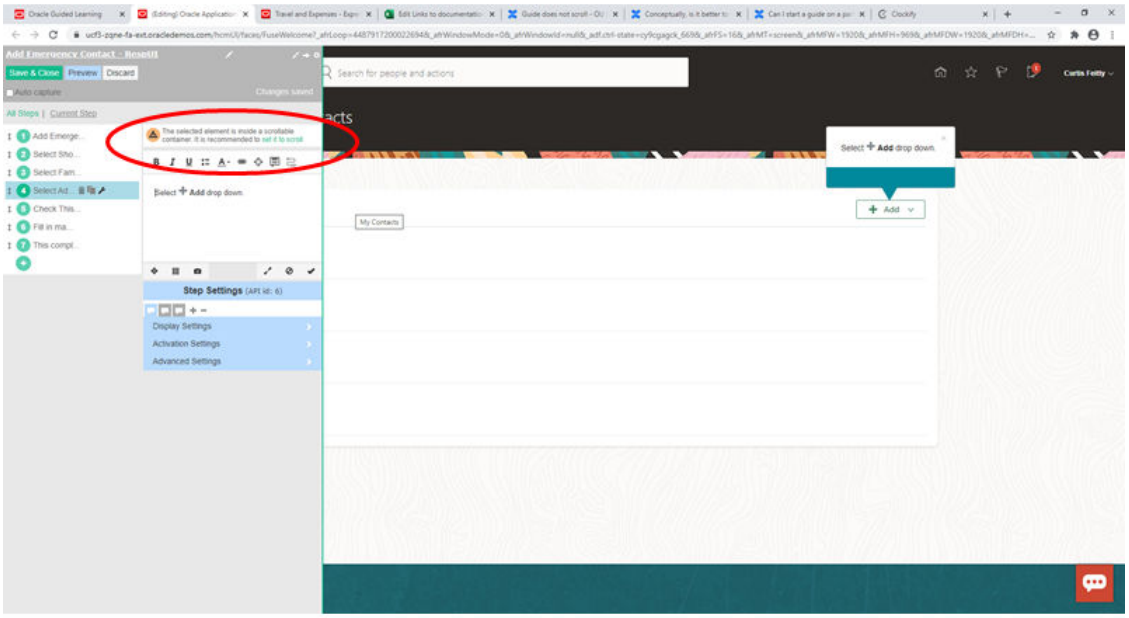
5. Set this additional help widget activation condition to inactive
6. Save settings
7. The next time you edit the guide, there should be no changes to the Activation Settings

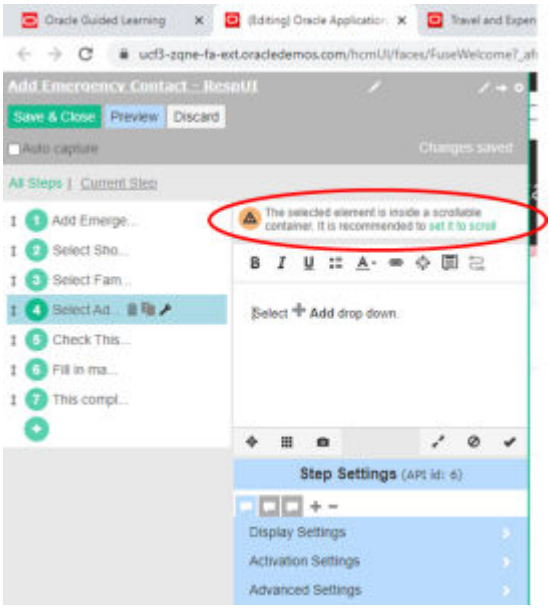
Guide Does not Scroll

Does your application use internal scrolling? That is, when you scroll the page does the top remain fixed? If so, the answer is yes. As a result you need to tell Guided Learning to use this scroll as opposed to the 'standard' vertical scrollbar of the browser. To do that you need to check the 'scroll parent container' option under the advanced settings.



Within the editor you will see a warning triangle when it is recommended for you to set the step to scroll. You can simply click on set it to scroll text in the warning or tick the scroll parent container box in Advanced settings





Guide Mysteriously Skipping a Step

If the guide you are working on is skipping a step, always check the DevTools Console tab first to see if you can troubleshoot the issue there.

```
<player> [DEBUG] {"msg":"doStep - loop over allSteps","numSteps":1,"stepId":"17"} context:"scenarioId - ujwei5xw"
<player> [DEBUG] {"msg":"doStep - loop over allSteps","numSteps":1,"stepId":"18"} context:"scenarioId - ujwei5xw"
<player> [DEBUG] {"msg":"doStep - loop over allSteps","numSteps":1,"stepId":"22"} context:"scenarioId - ujwei5xw"
<player> [DEBUG] {"msg":"doStep - loop over allSteps","numSteps":1,"stepId":"19"} context:"scenarioId - ujwei5xw"
```

The highlighted line is the one that represents the step being skipped. Based on this log file, it looks like the step was advanced normally, but it is not appearing when the guide is run. If this is what you are seeing in the Console, try the following.

Instructions

1. In the step that is skipping, go to the "Activation Settings/Show tip after line" value
2. Enter 5 seconds
3. Retest
4. If the guide now works properly, experiment with reducing the number of seconds to get a value that is as small as possible while ensuring that the step displays properly

This has been seen when the previous step is a drop down value being chosen. What is happening is that the click on the drop down is "echoing" and getting registered in the following step. The delay allows the click to complete before the next step appears, thereby avoiding the skip.

Guide not Starting on First Step

Problem Description: Guide doesn't start on first step

Steps to reproduce: Start the guide from the Help widget. This is an important detail - if you start the guide from the DevTools console, it will work properly. So, for example, when you choose a guide from the HelpMe widget, instead of starting on the first step, the guide starts on 8th step

Resolution

```
iridize.adfApp = true;
```

Add the above to the config.js file in the extension.

Sample:

++++++++++++

var ir_app_settings = {

"H2KN4GjLSO+zPiyRhWDgKw": [

'iridize.env="dev";',

'iridize.min=""; //this line is only for testing. remove this line in production.',

'iridize.useSessionStorage=true;',

'iridize.contextualMode=false;',

'iridize.autoSegmentation = true;',

'iridize.adfApp = true;'

]

};

var ir_url_mapping =

{ 'https://ucf5-zaxe-fa-ext.oracledemos.com': "H2KN4GjLSO+zPiyRhWDgKw" }

;

++++++++++++++

Guided Learning Does not Respect URL Condition

Detailed Question:

When testing for a Splash message it seems that Guided Learning does not respect the activation URL. When testing in dev environment the message pops up as soon as a person logs in to the dashboard.

Answer:

Try to log in using a **new** incognito window/tab.

Maybe you already had this guide running in this browser tab. Because Guided Learning stores the state of all guides per tab, if the guide already ran in this tab, even if with a different user, Guided Learning will remember than and retry to launch the guide.

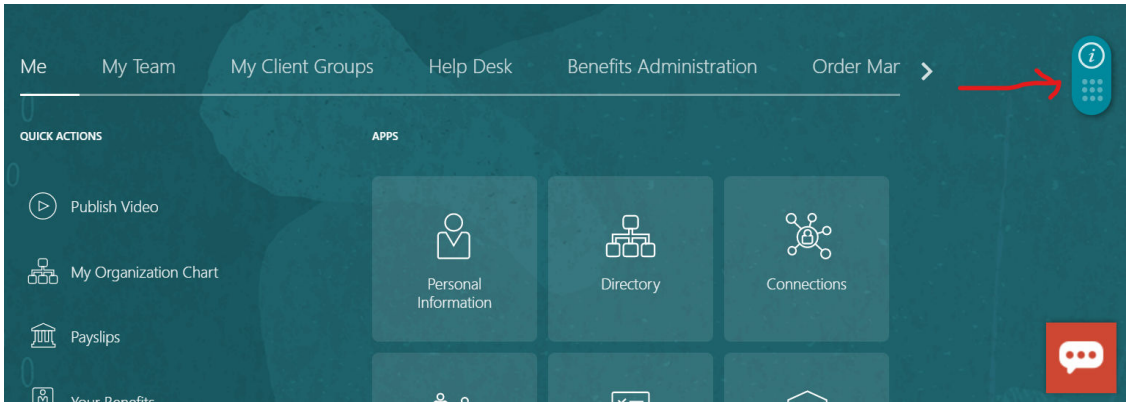
Also, I would recommend to remove the sticky setting because the guide should only appear on this URL and go away if the user navigates to a new page.

Help Widget Explained

What is the help widget?

The Help Widget is intended to be a one-stop-shop for customer self service. Used correctly, it should be the first place your users go to to search for help and assistance on your application. Placing the right content into the Help Widget will reduce the number of support requests and drastically improve the onboarding process of users.

Technically speaking, the Help Widget is an HTML component that sits on top of your application. Normally it contains a list of step-by-step, walkthrough guides and important links (e.g. a link to the product knowledge-base). By default the Help Widget is located on the right hand side of your application (see image below).



Delivering contextual help

The content of the Help Widget is completely dynamic. As the training professional, you have all the power to control what content it holds directly from the Guided Learning dashboard without requiring any IT resources.

As a result the Help Widget is a great way to deliver contextual help to your users. With it you can:

- Show content that is only relevant to the page that the user is currently on, using page conditions
- Show content based on the role of the logged in user, using user fields. For example, you can display two completely different guides to a sales rep and a sales manager.
- Show content based on user activity. For example, only show a link to guide 'update opportunity' if the user has already completed the guide 'create opportunity'.

All of this is possible using guide activation conditions.

Extended features

Customers can further leverage the Help Widget by posting links to training in the corporate LMS, policy documents and videos

Guide search

Allows your users to search for other relevant guides on the Guided Learning platform. That is, guides that are not displayed in the Help Widget but match a certain keyword. Your user will then be able to launch these guides directly from the search results.

How do I make an image a session variable?

Use case:

You want to branch a guide based on user's choice. You have already setup the session variables but instead of the buttons you want to use an image.

To do so you will need to open the extended editor and click on the source button to display the raw HTML source of your tooltip. You should see something like this:

```
<a class="next-btn--choice" data-iridize-role="choiceBt"
data-iridize-vars="{&quot;v1&quot;:&quot;yes&quot;}" href="javascript:void(0)">
yes
</a>
```

to replace the button with an image you will need to replace the yes (on the 3rd line above) with an img (image) html tag.

```
<a class="next-btn--choice" data-iridize-role="choiceBt"
data-iridize-vars="{&quot;v1&quot;:&quot;yes&quot;}" href="javascript:void(0)">
<img src="mypicture.png" />
</a>
```

How to configure Country Segmentation

Add the following code to the custom script:

```
window.ir_fields[“user_country”]=”#{Profile.values.FND_TERRITORY}”;
```

I cannot preview a guide that my colleague created

To preview a guide in the application, Guided Learning needs to be embedded in the application or you need to our browser extension installed locally.

To preview a guide through the OGL console, you need to have been granted the correct user access.

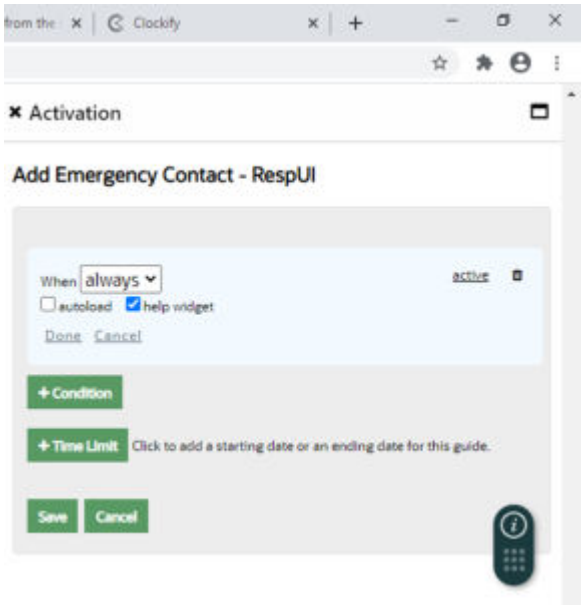
I have made some changes to a guide and saved them accidentally, can this be reverted?

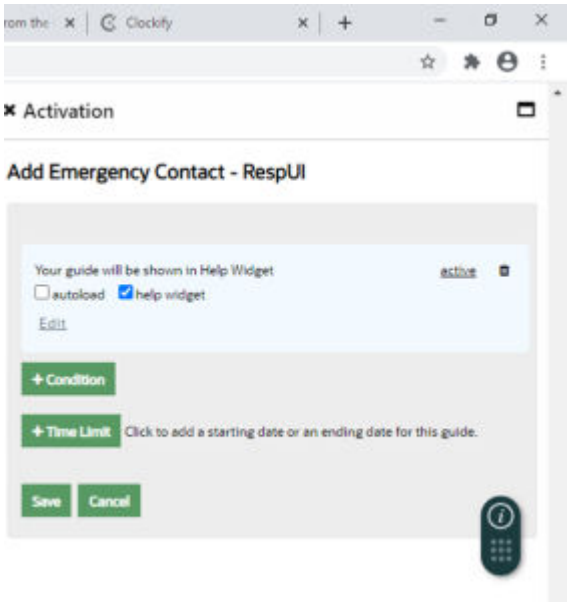
Yes.

When you edit a guide a new revision is created.

I have successfully setup Guided Learning but the Help Widget does not appear

Make sure that at least one guide is set to "always" display in the Help Widget by using the guide activation conditions.





Note: If using Guided Learning in "prod" mode, please give any changes you have made to the activation conditions around 10 minutes to propagate. You may also need to clear your browser's cache.

I need to create a guide on a page with a dynamic URL, what are the best practices?

Let's solidify this with a concrete, real life example:

You offer a document signature service; each time someone creates a document for signature, the URL changes based on random, system generated, document ID. The url looks like this:

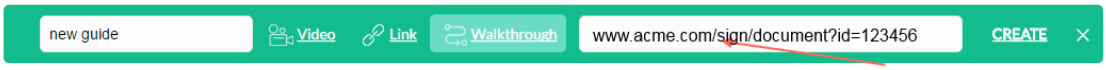
```
www.acme.com/sign/document?id=123456
```

Before anything, read the URL configuration for a guide FAQ.

Now let's break things down to work flow steps.

Creating/Editing the guide

To create your guide, simply choose a random document and enter the url in the Guided Learning guide creation form:



If you want to edit the guide and the document no longer exists, no problem, you can change the Editing URL. Also, once you click on the guide to edit it you can navigate to the desired page, the Guided Learning editor should stay with you on the left hand side.

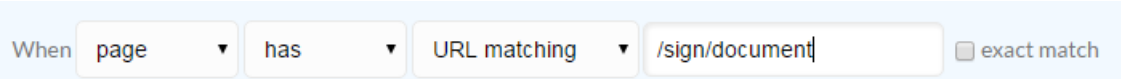
Previewing the Guide

Opening the preview of the guide should take you to the same random document you choose when creating the guide. If you want the preview to start on another page you can either change the URL of the guide or navigate to the relevant page as you normally would with the browser until you reach the right page - the guide should start automatically.

Launching the Guide

Assuming you want to automatically launch the guide on all these pages, no matter what the document ID is, you will need to URL Matching condition in the guide activation conditions.

Based on the above example the condition would look like this:

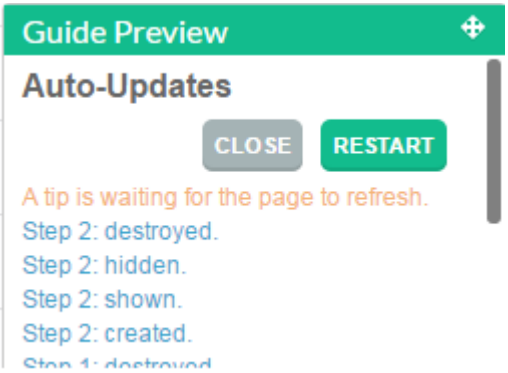


Notice that the we only tell Guided Learning of the part of the URL that matters to us. Since we want the guide to show on all documents we simply omit the id parameter from the URL condition. Alternatively if we want the guide to launch only on a specific document we could say:

```
/sign/document?id=123456
```

Furthermore, Guided Learning is configured by default to ignore the domain names. this means that if your application runs on multiple sub domains you will not need to create special guides and conditions for each sub domain.

I receive the message "A tip is waiting for the page to refresh." What does that mean exactly?



This message means that the tooltip on the next step, probably step 3 is waiting for the page to refresh.

To change this behavior, open the 3rd step in the Guided Learning editor and under the activation settings, unset the ***Show tip after page refresh*** setting.

When should I use this setting?

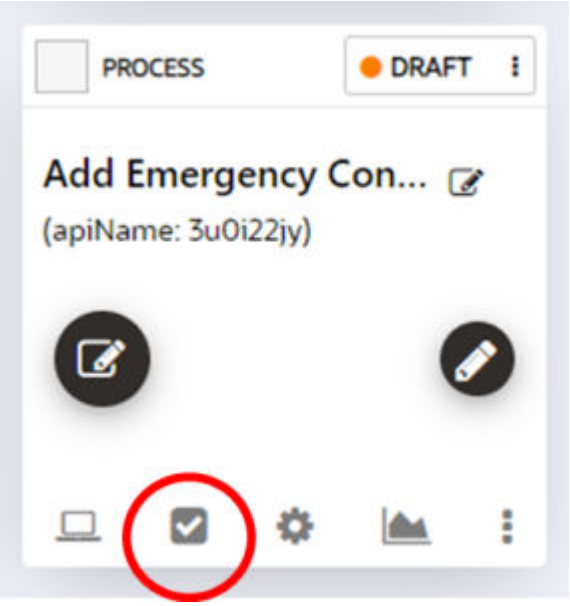
Consider the following situation, you have a guide for running a company search on LinkedIn. Your first step will be to enter a company name in the search field and than to click on the magnifying glass in order to run the search. For the third step, you would like a ***splash*** screen that will pop up once the search results are loaded.

If you do not set the '***Show tip after page refresh***' setting, as soon as you click on the search button the Splash Screen you set up in step 3 will appear, even though the search has not finished running. Then LinkedIn would load the search results, and the Splash Screen will display again. To avoid this behavior the '***Show tip after page refresh***' should be set to delay the splash screen from appearing until after the search has completed and the page has refreshed.

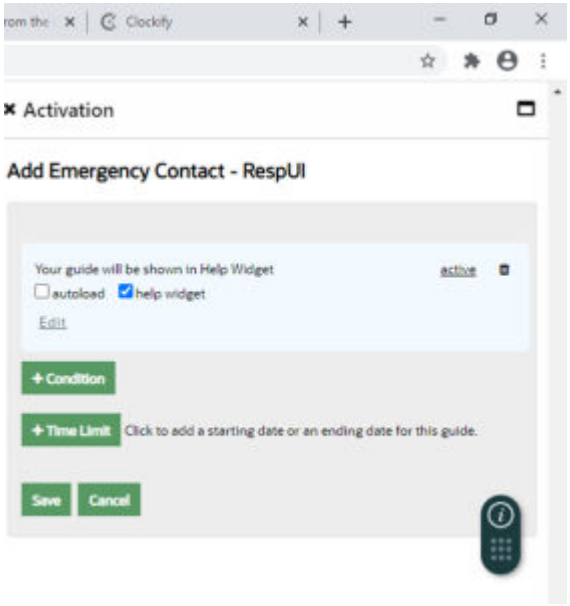
I setup Guided Learning and none of my guides are showing, what's missing?

Please verify that at least one of your guides has active activation conditions.

Show me how



1. Click on activation settings in any guide



2. Check that at least one activation condition is set to show flows in the widget. This example is 'always show.'

I want a guide to show on 2 different pages with different URLs. How should I set up the conditions?

When you create a URL matching condition you can use a regular expression.

To tell Guided Learning that the string is a regular expression surround it with square brackets [ ].

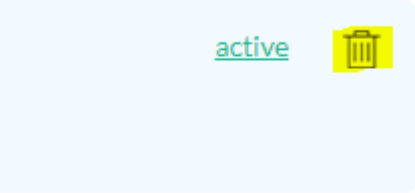
So, let's say you want your guide to display if either the word “homepage” or “adlist” appear in the URL.

The regular expression for that is homepage|adlist and with brackets ***[homepage|adlist]***

If I want a guide to only show when referenced/linked from another guide but I do not want it to be in the help guide list, how do I set the guide activation?

You will need to delete all activation conditions for the guide.

Click the Trash icon for all of the activation conditions in the guide



After doing so you will see this warning, which is exactly what we want.

"For your guide to be shown it must have at least one active condition."

Then click OK to save your work.

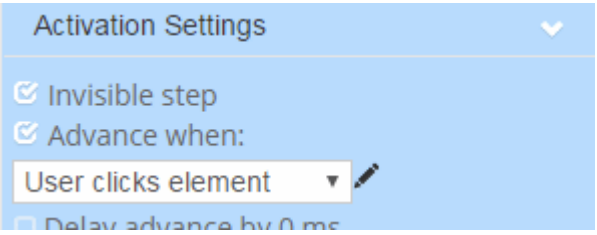
Invisible Steps: React to User Interactions

- Is it possible to start a guide after a user clicks on a certain button?
- Can I launch a guide after a user has started filling out a form?
- Can I wait for a user to click on a button and continue the guide only afterwards?

The answer to all these questions is **YES**, that's what the '**invisible step**' is for.

Normally, any step in a guide/walkthrough will have at least one tooltip that is displayed on the screen and assists the end user. An **invisible step** has no visual representation. But, it still has all the powers of a normal step; it can still wait for any type of user interaction.

For example, the settings below will wait for the user to click on an element before showing the next tooltip.



Similarly, you can wait for the user to enter text in any form, or to click on the TAB key.

ADVANCED: If you leave the 'Advance when' setting unchecked the player will advance to the next step as soon as the target element is found on the page. An example of this would be skipping navigational steps.

Is it possible to change the font to an option not available within the dropdown list? We would like to add a font called "Proxima Nova" if possible.

Changing the drop down is not currently supported.

But, if your font is supported by all common browsers; or if you already downloaded this font to the end user's browser, you can simply switch to source view in the Guided Learning editor and change it. like so:

```
<p style="font-family: "Times New Roman", Times, serif;">text in Times New Roman</p>
```

Text is in Times New Roman.

If you want to globally change the font of all text in your guides download our design kit from the theme page.

Is it possible to change the text on the buttons in the interactive tour? Either by having the same language shown for all users, or making the button text dependent on a user setting?

You can update the default text on the buttons for all guides and all users.

We call this a **tiplate** (tip + template).

If you want to update the text of the next button on a specific guide for all users you can do that using the setting 'Change Next button text to'. You will find it in the display setting in the Guided Learning editor.

If you want to change the text of the next button on a specific guide based on a certain user setting you can do that too.

First you will need to set an Guided Learning variable like so:

```
iridize("api.vars.set", {"username":"007"});
```

than you can set the 'change next button text' setting to:

```
{{username}}
```

and the button will then display 007 (instead of next)

Is there a way to search for guides by keyword through the Guided Learning JS API?

Guide searches happen on the client-side in order to avoid extra communication with the servers.

We do not currently have a dedicated API function for searching but we do have an API function that lists all the guides that you can use as described below:

```
function search(query){
    iridize("api.guide.list",  {}, function(data)  {
        var guidesList,guide,i;
        // get the array of guide information objects
        guidesList = data.guides;
        // print the guides to browser console
        for (i = 0; i < guidesList.length; i++)  {
            guide = guidesList[i];
            if(guide.displayName.indexOf(query) != -1){
                console.log('name matches query: ' + guide.displayName);
            }
            if(guide.description.indexOf(query) != -1){
                console.log('description matches query: ' + guide.description);
            }
        }
    });
};
```

My application uses popup windows/multiple browser tabs, what is the best way to create guides?

In case you want to instruct your users through a process that spans multiple tabs or windows you will need to create a separate guide for each window and link the guides together.

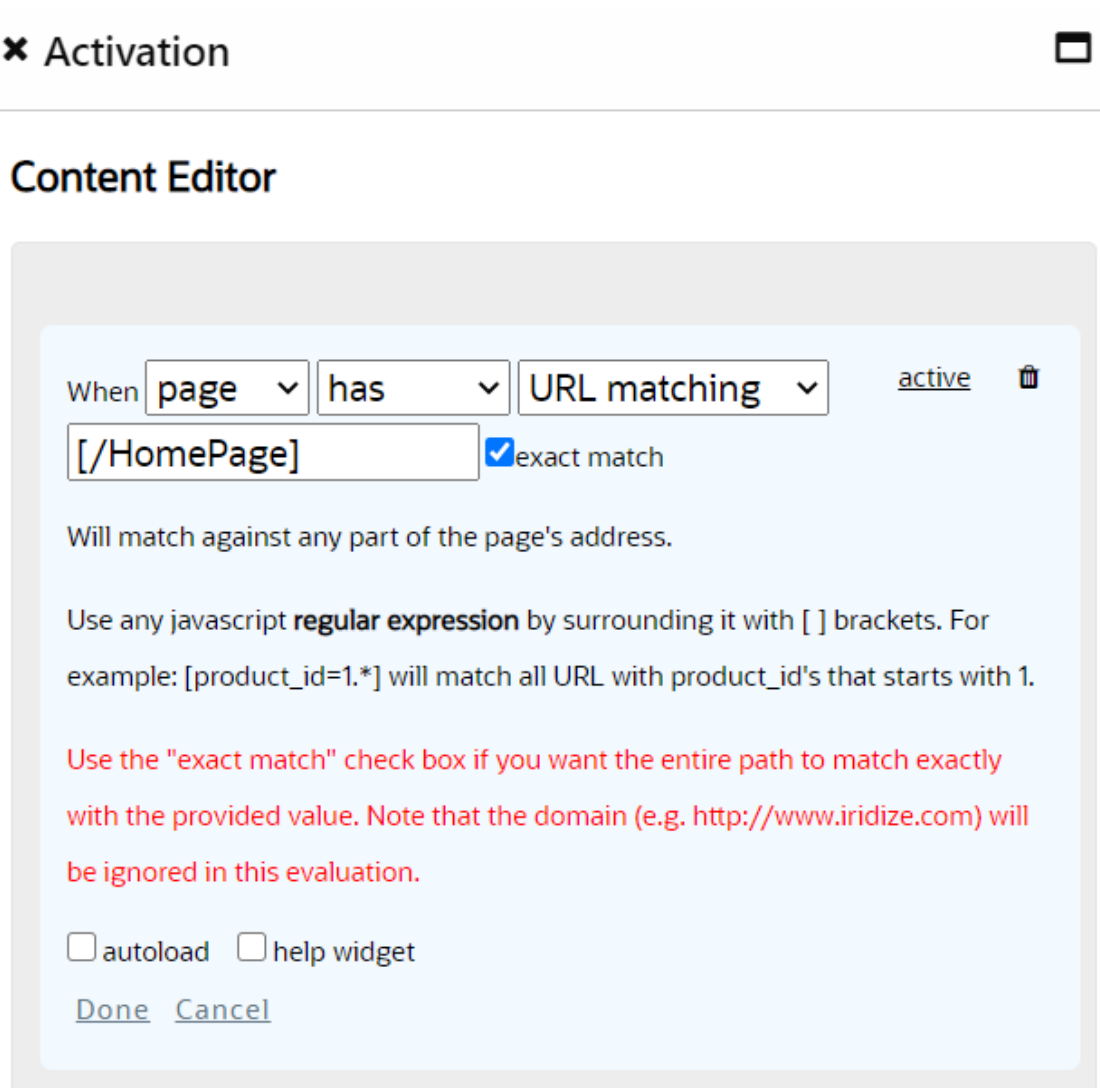
Often, you will not want to display the help widget in such popup windows. If this is the case use the special API setting to disable the help widget.

```
iridize.showStartPanel=false;
```

My guide does not load on the right URL

In order to support running the same set of guides on multiple domains and sub-domains, by default, Guided Learning does not take into account the domain name when evaluating page conditions.

If you'd like to be sure that the domain name is taken into account, either set the **exact match** flag or use a regular expression.



No checkboxes were selected

Let's quickly explain what the following error message means:

```
Failed to save conditions: Condition 1: No system tags selected.
```

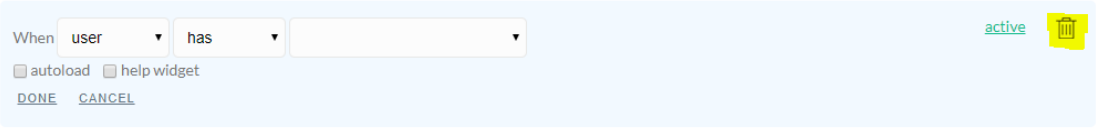
Every activation condition must be assigned to at least one 'activation mechanism'.

Currently Guided Learning support 2 such activation mechanisms.

1. **Autoload** - Launch a guide automatically as soon as the page is loaded.
2. **Help Widget** - Display the guide in the help widget.

If you don't specify at least one of the activation mechanisms Guided Learning will not know when to apply a given condition.

If you want your guide to have no activation conditions, this can be very handy if you want to create a **launcher** of an **email campaign**, you need to delete all conditions by clicking on the trash-can icon.



Our product is internal, behind a firewall, can we still create Guided Learning guides for them?

Yes.

Guided Learning uses a browser extension to create and manage your guides. As long as your computer has access to both the internet and the internal network you will be able to create guides.

Remove Border from Table

When you create a tip with a table in it, say, for formatting purposes, you don't want the border to be visible. In order to have no border, the following must be added to the theme:

```
div.sttip table, div.sttip tr, div.sttip td {

 border: none;

 padding: 6px 9px;

}
```

Scroll Parent Container: Handling Internal Scrollbars

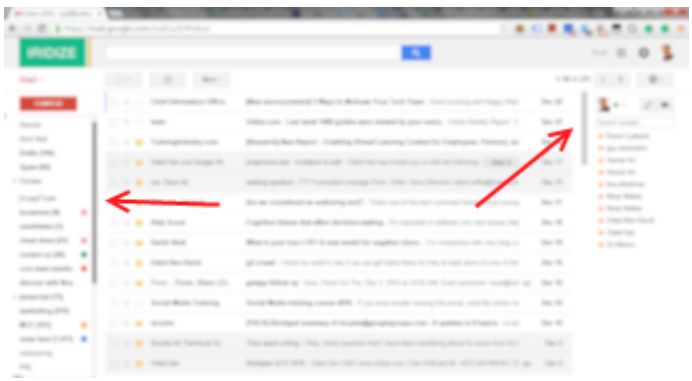
Overview of Scrollbars

To understand this setting we must first understand what a scrollbar is.

"A **scrollbar** is a graphical control element with which continuous text, pictures or anything else can be scrolled including time in video applications, i.e., viewed even if it does not fit into the space in a computer display, window, or viewport."

You would normally see a scrollbar on the right hand side of your browser. It allows you to scroll the text up/down in order to see the whole page. Sometimes, however, a web application would include additional scrollbars.

Gmail is a great example of this (see image below with 2 internal scrollbars highlighted).



Back to Oracle Guided Learning...

Let's say you want to create a guide on a page with an internal scrollbar. And let's say that you need to scroll that internal scrollbar in order to highlight a certain element in your guide.

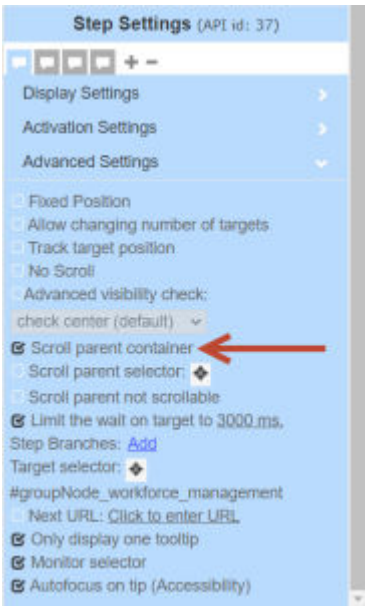
Notice that in order to view this element (for example a certain email in your gmail inbox) you need to use the internal scroll. Scrolling the standard browser scrollbar on the right will not help at all.

When you run this guide, Guided Learning will try to automatically sense that you element is contained in an internal scroll but sometimes it will not succeed in doing so. For this reason, by setting '***Scroll parent container***', you can explicitly tell Guided Learning that it needs to scroll an internal scroll to bring a certain element to view instead of scrolling the default browser scrollbar.

Scroll Screen to See the Section where the Tip is Pointing

This is the default behavior for any tooltip. If it doesn't scroll automatically it normally means that you have an internal scroll.

Try setting the *'scroll parent container'* in the advanced settings menu.



Select the First of Multiple Same Elements

Problem

Often encountered in tables, if you have multiple elements with the same name, how do you select a specific one based on position in the list

Solution

If the element you want to select is .subTextCustomer.wordEllipses but there are multiple such elements on the page, and the only one you are interested in is the first one, then use this to select:

```
.subTextCustomer.wordEllipses:eq(0)
```

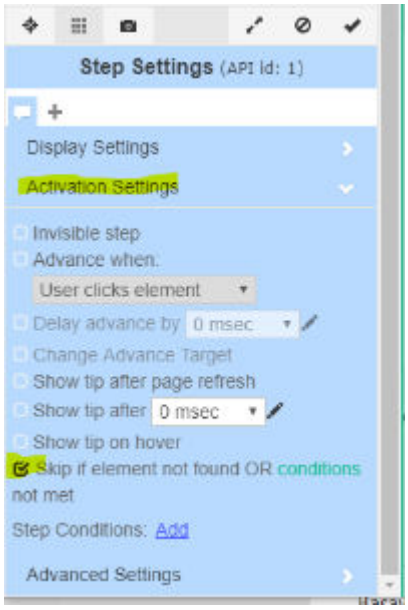
Skip Step not Working

When trying to make a step skip based on the target element not being visible, sometime the guide does not skip the step as expected. When this happens, check to make sure that "Monitor Selector" is not chosen.

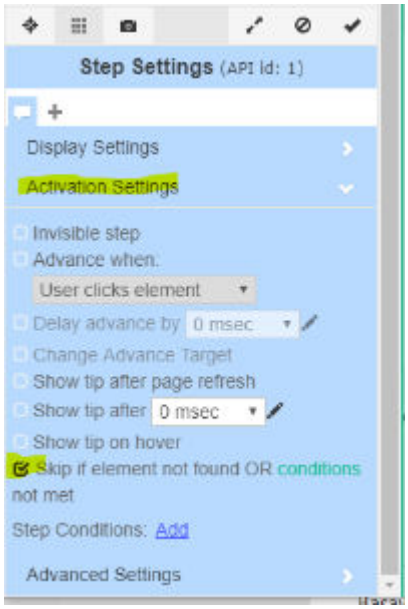
Instructions

Add the steps involved:

1. Select "Skip if element no found..." option in the Activation Settings section



2. Make sure that Monitor selector option in the Advanced Settings section is NOT checked



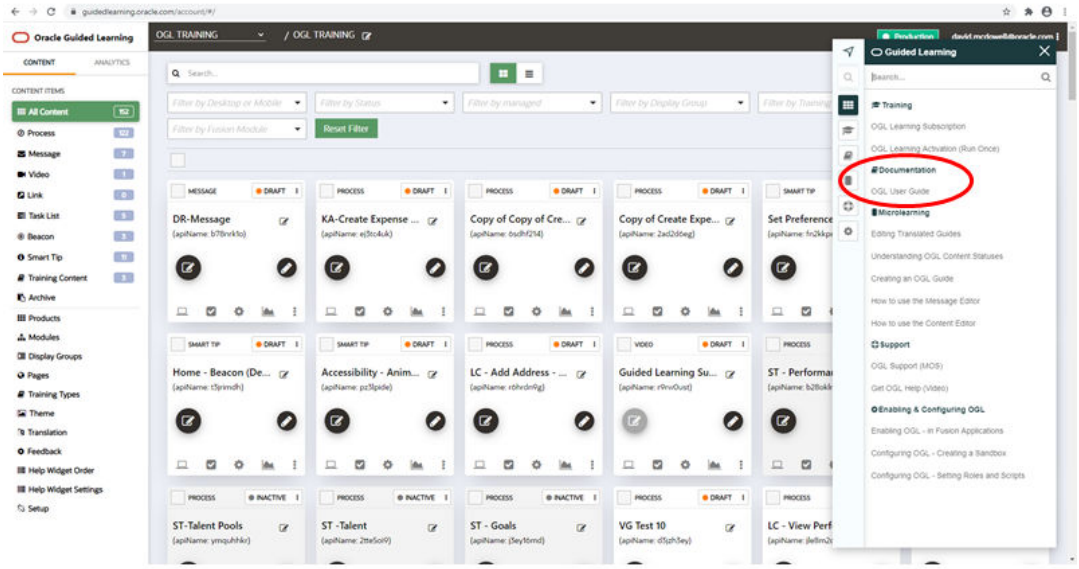
Unable to Edit Guide Options

Whenever you edit a guide, Oracle Guided Learning locks it so that nobody else could edit it while you are. Moreover, if you edit a guide in one browser tab or window and try to edit that same guide in another tab Guided Learning will block you from doing that and issue the following warning to the screen (as can be seen in the image above):

This scenario is currently opened for editing in another window. You may only view the options at this time.

Normally this could happen if:

1. **A colleague is editing the guide**: If this is the case there is nothing you can do but ask the colleague to close his editing session.
2. **You have the guide open for editing in another window/tab:** Go to that tab and either make your change there or close it.
3. **Guided Learning thinks that you have the guide open for editing in another window/tab but you don't:** This could happen if you indeed opened the guide for editing and closed the browser tab/window without clicking on discard in the Guided Learning editor. To unlock the guide in such a situation, open the guide for editing again:



4. Click the Pick up where you left off link. Once the Guided Learning editor is loaded, click on the Discard button. This should unlock your guide and allow you to edit it again.

Use an Image Map to Launch a Guide

An image map is an image with clickable areas.

What if you want that if the user clicks on one of these areas an Oracle Guided Learning guide would launch?

To define an image map you can use online tools like https://www.image-map.net/ or follow this tutorial https://html.com/images/how-to-make-an-image-map/

After you have created your image map, you will need to add a few attributes to each area in the map. These attributes will tell Guided Learning what to do when the area is clicked. The best way to get these attributes is to create a Launch guide button and copy these attributes from it.

For example, your area could look like this

```
<area alt="Sun" coords="0,0,100,99" shape="rect" />
```

and you will change it to:

```
<area alt="Sun" coords="0,0,100,99" data-iridize-
nextscenario="{&quot;nextScenario&quot;:&quot;fd0qtw7w&quot;,&quot;dontClose&quot;:false,
&quot;next&quot;:&quot;&quot;}" data-iridize-role="nextScenarioBt"
href="javascript:void(0)" shape="rect" />
```

We have clients who use different terminology such as "Intake" instead of "Class". Is there any way to have two different terminologies depending on the viewer?

Yes.

Use our javascript API to set the preferred terminology for the user.

```
iridize("api.vars.set", {"terminology":"Intake"});
```

If you have already set this up using api.fields.set, you can skip this.

Then, in any place in your guide you can use *{{terminology}}* and the Guided Learning player will know to replace it by *"Intake".*

We use a different subdomain (sub-domain) for each customer; can the same set of guides be used on all domains by all customers?

All of your guides in Guided Learning are grouped into what we call applications. An application is essentially the product that you want to use Guided Learning on, this can be your very own SaaS offering or any other web application. Each application has a unique identifier.

When you embed Guided Learning, as part of our Javascript API you state which set of guides you want to use by providing the application identifier.

You can provide Guided Learning with the same application identifier on multiple domains, sub-domains, or app instances; as a result the same set of guides will be used in all.

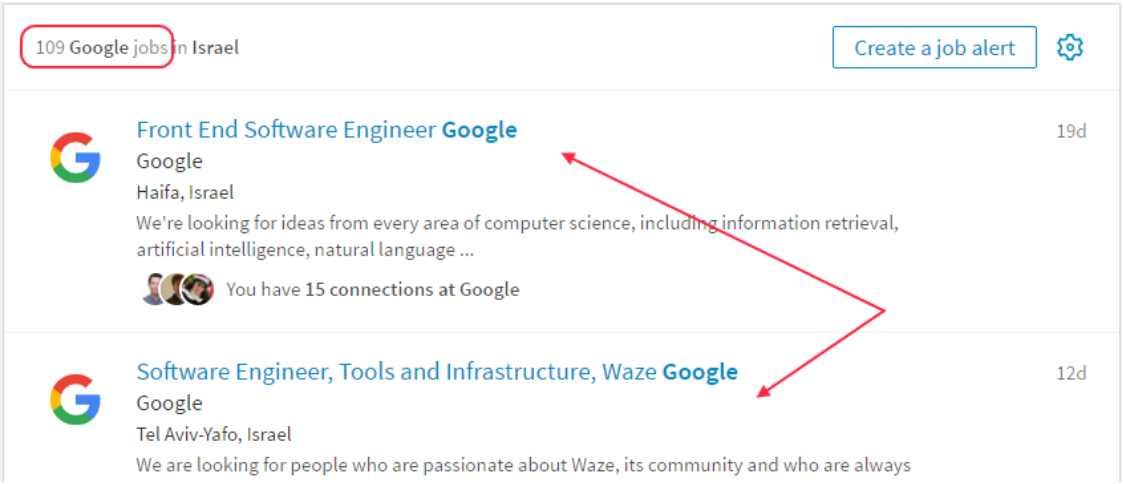
What if I want the guide to advance when user clicks on ANY element in a list?

No problem.

Actually you can choose from one of the 2 options below depending on your preference.

Example

Say we want to create a guide for LinkedIn on how to apply for a job @ Google. You explain to the user how to run a Job search and now, when the user is on the search results page you want the guide to continue no matter which job the user selected from the list.



There are 109 google jobs posted on linkedin and we want Guided Learning to grab a click on all 109!

Our guide is not one step #3 and we are instructing the user to **"click on any job for the list below".**

Option 1 - Multiple Tips

Instead of trying to identify a click on any one of the 109 jobs we will add another tooltip to step #3 that will only display if the user is on a job listing details page (obviously this means s/he clicked on an item from the list)

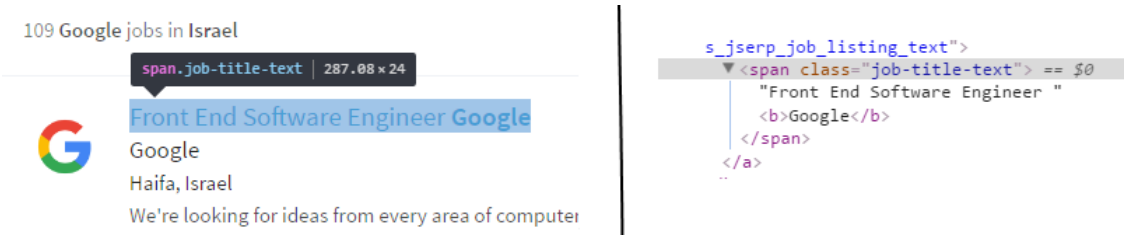
To do this we need to:

1. Add another tooltip to step #3
2. Add a condition to this tooltip so that it will only show on the details page. It is common to use the URL condition or a visible element condition.
   In the LinkedIn example, I would have used, 'when page has url matching /jobs/view/'.

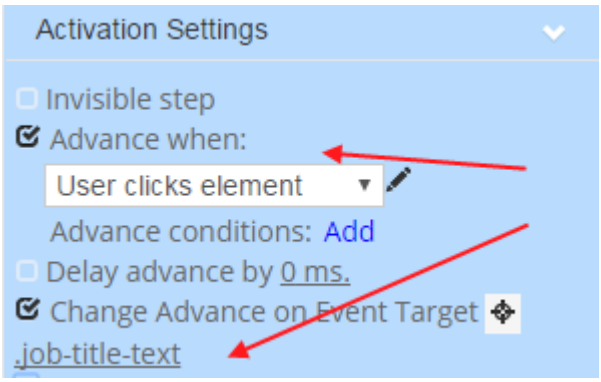
Option 2 - Manual Selector

To follow this option you must have a good understanding of CSS and specifically CSS selectors.

When you place an Guided Learning tooltip on a page, Guided Learning works hard to find the best suitable identifier for the element you want the tooltip to point it. A crucial part of this is to find a unique identifier that will match only the element that you selected. Here, we want to bypass this mechanism and choose a selector that will match all of the search results.



In our example, all job listing links share a common class "job-title-text". In order to have Guided Learning wait for a click on ANY one of them, we need to check '***Change Advance on Event Target***' and manually change it to ***.job-title-text*** .The '.' prefix is not a mistake, it denotes a CSS class.



Summary

Each of the above options has its merits. Option 2 is quicker to set up, Option 1 does not require any technical knowledge and creates a much better user experience whereby if the user goes back to the search results page the tooltip to **"click on any job for the list below"** will display.

What is Advanced Visibility Check?

On some applications, OGL might think that an element is visible while it is not. This can happen if the web application uses multiple layers and an element could be hidden behind another one that was added on top of it. This setting will run some extra checks against the element.

Advanced Visibility Check asks the browser; what is the topmost element given coordinates on the window? If the element whose visibility we test is returned by the browser as the topmost element than we conclude it is indeed visible.

This setting is not used by Guided Learning by default because it is somewhat resource intensive on the browser.

The advanced visibility check can be turned on under the advanced settings of a step or when configuring step conditions.

What is fieldsNotSet error?

When using Oracle Guided Learning in a Single Page Application (SPA), sometime the embed code is executed before the user has logged into the application. In this situation one must call

```
iridize('api.fields.set', {})
```

This tells Guided Learning that this is an SPA and to wait for the user to be set before loading any content.

If within 5 seconds after the page loaded the user is still not set Guided Learning will issue the **fieldsNotSet** error to the browser's console.

This error log was added to assist developers in embedding Guided Learning. The error does not affect the loading of Guided Learning and as soon as **'api.fields.set'** is called the content will be loaded correctly.

What is length of Description Field?

The limit is 2,147,483,647 characters.

However please keep in mind the user experience.

What is routeNotSet error?

When using Guided Learning in a Single Page Application (SPA), one must call

```
iridize('api.route.wait', {})
```

This tells Guided Learning that this is an SPA and to wait for the route to be set before loading any content.

If within 5 seconds after the page loaded the route is still not set Guided Learning will issue the **routeNotSet** error to the browser's console. This error log was added to assist developers in embedding Guided Learning. The error does not affect the loading of Guided Learning and as soon as **'api.route.update'** is called the content will be loaded correctly.

To configure this timeout set **iridize.setRouteTimeoutMillis** in your embed code.

What is the best way to embed a simple audio file into a step?

Guided Learning does not offer a hosting service for audio files.

Assuming your file is hosted elsewhere you should be able to use the media embed button in the extended editor to embed it.

What is the difference between the two Oracle Guided Learning deployments?

Guided Learning supports two deployment modes: production and development.

You define your deployment mode in the embed code by setting the env variable.

```
iridize.env=["prod" | "dev"]
```

When I copy the URL of the page I want the guide to start from, it does not load correctly in the browser

Some web applications are sensitive to pages being accessed directly as opposed to normal site navigation.

If this is the case, when creating the guide, simply copy the homepage or dashboard URL in the OGL editor and once the page is loaded simply navigate to the desired page as you normally would.

Why doesn't my guide show up?

If you have created a guide and it does not show up, look no further, you have come to the right place.

This article outlines a checklist that you can go over in order to get to the bottom of the issue.

Is Guided Learning embedded?

Before we do anything, let's make sure that Guided Learning is embedded on the page where the guide should appear.

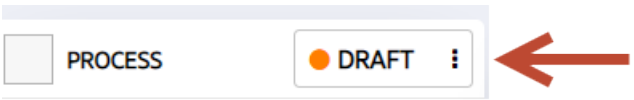
The easiest way to verify that is to check if there are other guides displaying on this page. Even if the help widget appears it means Guided Learning is embedded properly.

If Guided Learning is not embedded you will need to update the JS in the Fusion instance.

Is the guide published?

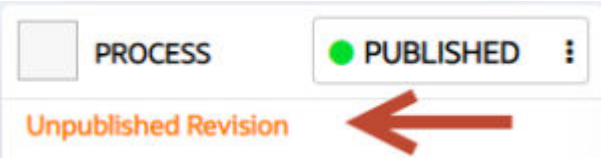
Every Guided Learning guide has multiple revisions. This is important if you want to update a guide that is currently in production with new new content that is only available in your staging/ UAT.

If your guide does not have a published revision it will not show in production at all.



[A draft guide is indicated by the orange ***DRAFT*** label]

If the latest revision of the guide is a draft you will only see you latest changes in the staging/UAT environment.



Unpublished Revision indicates that the latest revision of the guide is a draft.

In both situations, you will only see the guide in your staging/UAT environment. To see the guide in production click PUBLISH to publish any recent updates.

10

Multi Language Support

This chapter contains OGL Multi Language Support knowledge articles.

Exporting and Importing Guide Content

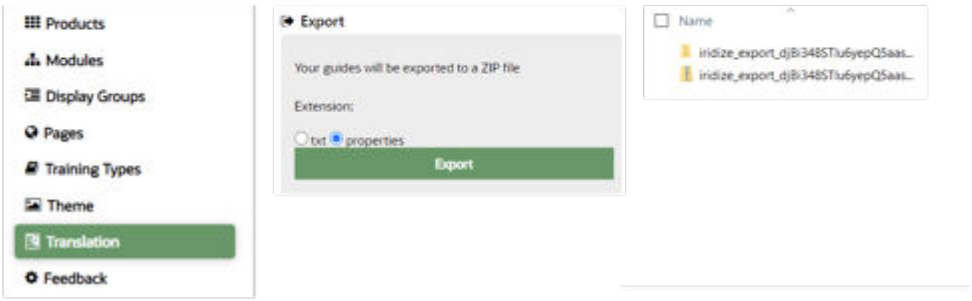
This section explains the process for exporting and importing Guided Learning guides for translation purposes. Translation of guides allows you to dynamically control which language of the guide will be displayed to the end user. Once the guides are exported you can translate the relevant text in the provided files and import them back to Guided Learning with the new language.

Export

1. Click **Translation** on the left panel of the OGL Console
2. Click **Export** (make sure the "properties" option is selected)

Guided Learning items will be saved as a ZIP file with all the exported content in the target folder (usually your "Download" folder).

Extract and save the information in the ZIP file to a folder of choice. If your operating systems does not support ZIP files you can use one of the following software providers: Winzip, Winrar, izip.



Your exported folder should include the following directories:

- **active** - This folder contains all published/active revisions of your guides; it should match what you see in your production environment. The folder will contain a list of directories, one for each language. In each such language folder you will find a list of files named according to your guides' api-names. The folder names '--' denotes the native language you used to create your guides originally; this will be English in most cases.

- **testing** - This folder contains all draft revisions of your guides; it should match what you see in your staging/test environment. The structure of this folder is identical to the active folder.
- **import - This folder has no real content. It is where you will need to place your translated guides before importing them back to Guided Learning. For more information about the import process please read the section below.**
- **README.txt** - this is the file you are reading now :)

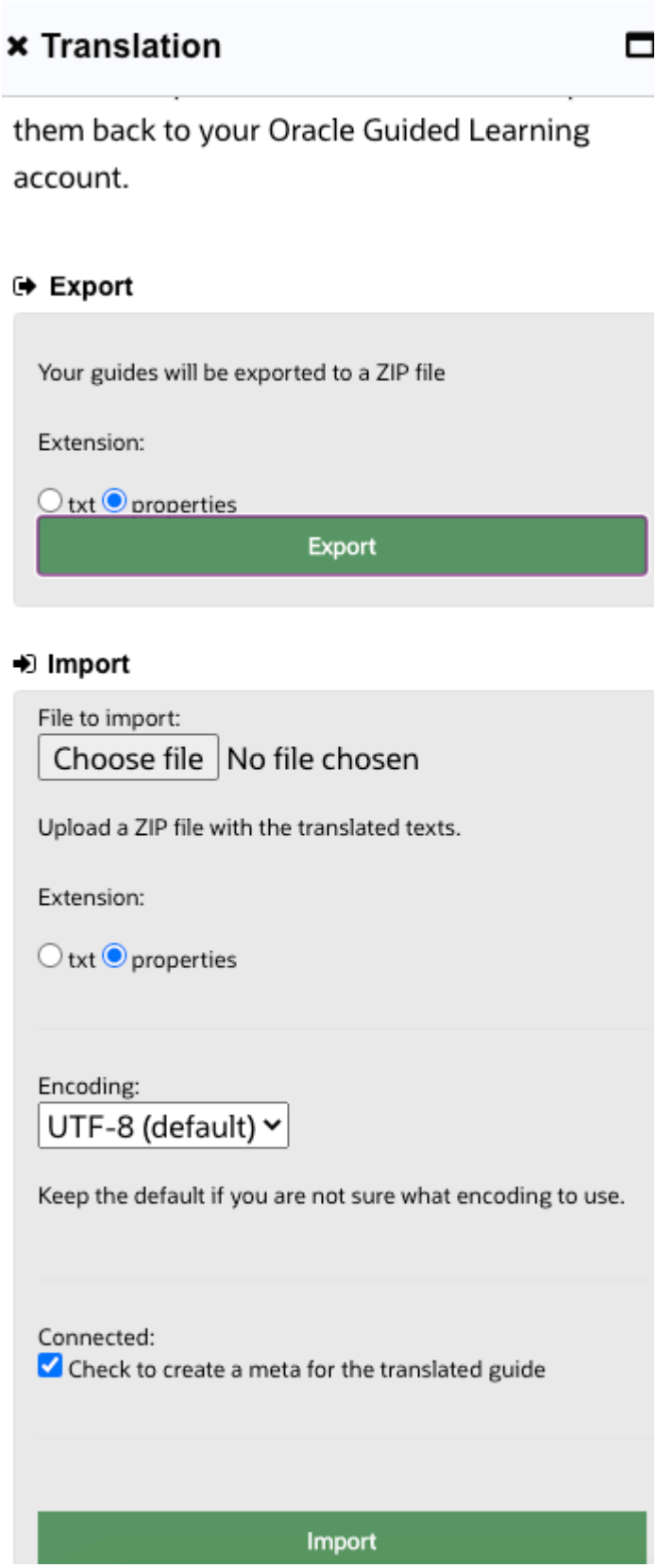
Import

In order to import the translated guides back to Guided Learning you will need to populate the **import** folder with the translated content.

The **import** folder should include a folder for each language you support. The name of each folder signifies the language it contains and should follow this naming convention. For your convenience, during the export process we have generated in the **import** folder a folder for each language you currently have guides for in the Guided Learning servers. You can add more language folders if needed.

To import the translated content you will need to ZIP the contents of this language folder only and upload it using the “Import” button in the Translation section on the left panel of your OGL Console.

- Click Translation on the left panel of your OGL Console
- Choose the file to import; this is the zipped language folder in your import folder
- Click Import (make sure Connected is checked)


Translation
them back to your Oracle Guided Learning account.
Export
Your guides will be exported to a ZIP file
Extension:
txt properties
Export
Import
File to import:
Choose file No file chosen
Upload a ZIP file with the translated texts.
Extension:
txt properties
Encoding:
UTF-8 (default)
Keep the default if you are not sure what encoding to use.
Connected:
Check to create a meta for the translated guide
Import

Foreign Language not Showing in Help Panel

So you've created your connected language folders and have content in them that you can see in the Console, you've updated the application.properties file and translated the guide property files, but still the Help Panel shows the same content that you see for the default language.

Did you update the Embedded Code?

If not, follow the steps listed below:

Instructions

```
JavaScript:
Add the following line to the Advanced Script:
iridize.lang = document.documentElement.lang == "en" ? "--" :
document.documentElement.lang.toLowerCase();In JS

Add the following line to the Config.js:
iridize.lang = document.documentElement.lang == 'en' ? '--' :
document.documentElement.lang.toLowerCase();

Fusion Embed:
Add the following line to the Custom Script:
iridize.lang = document.documentElement.lang == "en" ? "--" :
document.documentElement.lang.toLowerCase();
```

Native Fusion Integration for Language

If you have integrated via the native Fusion method, here is how to get additional languages working in your appId

Instructions

On the Configure Guided Learning page in Fusion

1. In the Customer Script field in the Advanced Settings section
2. Enter:

```
iridize.supported_langs = ['--', 'ja','es'];

iridize.lang = document.documentElement.lang == 'en' ? '--' :
document.documentElement.lang;

iridize.lang = iridize.supported_langs.indexOf(iridize.lang) >= 0 ? iridize.lang :
'--';
```

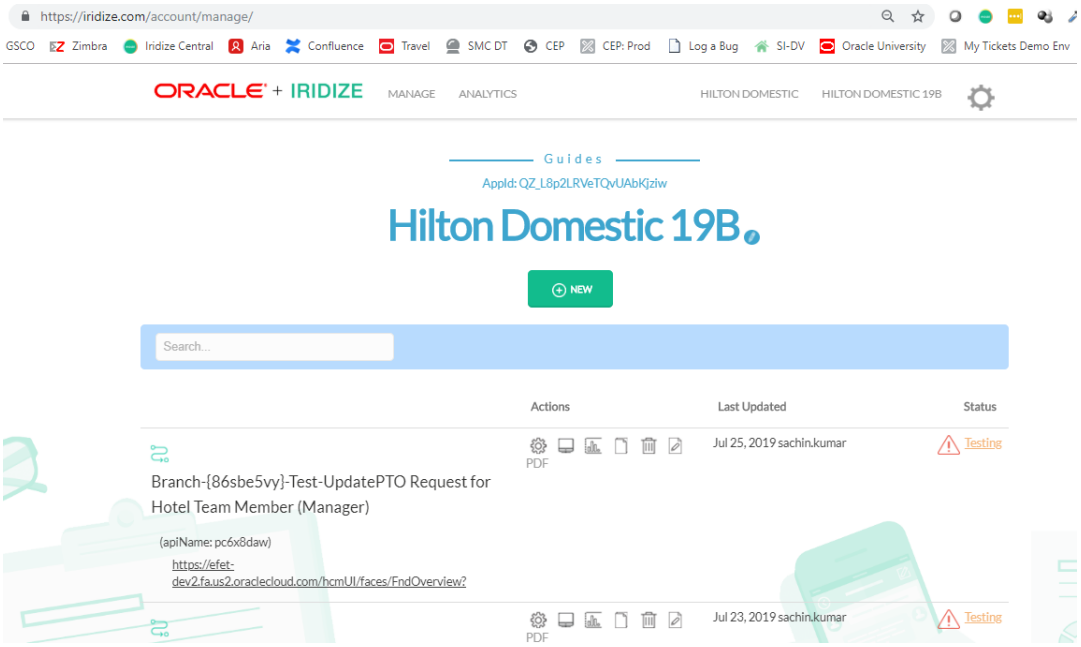
3. Replacing ja and es with whatever languages you want using the two letter abbreviation. Here is a link to the language abbreviation naming convention

Note: make sure you add your required languages in the 1st line of code, '-- ' always needs to be there.

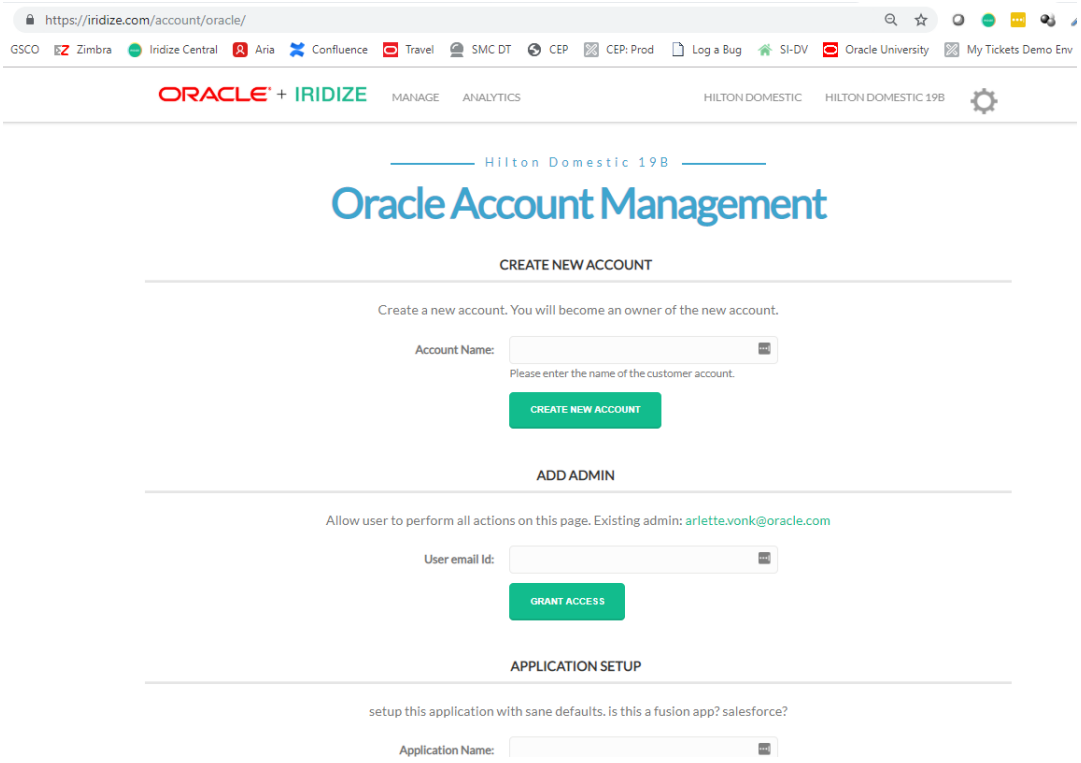
Remove a Language from an appID

Sometimes an account will scale back on their guided learning implementation and no longer need a certain language that had been added to an appId. There is an easy way to do this now.

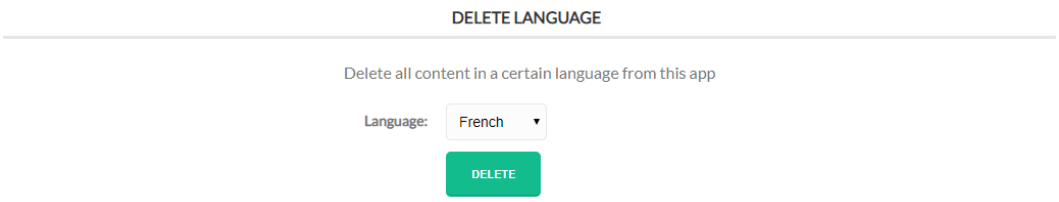
1. Go to the OGL Management Console for the appId you are working on. See below for example:



2. Change the url by replacing "#" with "oracle". See below for example.
(**NOTE: Special permissions must be set up for your account on the back end to complete this task**)



3. Scroll down to the Delete Language section:



4. Choose the language you wish to remove and select Delete.

Switching Language at Runtime

For the language change to take effect you should tell Guided Learning to refresh itself.

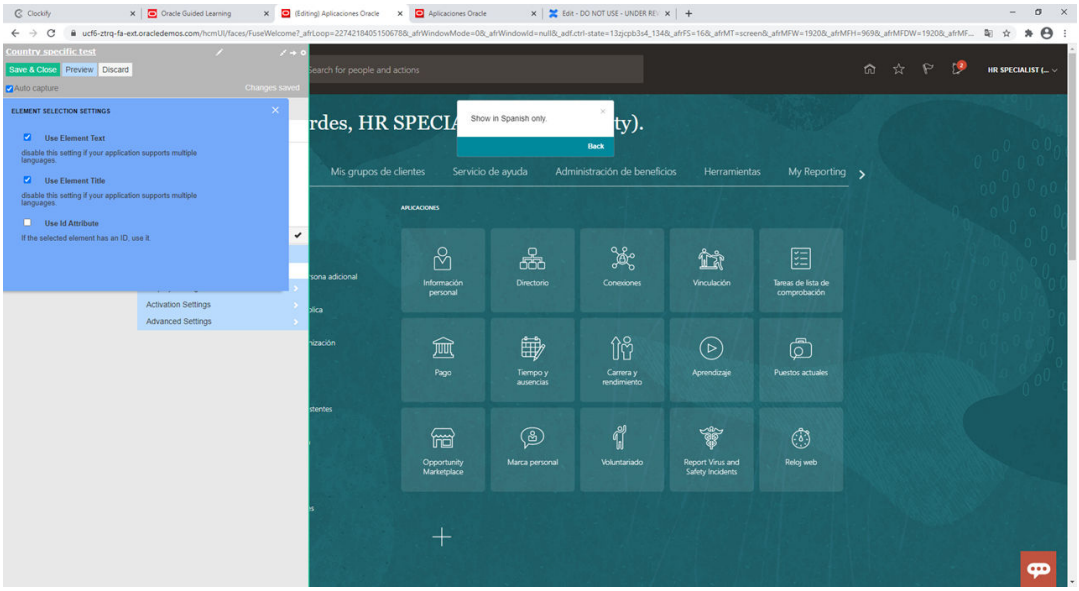
So after setting *iridize.lang* you will need to call the following API function:

```
Iridize('api.guide.refresh',{})
```

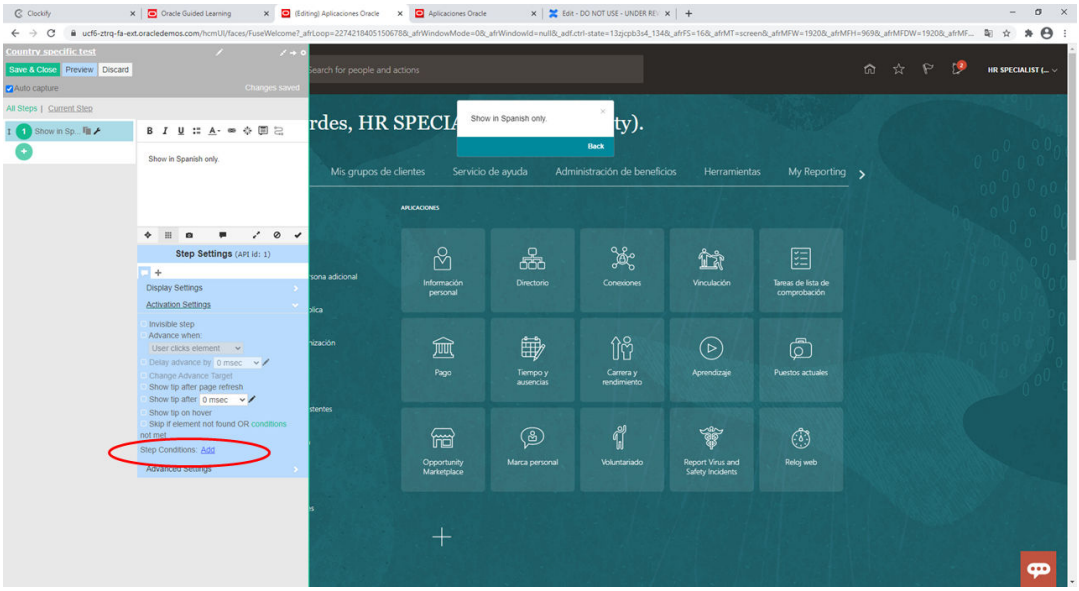
Add a condition so a guide only appears in a certain Fusion language

If a guide is pertinent to only one language, follow these steps so it is active only where desired:

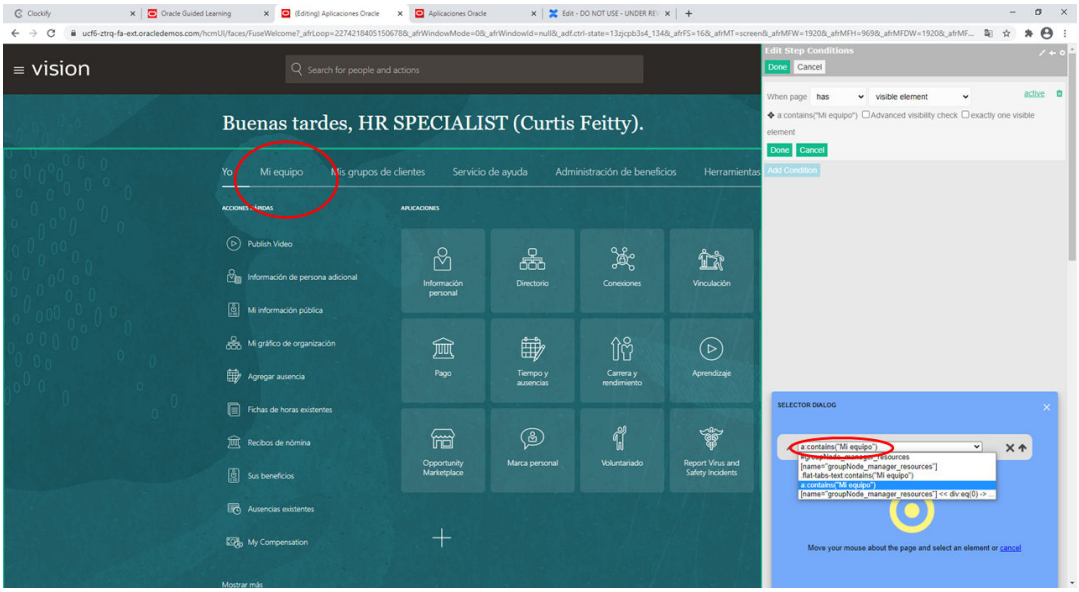
1. Create your guide normally
2. Open Fusion in the language where you wish the message/guide/link to appear
3. Launch the guide from the OGL Editor and select '**Use Element Text**' under ELEMENT SELECTION SETTINGS



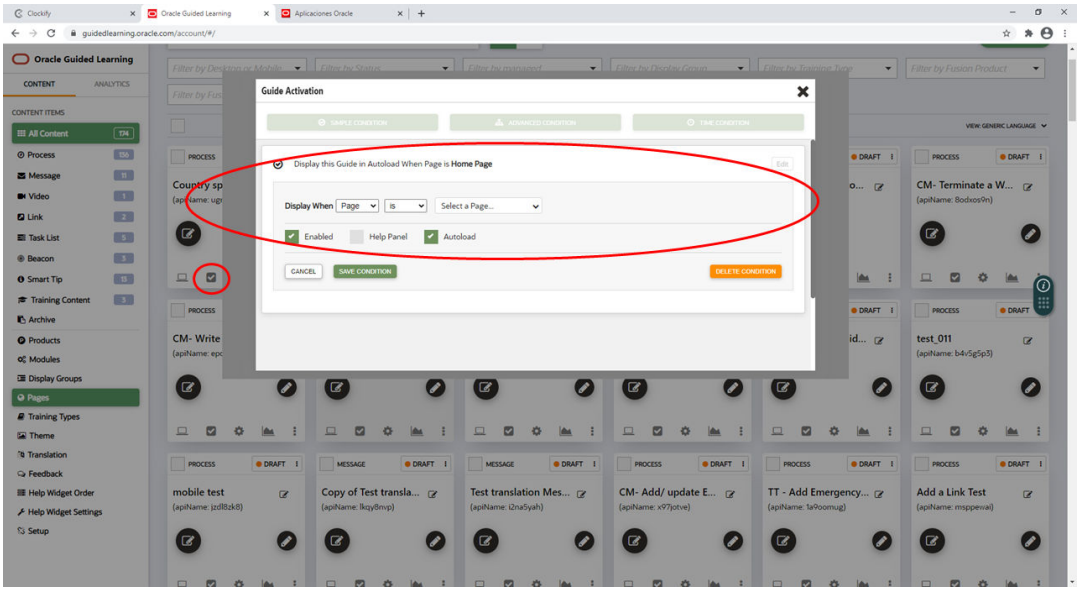
4. In Activation settings, select '**Add**' to add a step condition



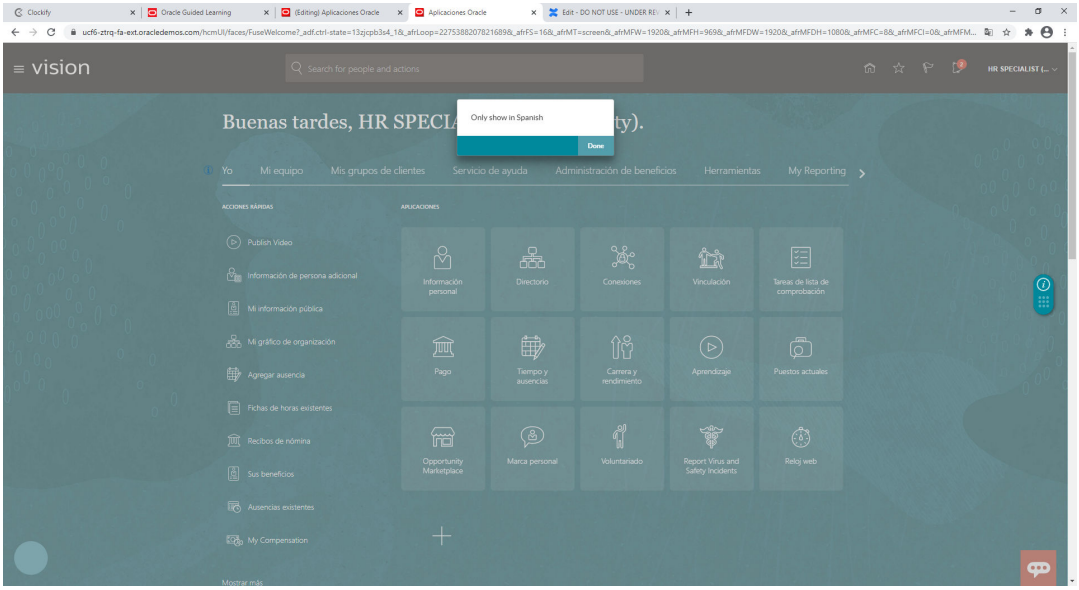
5. Select an element in the language where the guide should appear.
Note: This must be specific to this language only. The example below is 'Mi equipo' (My team).



6. Open guide activation settings in your guide and add a '**page**' condition. This example uses the HomePage.



7. Your guide will now only show in the required language, on the required page (in our example the Home Page) when the 'Mi equipo' element is visible



How to Request Machine Translation

Machine Translation may be requested up to 4 times per year using the following process. Review and validation is your responsibility.

1. Request desired language(s) by contacting your OGL Project Manager or by raising a support ticket. Refer to languages available below. Oracle will run Machine Translation, import the languages into the OGL console, and notify you of completion.
2. Validate the guides are functional in the foreign language and update both the guides and the translation(s) as necessary

- Quality of translation estimated **accuracy is 80%**

- Estimation of effort to validate and update: **15-30 minutes per guide per language**

Languages available

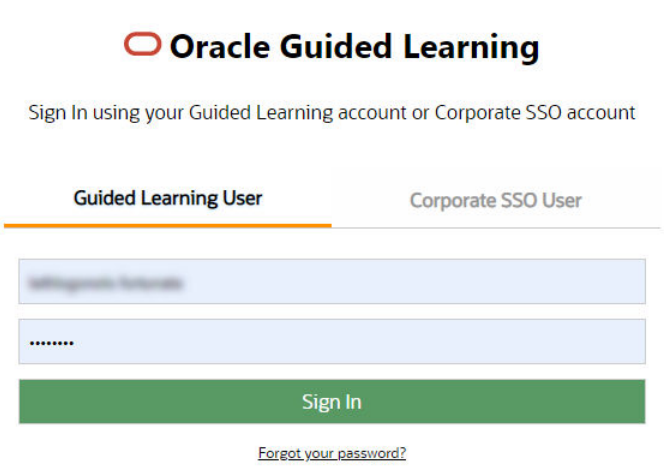
- Arabic
- Dutch
- French – Canada
- French - France
- German
- Italian
- Japanese
- Korean
- Polish
- Portuguese - Brazil
- Romanian
- Spanish - Worldwide
- Simplified Chinese
- Traditional Chinese
- Swedish
- Russian

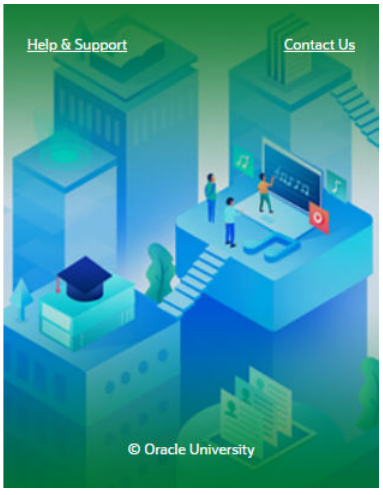
Hotspot Creation for Tracking Desktop Views and Mobile Views

Use Hotspots to gain insight into how your user population is accessing the application. Are users accessing the application via mobile devices or desktop devices?

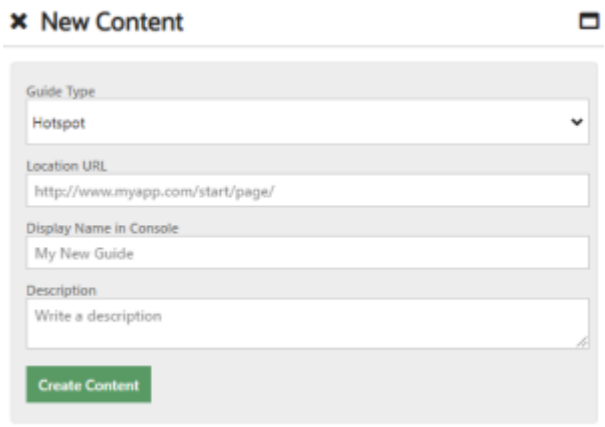
Steps

1. Open **Google Chrome**
2. Enter the **OGL Console URL**
3. Sign in using your username and password



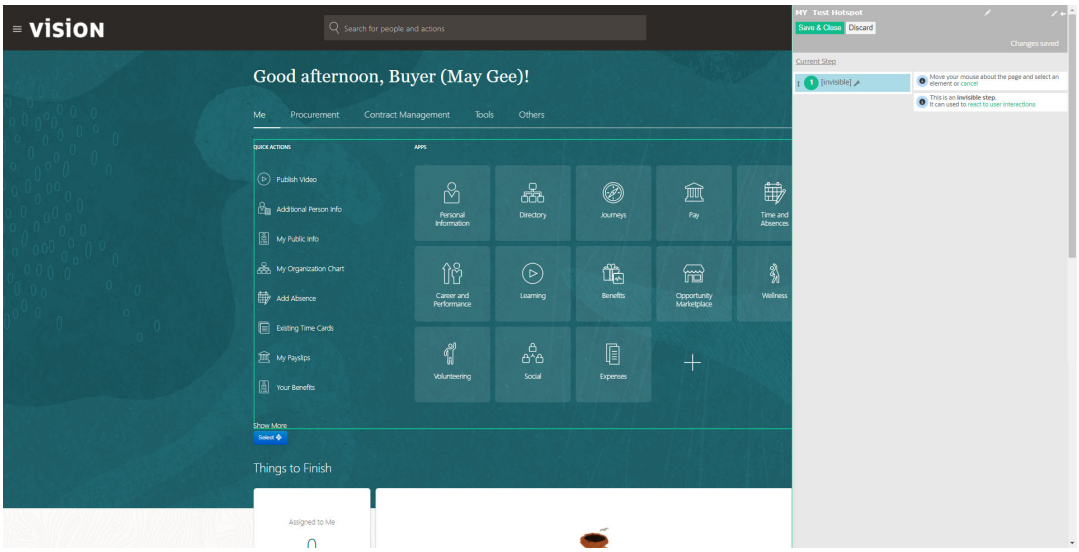


4. In a new tab, enter the **Application URL** and sign in to the application
5. Go to the OGL console, then select the **New Content** button.
6. Select **Hotspot** from the **Guide Type** dropdown



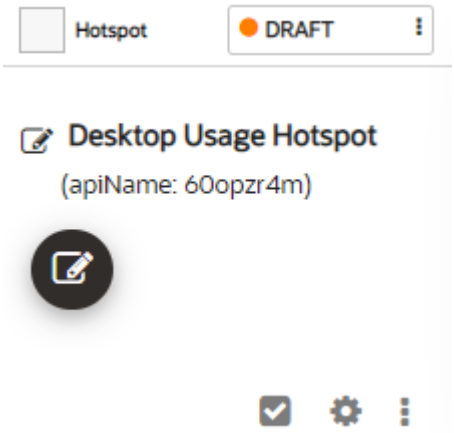
7. Enter the **Application URL** in the **Location URL** field
8. Enter **Desktop Usage Hotspot** in the **Display Name in Widget.** Replace XX with your initials.
9. Enter "This Hotspot monitors the Desktop usage" or similar in the **Description** field
10. Select **Create Content**.
Note: The OGL editor will open in a new tab.

11. Select any element on the page using **right-click**



12. Select the **Step Settings** icon
13. Select **Advanced Settings**
14. Select the **Target Selector** icon
15. Select the **Edit Selector** icon
16. Enter **body** in the selector field

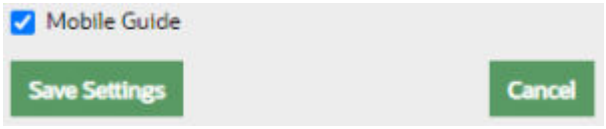
17. Select the **Check** icon to save
18. Verify that your element selector is **body**, listed below/next to the **Target Selector** icon
19. Select **Activation Settings**
20. Verify that the **Advance When** setting is checked and **User clicks element** has been selected from the dropdown
21. Select **Save and Close** to save your work and close the editor. Your item will be available in the console.



22. Select the **Activation** icon
23. On the **Guide Activation** window, select **Advanced Condition**
24. Set the condition to **Display when Page has URL matching /HomePage**

25. Select the **Autoload** checkbox, then select **Save Condition**
26. Select the **More** icon, then select **Clone**

27. Rename the cloned Hotspot by clicking the **Edit Guide Name** icon, then rename the Hotspot to **Mobile Usage Hotspot**
28. Select the **Save** icon to save the name changes
29. Select the **Settings** icon for the Mobile Usage Hotspot
30. Update the **Description** to "This Hotspot monitors the Mobile usage" or similar
31. Select the **Mobile** setting checkbox, then select **Save Settings**

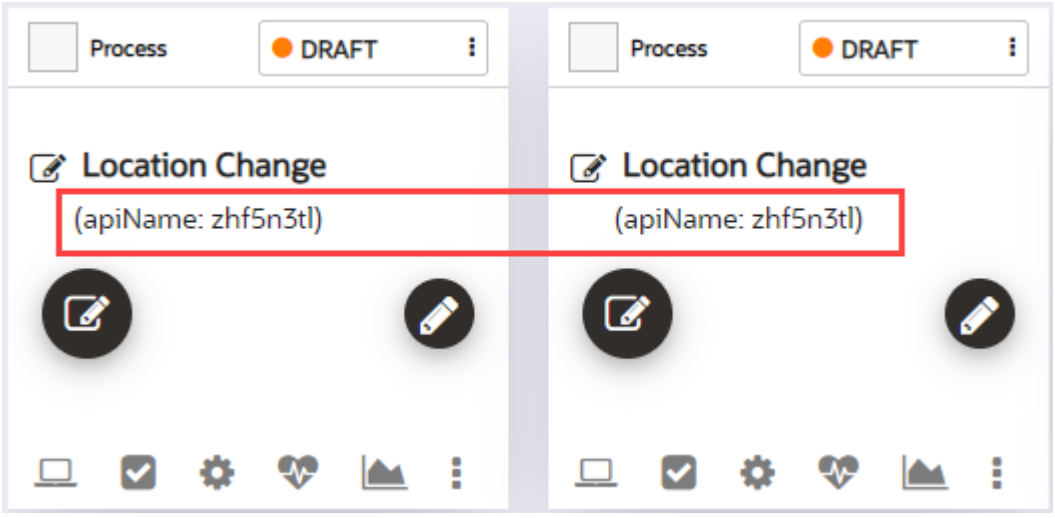


32. Publish both Hotspots

Moving Guides into an AppID with Existing Identical Guide API Names

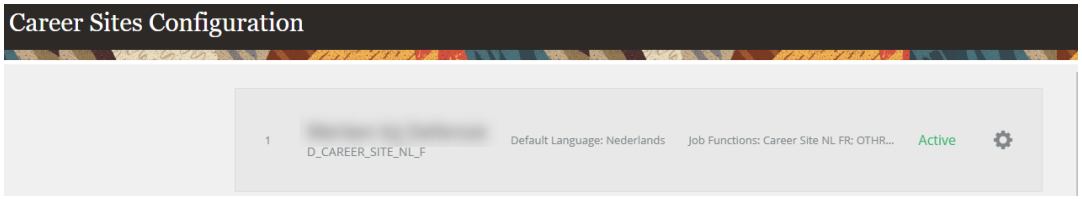
Where a set of guides are migrated/copied/moved into an ApplicationID/Account containing guides with identical guide APINames, there will be duplicate APINames in the ApplicationID/Account. The guide(s) being migrated from the source AppID will NOT overwrite the existing guides (ApiName(s)) in the destination AppID.

In this case, only one of each ApiName should be active in the AppID. Duplicates should be inactive or deleted.

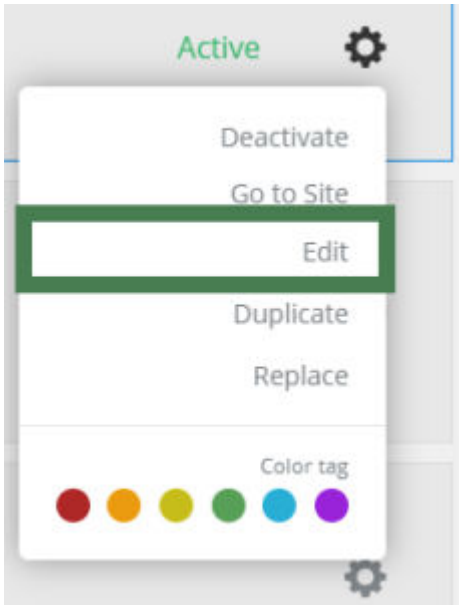


How to Enable OGL on Career Sites

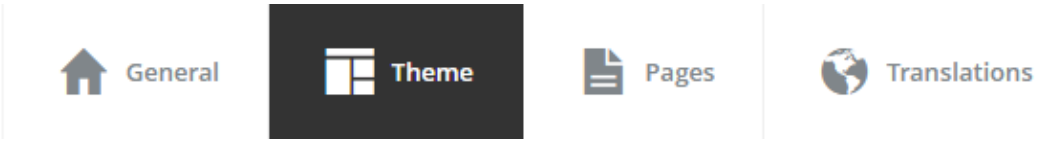
1. Open the **Career Sites Configuration** task in **Setup and Maintenance**



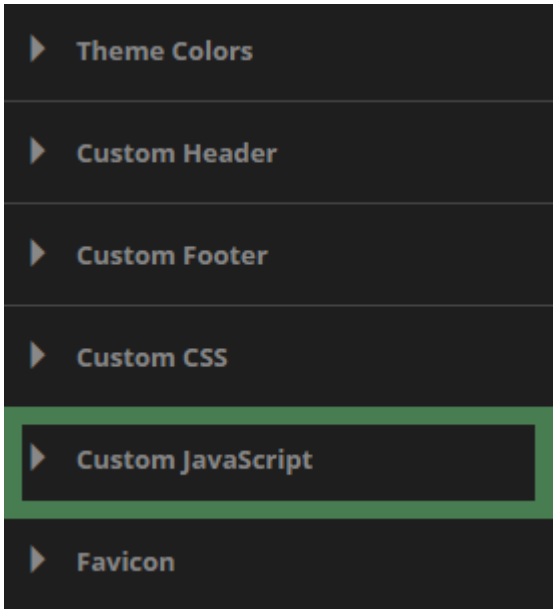
2. Select  for the target **Career Site**, then select **Edit**



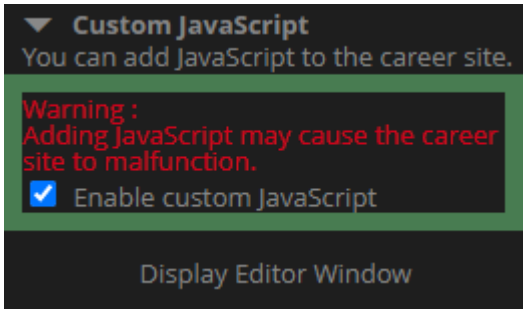
3. Select the **Theme** tab for the Career Site



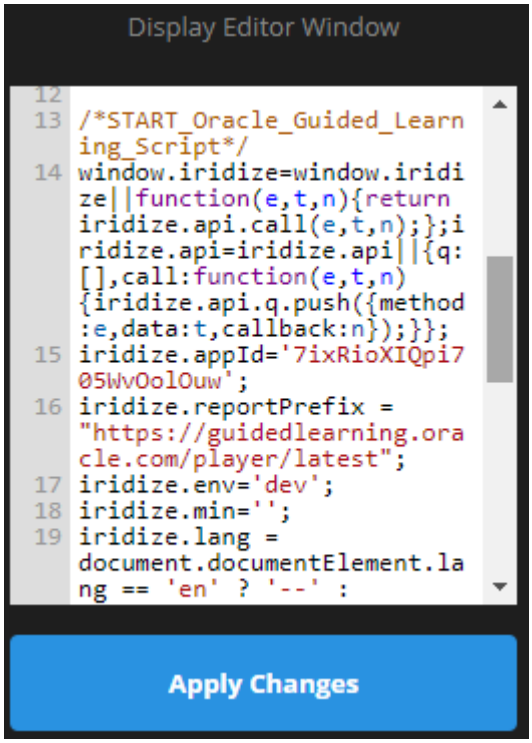
4. Locate the **Custom JavaScript** section and select to expand menu



5. Check the **Enable custom JavaScript** checkbox

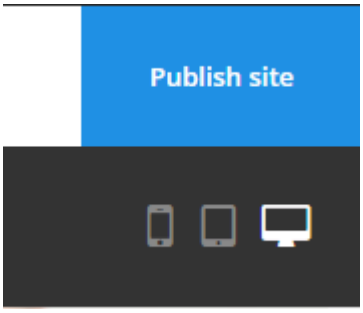


6. Paste the provided OGL JavaScript in the **Display Editor Window** field, then select **Apply Changes**.
IMPORTANT: Paste as Plain Text.



7. Verify that the **OGL Help Widget** is visible

8. Select **Publish site**



Content Security Policy (CSP) Error Correction

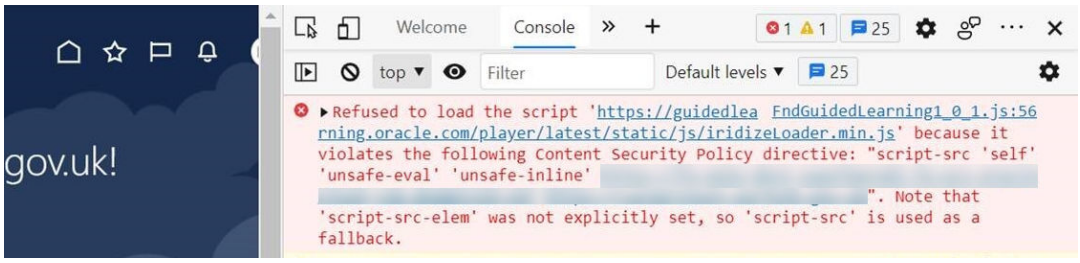
What is a CSP Error?

In simple terms, the OGL Help Widget will not load/display in the application due to a security policy.

Instead of blindly trusting everything that a server delivers, CSP defines the Content-Security-Policy HTTP header, which allows you to create an allowlist of trusted content sources and instructs the browser to only execute or render resources from those sources. The web's security model is rooted in the same-origin policy. For example, code from https://mybank.com should only have access to https://mybank.com's data, and https://evil.example.com should certainly never be allowed access. With this policy defined, the browser throws an error instead of loading scripts from outside sources *(Source:* Content Security Policy*)*. This includes data from Oracle Guided Learning.

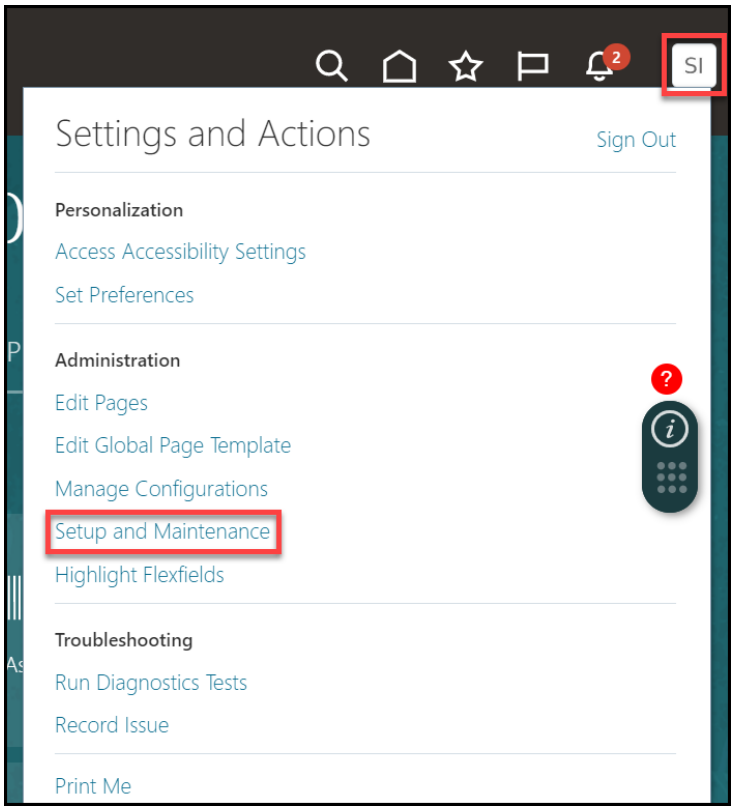
How to Troubleshoot

1. Right-click (Control-click on a **Mac**) on the application **Homepage**, then select **Inspect** from the menu.
2. Switch to the **Console** tab. Note the error in the Console relating to the Content Security Policy. It will read "*Refused to load the script...*" followed by the OGL URLs.



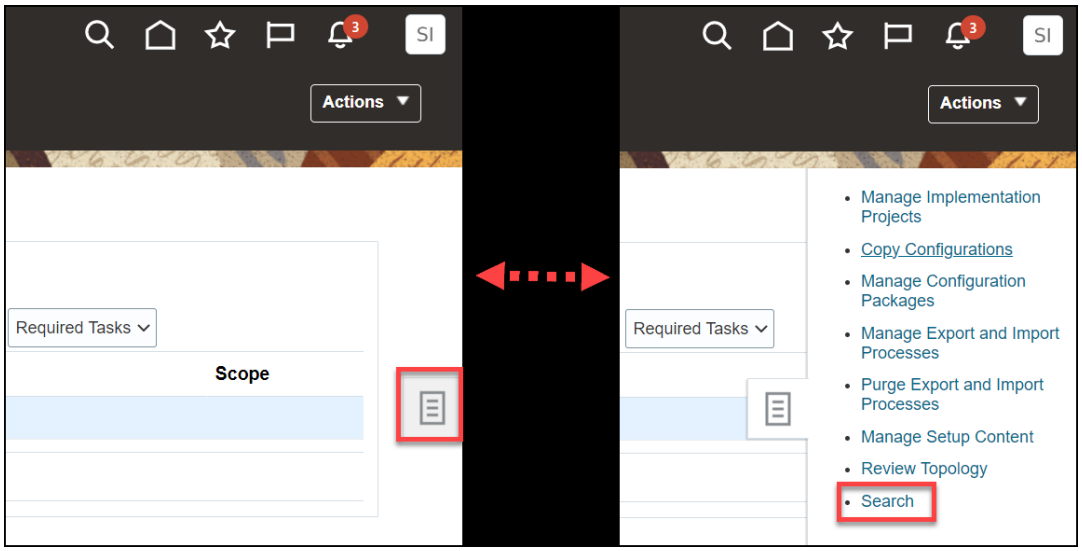
How to Correct a CSP Error

1. On the Fusion application **Homepage**, select the **Settings and Actions** icon (SI) and then select **Setup and Maintenance**.

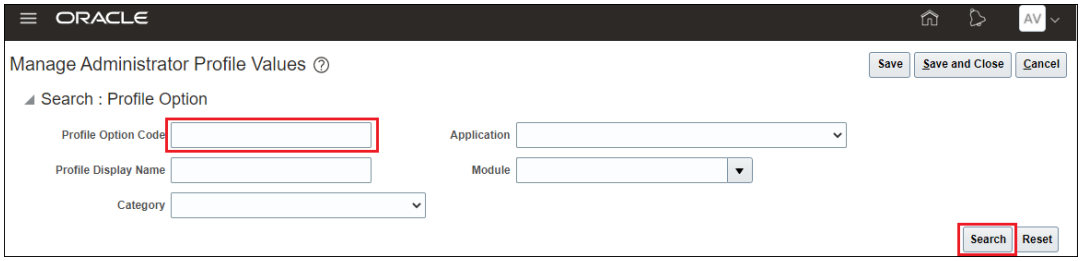


Alternatively, select the **Navigator** icon ( ) → **Others** → **Setup and Maintenance.**

2. Select the **Tasks** icon ( ) on the **Setup and Maintenance** screen to view the slide menu. Then select **Search** from the options in the slide menu.



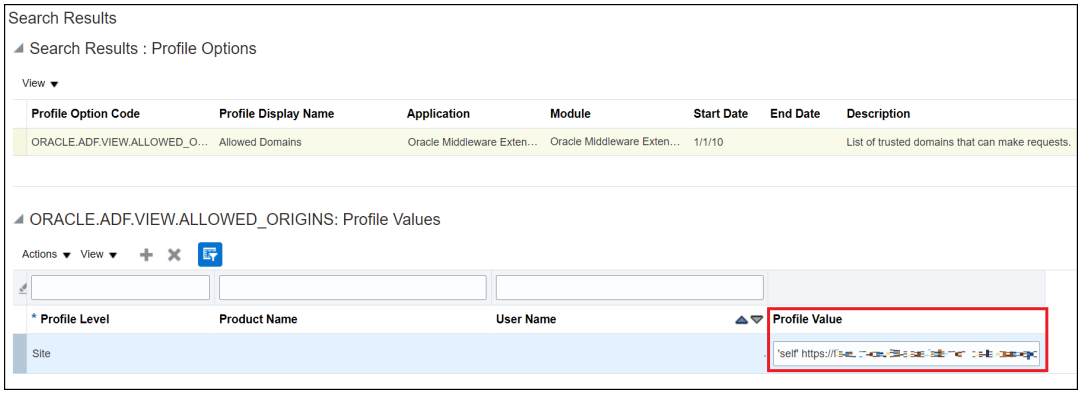
3. In the search field, enter *"**Manage Administrator Profile Values**"* and then select the **Search** icon ( ).
4. From the search result, select **Manage Administrator Profile Values**.
   A new window opens now, where you can manage **Administrator Profile Values.**
5. Enter **ORACLE.ADF.VIEW.ALLOWED_ORIGINS** in the **Profile Option Code** field and then select **Search**.



> **Note:**
>
> More information on this profile option can be found on Cross-Origin Resource Sharing.

6. Scroll down to view the **Profile Value** results displayed.

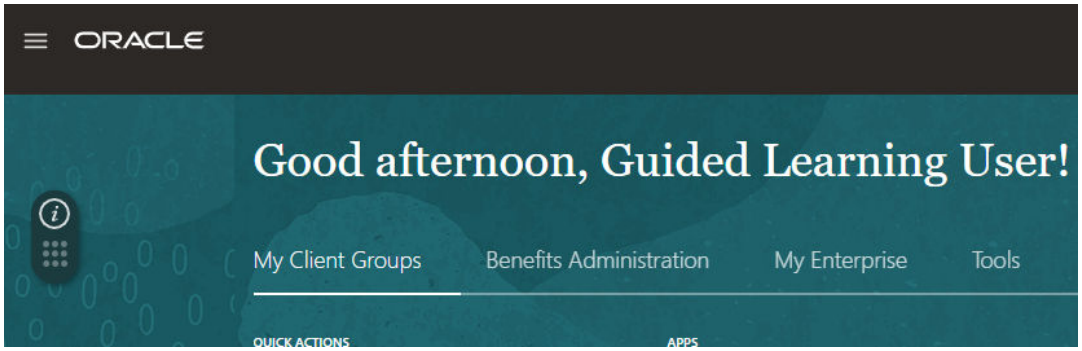
The field value may appear as "**'self'**" or "**'self' https://xyz.com**" (with your organization-specific domain).



7. Depending on your data centre region, enter the **Profile Value** as provided below:
   - **NA Tenancy: 'self' https://guidedlearning.oracle.com**
   - **EMEA Tenancy: 'self' https://guidedlearning.oracle.com https://guidedlearning-emea.oracle.com**
   - **APAC Tenancy: 'self' https://guidedlearning.oracle.com https://guidedlearning-apac.oracle.com**

> **Note:**
>
> - You may find the field value appears as **'self' <third party URL>** or 'self' with your organization-specific URLs. Do **NOT** overwrite any URLs that are already defined, add the OGL URLs to the existing URLs.
> - Use the whitespace separator (single space) between the URLs (i.e. **'self'<white_space><url_1><white_space><url_2><white_space><url_3>**).
> - You can find more details on this profile value in this Oracle article.
> - This applies for Fusion version 21.04.0 or later.

8. Select **Save and Close** to save your changes and exit the **Manage Administrator Profile Values** task.
9. Log out of the application and log in again.
10. Go to the **Homepage** to verify that the **OGL Help Widget** is visible.



You have now successfully resolved the CSP error.

CSP Error in Non-Fusion Applications
If you experience a CSP error in your non-Fusion application, contact your IT support team to ensure that OGL-related servers and services are whitelisted.

Guide Activation Condition Troubleshooting

When Page Has Element / Visible Element

Depending on the version of JavaScript used to enable OGL on the application, activation conditions may not work as expected. This affects the *'when page has/has not element'* and *'when page has/ has not visible element'* conditions.

Review the OGL JavaScript and verify if it contains the following line:

```
window.addEventListener('load', function(){var e=document.createElement("script");var
t=document.body;e.src=("https:"==document.location.protocol?"https:":"http:")+"//
guidedlearning.oracle.com/player/latest/static/js/iridizeLoader.min.js";e.type="text/
javascript";e.async=true;t.appendChild(e); });
```

Why does this happen?

The **window.addEventListener** line ensures that OGL loads after the application page has loaded. Therefore, it is not possible for OGL to determine whether or not the element is on the page.

Why should I do next?

DO NOT use the ***When Page Has Element/Visible Element*** conditions. Doing so will create inconsistent user experiences. Instead, set alternative conditions which suit the requirement.

11

Troubleshooting

This chapter describes problems that you may encounter when using Oracle Guided Learning and provides you with information about how to diagnose and overcome such problems.

Content Security Policy (CSP) Error Correction

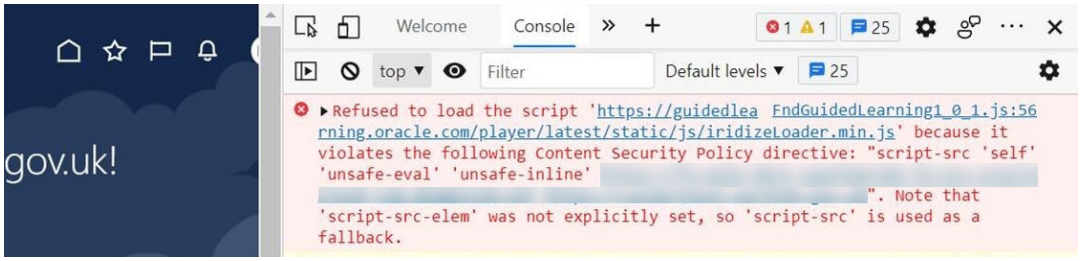
What is a CSP Error?

In simple terms, the OGL Help Widget will not load/display in the application due to a security policy.

Instead of blindly trusting everything that a server delivers, CSP defines the Content-Security-Policy HTTP header, which allows you to create an allowlist of trusted content sources and instructs the browser to only execute or render resources from those sources. The web's security model is rooted in the same-origin policy. For example, code from https://mybank.com should only have access to https://mybank.com's data, and https://evil.example.com should certainly never be allowed access. With this policy defined, the browser throws an error instead of loading scripts from outside sources *(Source:* Content Security Policy*)*. This includes data from Oracle Guided Learning.

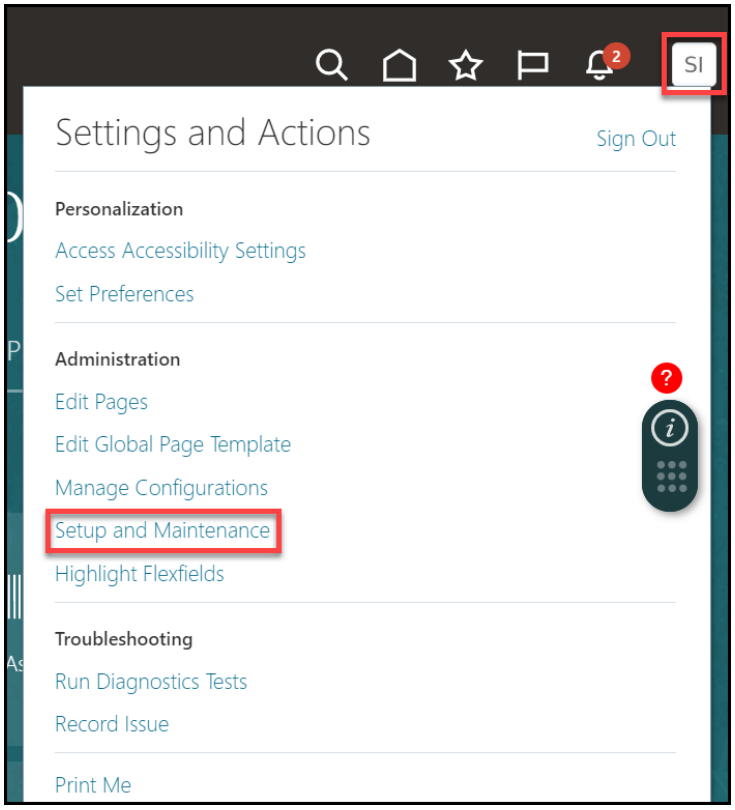
How to Troubleshoot

1. Right-click (Control-click on a **Mac**) on the application **Homepage**, then select **Inspect** from the menu.
2. Switch to the **Console** tab. Note the error in the Console relating to the Content Security Policy. It will read "*Refused to load the script...*" followed by the OGL URLs.



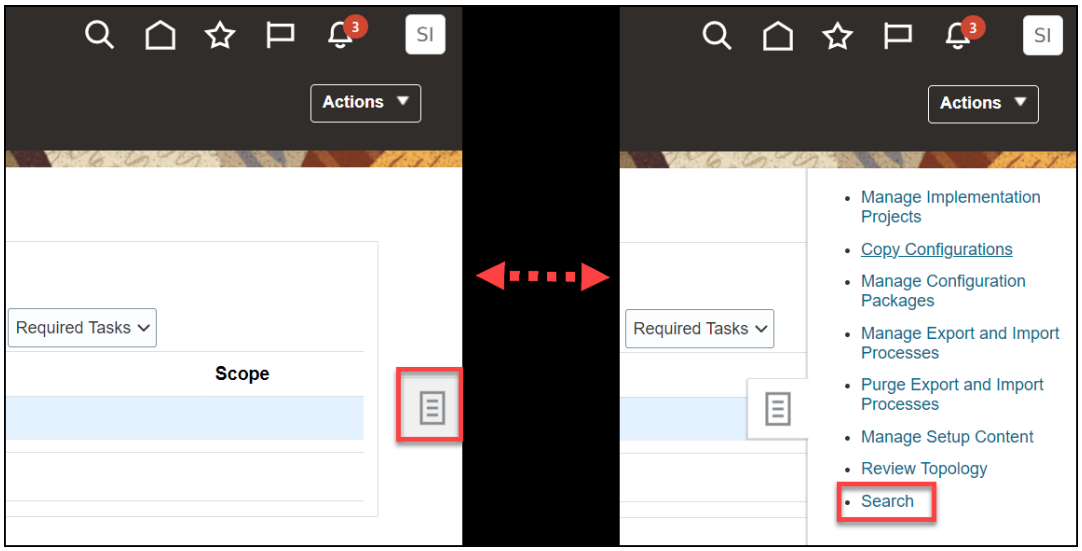
How to Correct a CSP Error

1. On the Fusion application **Homepage**, select the **Settings and Actions** icon ( ) and then select **Setup and Maintenance**.

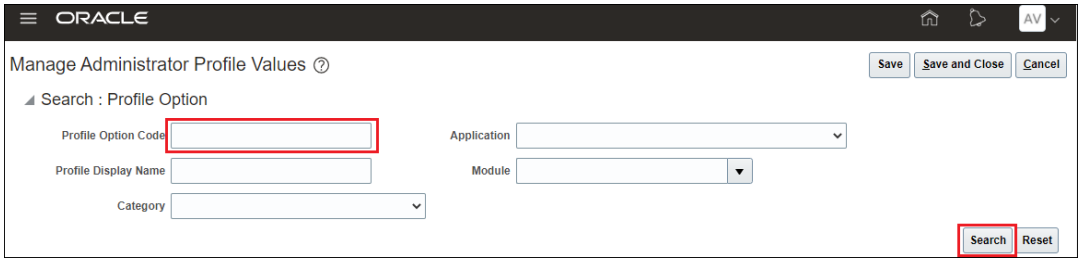


Alternatively, select the **Navigator** icon ( ) → **Others** → **Setup and Maintenance.**

2. Select the **Tasks** icon ( ) on the **Setup and Maintenance** screen to view the slide menu. Then select **Search** from the options in the slide menu.

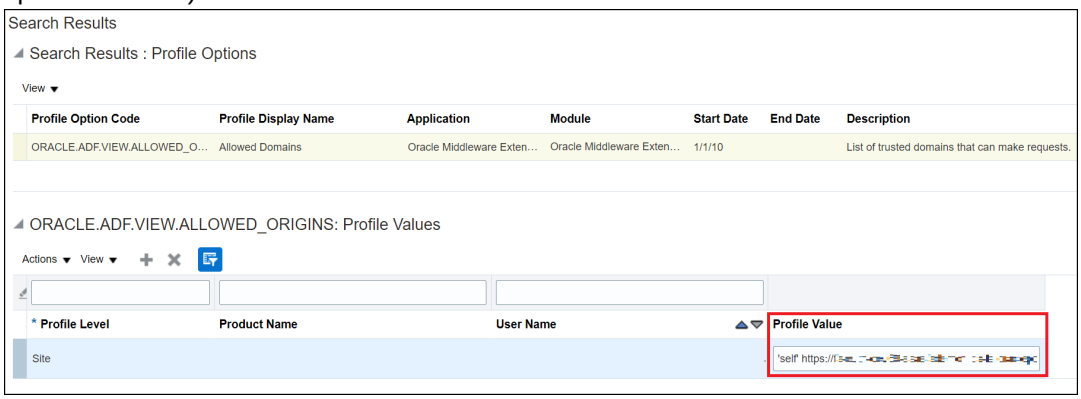


3. In the search field, enter *"**Manage Administrator Profile Values**"* and then select the **Search** icon ( ).
4. From the search result, select **Manage Administrator Profile Values**.
   A new window opens now, where you can manage **Administrator Profile Values.**
5. Enter **ORACLE.ADF.VIEW.ALLOWED_ORIGINS** in the **Profile Option Code** field and then select **Search**.



> **Note:**
>
> More information on this profile option can be found on Cross-Origin Resource Sharing.

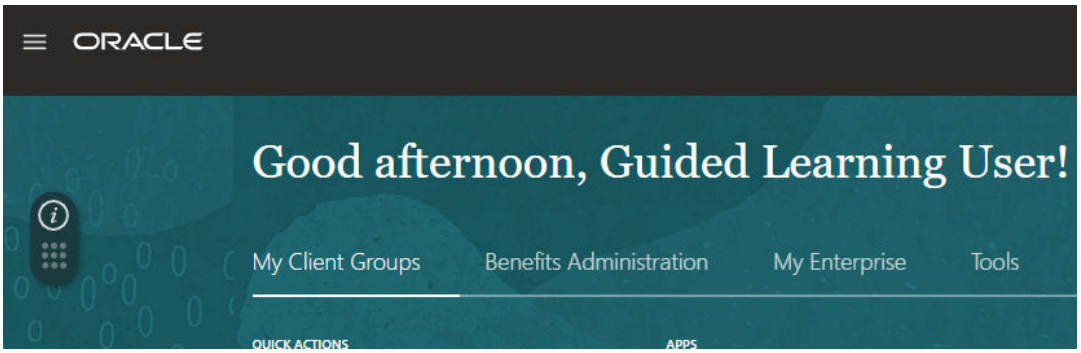
6. Scroll down to view the **Profile Value** results displayed.
The field value may appear as "**'self'**" or "**'self' https://xyz.com**" (with your organization-specific domain).



7. Depending on your data centre region, enter the **Profile Value** as provided below:
   - **NA Tenancy: 'self' https://guidedlearning.oracle.com**
   - **EMEA Tenancy: 'self' https://guidedlearning.oracle.com https://guidedlearning-emea.oracle.com**
   - **APAC Tenancy: 'self' https://guidedlearning.oracle.com https://guidedlearning-apac.oracle.com**

> **Note:**
>
> - You may find the field value appears as **'self' <third party URL>** or 'self' with your organization-specific URLs. Do **NOT** overwrite any URLs that are already defined, add the OGL URLs to the existing URLs.
> - Use the whitespace separator (single space) between the URLs (i.e. **'self'<white_space><url_1><white_space><url_2><white_space><url_3>**).
> - You can find more details on this profile value in this Oracle article.
> - This applies for Fusion version 21.04.0 or later.

8. Select **Save and Close** to save your changes and exit the **Manage Administrator Profile Values** task.
9. Log out of the application and log in again.
10. Go to the **Homepage** to verify that the **OGL Help Widget** is visible.



You have now successfully resolved the CSP error.

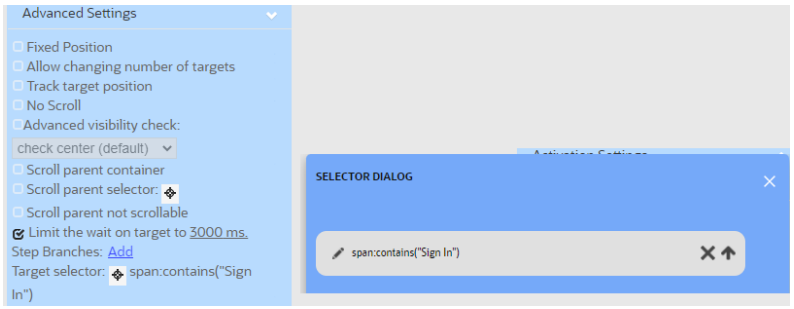
CSP Error in Non-Fusion Applications
If you experience a CSP error in your non-Fusion application, contact your IT support team to ensure that OGL-related servers and services are whitelisted.

OGL Slowing Down Host Machine

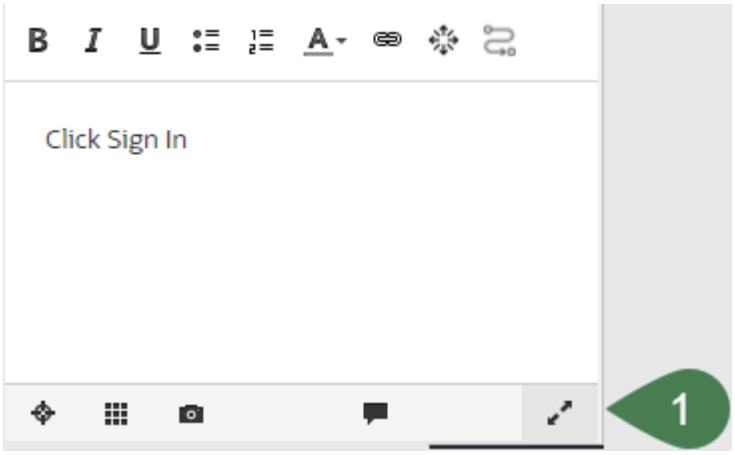
If you determine that OGL is slowing down the performance of the host machine, this is probably due to an OGL item containing a selector that matches the tooltip text.

A selected element could be **span:contains(**"Name"**)** and the guide tooltip could contain the word "Name" but wrapped in **span:contains("**Name**")**, this creates an infinite loop between an element that is in on the host application and the OGL tooltip.

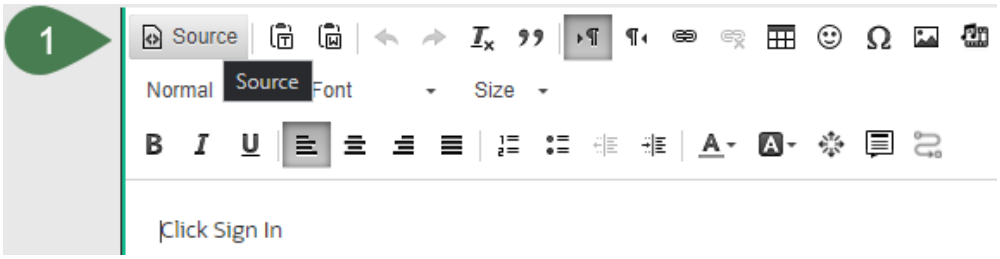
1. Open the suspect guide in the full editor and verify the selector under Advanced Settings or the Selector Dialog.



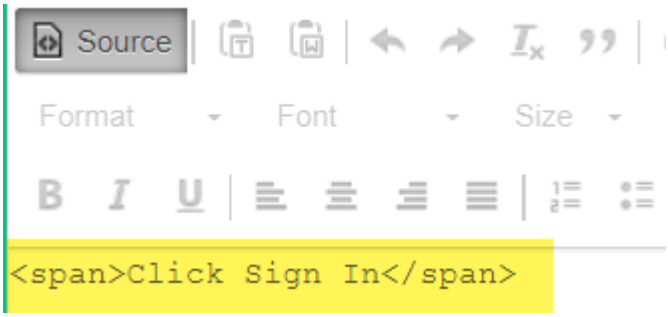
2. Expand the Text Editor.



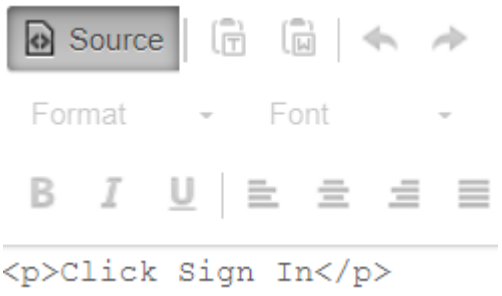
3. Select Source to view the source code.



4. Scan through the code to find the text wrapped in the **** tag and containing the matching text i.e. **Sign In**.



5. Next, we recommend you update this a tagother than , this can be <p>,<label>, or any other suitable HTML Element.



6. Once updated, you can save your changes and the issue should be resolved.

Tip Flickering

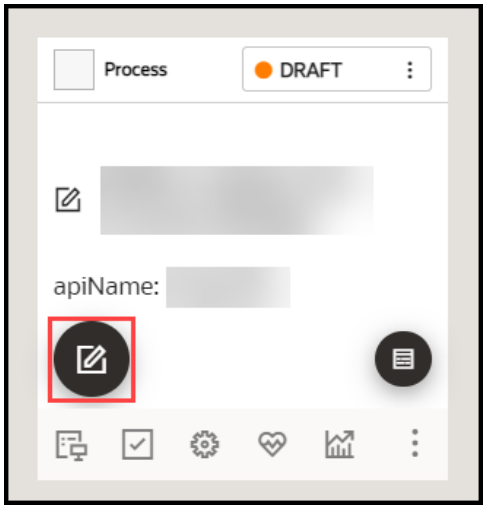
Flickering occurs when you land on a page, and the tip appears, disappears, and reappears, often rapidly and continuously.

How to Fix Flickering

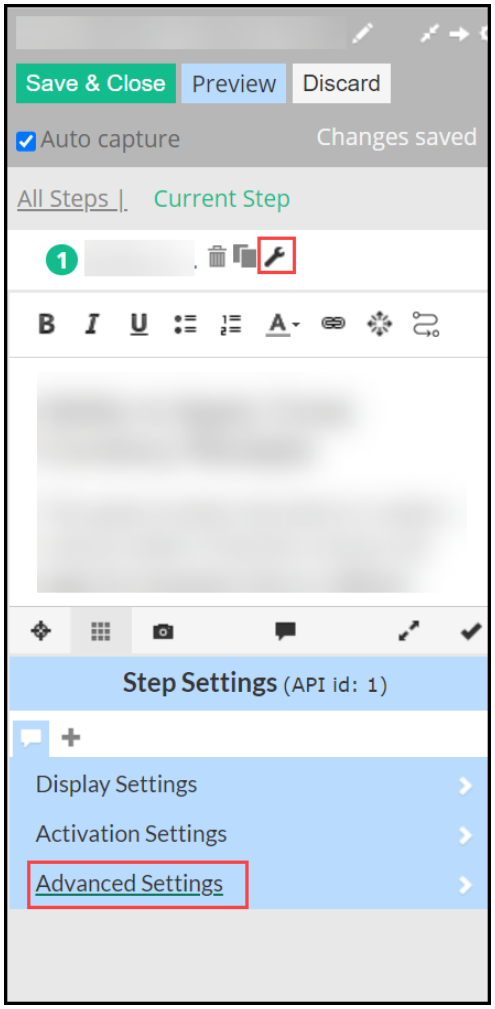
With just a few changes in the Advanced Settings of the tip, you can easily resolve any flickering issues.

To resolve the flickering of the tip:

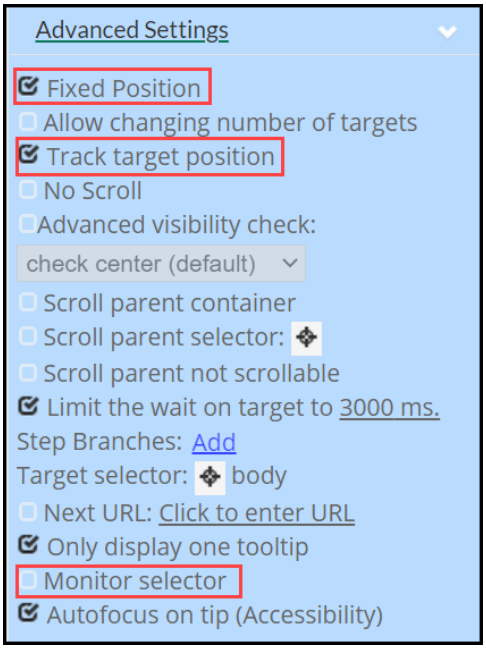
1. On the faulty Guide, select the **Editor** icon to open the OGL editor.



2. Locate the faulty step in the guide.
3. On the faulty step, go to **Step Settings > Advanced Settings**.



4. Under **Advanced Settings**,
   a. Turn on the **Fixed Position**.
   b. Turn on the **Track Target Position**.
   c. Turn off the **Monitor Selector**.



This combination of settings should resolve the flickering issue.

Tip:

If the issue persists, turn the **Monitor Selector** off and attempt the following steps:

Turn on the **Fixed Position**, turn off the **Track Target Position**, and switch them around if needed.