HCL MT CH-MSP Product Documentation Connections

Cloud Application Extension and Customization Migration

Edge Proxy considerations

Embedding the Navigation Bar

Running list of Issues (MT-MSP)

Cloud Application Extension and Customization Migration

Documentation and Examples

A public repository connections-msp for documentation and example extensions / customizations for Connections MSP environments has been defined on github.com.

In this repository, the following information is available to guide the process of migrating integrations and customizations:

- Extension and customization migration process documentation.
- Catalog app extensions for use with Connections MSP environments.
- Examples for extending the UI using Customizer.

Known Issues / Limitations

Admin Accounts

To be allowed access to create / update appregistry extensions via the appregistry UI or the appregistry services API, the user must be using an account mapped to the **orgadmin** role within the Common app. All appregistry extension items are scoped with the organization ID from the organization to which the account belongs. It is not possible for a Connections administrator account, mapped to the global admin role, to manage extensions on behalf of organizations because the account could not be related to the specific organization whose extensions are to be modified in order to maintain the scoping by organization ID.

There is no concept of a single appregistry extension definition that applies across all organizations; each organization has to create the extension in the context of their own organization.

Edge Proxy considerations

It is expected that there is some sort of reverse proxy server positioned in front of Connections which provides various services. This article will list items that should be implemented at the edge proxy.

1. Remove basic authentication headers

Basic authentication should be disabled for the Connections instance. WebSphere does not provide an option to do this easily, though there are several ways this would be accomplished. One suggestion is to remove an Basic authentication headers from incoming requests. This can be accomplished with a rule at the edge proxy that looks for the **Authentication** header with a value that starts with "**Basic** ". It is suggested to do a case insensitive comparison. If that header is found, simply remove it.

The following rule works on an NGINX server (though there may be other ways to implement a similar effect within NGINX. Please suggest if you find something better!)

```
map $http_authorization $authVal {
    default $http_authorization;
    ~*(^basic)  "";
}

proxy_set_header Authorization $authVal;
```

Note that if you are using basic authentication for the internal Connections provisioning APIs, then you should ensure that the provisioning apis are run directly against Connections or IHS that is in front of Connections. Presumably this endpoint is only available internally to the MSP, which is where the provisioning APIs are also executed. This avoids the header being stripped from these API requests.

Embedding the Navigation Bar

The following information shows how to embed and use the Connections navigation bar on other web pages and applications to assist users in navigating between applications.

This is an example where the Connections navbar has been added into another application page:

<Image Required>

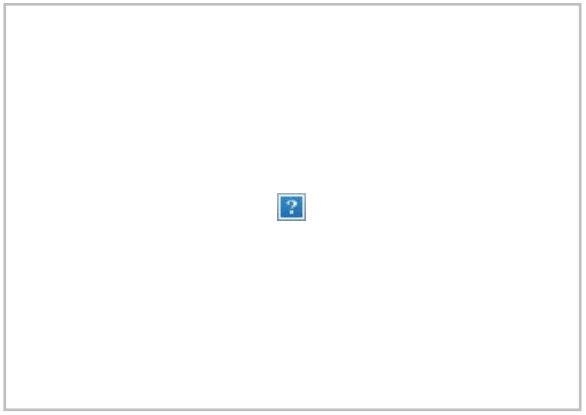
Setting Up the Required Files

The script and CSS files are common to all tenants and only need be set up for the first customer that wishes to make use of embedding the navigation bar.

1. Create a directory under the Customizer persistent volume **/pv-connections/customizations** to host the necessary shared files. Examples used here use the name **embed-navbar** but the directory name can be anything; all orgs wishing to make use of the feature will have to know what it is in order to reference it properly in their <script src> tag.

2. Copy the following files into the directory:

File Name	Purpose
embed-nav.css	Additional styling added to navbar
embed-nav.js	Additional JS added to navbar content
embed-nav.json	Customizer extension definition
get-navbar.js	Script to embed navbar in iFrame on web page
navbar.html	(Optional) Example / test page



The files can be found in this repo https://github.com/ibmcnxdev/connections-msp/tree/master/customizations/service_menu/embed-navbar; there are minified versions of the .js files with .min.js names. Adjust the extension json to use that file name if preferred.

Including the Navigation Bar

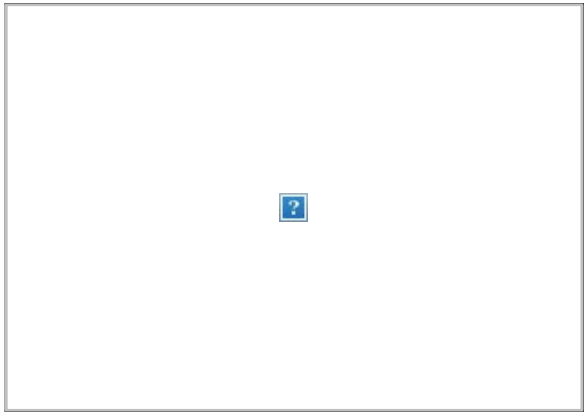
Creating the Customizer Extension

Connections produces just the re-usable header content via a request to **/homepage/web/pageHeader** but some additional styling and javascript is required and is injected by a Customizer extension. Because Customizer is processing this response like any other, any customizations applied to the navigation bar by Customizer will also appear in the embedded navigation bar.

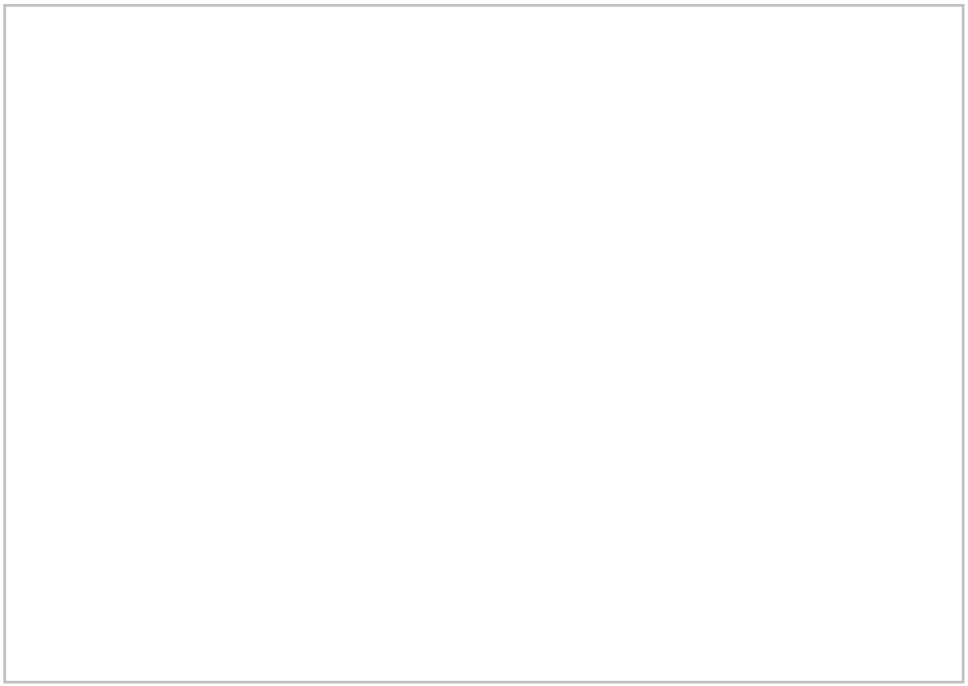
Each customer must set up their own Customizer extension in the app registry since they cannot be shared across organizations.

1. Using the app registry client **https://<vanityHost>/appreg/apps** create a new app, click on Code Editor link and remove the initial json outine, then paste in the json from the **embed-nav.json** example file (screenshot below).

Notice that url match rule which indicates that although this is on the /homepage path, it is only actioned by Customizer when the url contains pageHeader, so it will not affect any other /homepage/….requests.



2. When the request for **/homepage/web/pageHeader** is made, the response should contain src tags for the two included files and they should be seen to be loaded in the browser network trace:



Manually test this in the browser to make sure the two files are being loaded.

Adding the Script Tag

The last part of the puzzle is to insert a script source tag into the page(s) where the navigation bar is desired.

To do this, a tag like the following is placed in the html page referencing the **get-navbar.js** script along with some parameters that can be specified; some required and some optional.

```
<!-- Script source and supported parameters -->
<!-- anchorid       : (Required) element ID to attach iFrame -->
<!-- hostname       : (Required) vanity host name from where to fetch navbar header content -->
<!-- framewidth     : (Optional) width of navbar iframe container (default = 100%) -->
<!-- frameheight    : (Optional) height of navbar iframe container (default = 50px) -->
<!-- maxframeheight : (Optional) maximum height of navbar iframe container (default = 500px) -->
<!-- frameborder    : (Optional) border of navbar iframe container (default = 0) -->
<!-- framepadding   : (Optional) padding of navbar iframe container (default = 0) -->
<!-- framemargin    : (Optional) margin of navbar iframe container (default = 0) -->

<script id="embedCNXNavbar" src="https://mtdemo1-orgc.cnx.cwp.pnp-hcl.com/files/customizer/embed-navbar/get-navbar.js" anchorid="lotusContent"
hostname="mtdemo1-orgc.cnx.cwp.pnp-hcl.com"
frameWidth="100%" frameheight="50px" maxframeheight='500px' frameborder="0" framepadding="0" framemargin="0"></script>
```

The example above (and the provided navbar.html) contains comments that define the allowable parameters but only the **<script>. .. </script>** tag content is actually necessary.

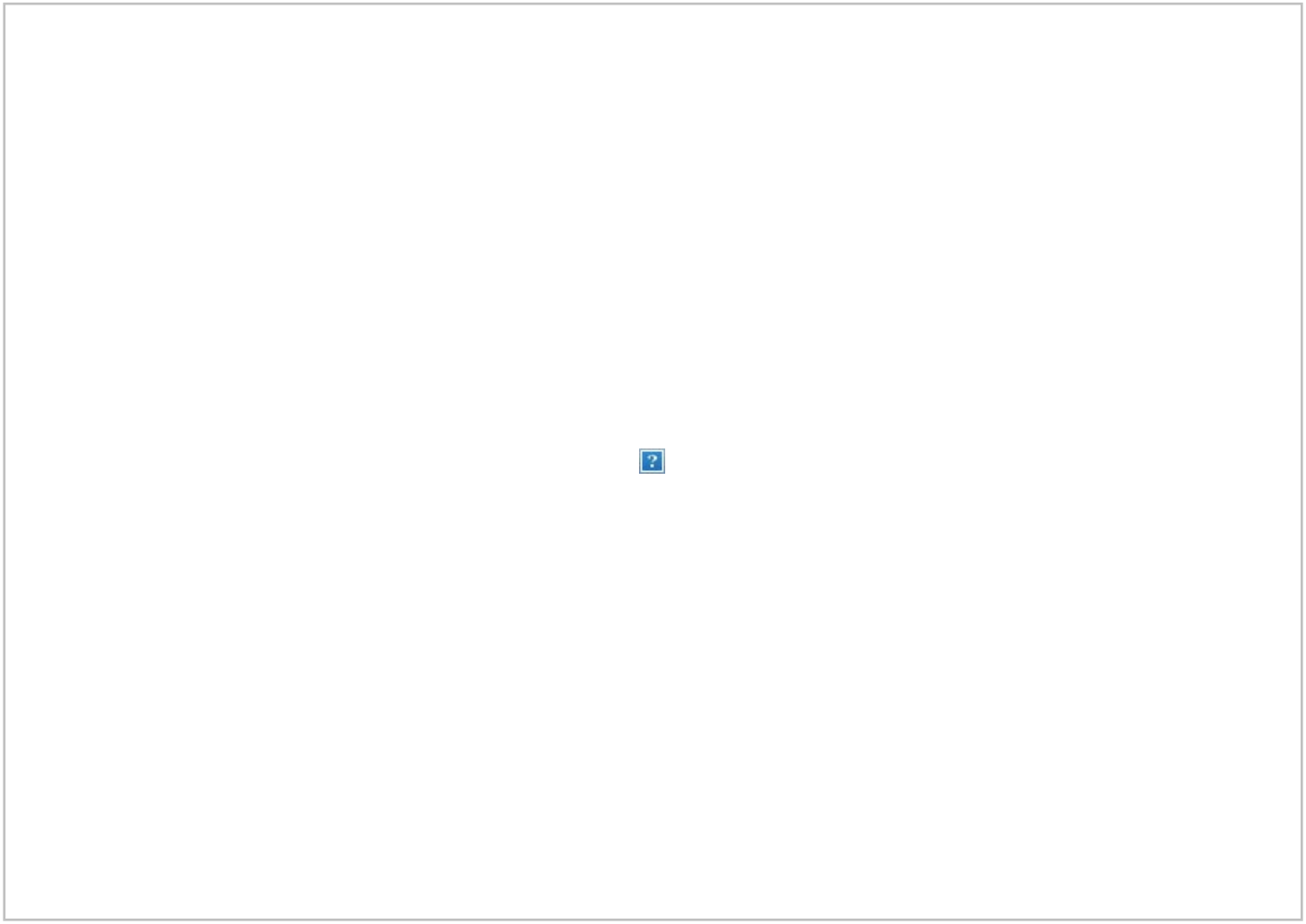
If optional parameters are not specified, the indicated default values will be used, and optional parameters need only be included if the value is required to be different than the default.

Note that the **src=** url entry should use the vanity host name of the specific customer org and should match the hostname property that is being passed to the script.

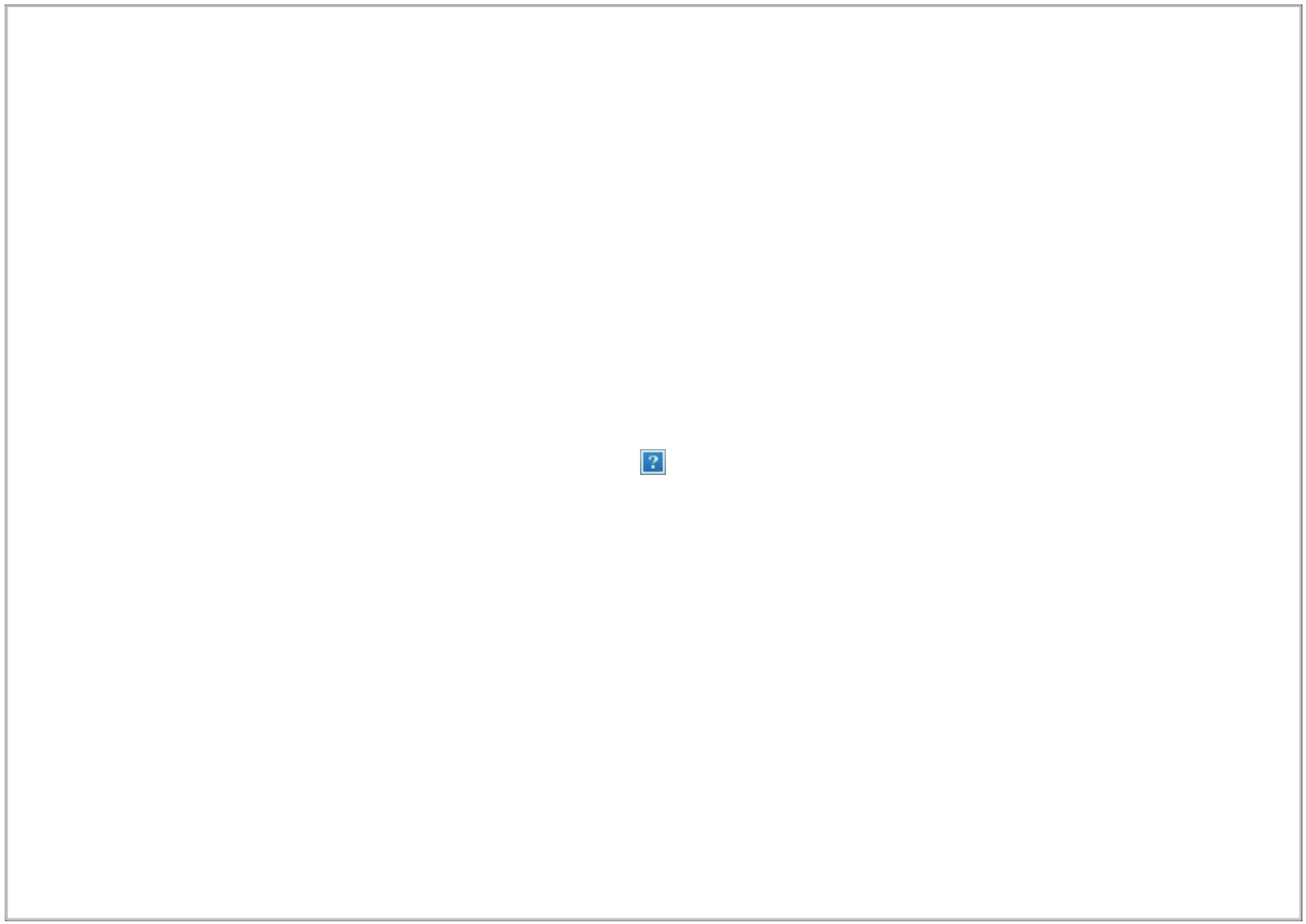
Now when this page is loaded, the following process should occur:

1. The **get-navbar.js** script is loaded in the browser and executed.

2. It determines the set of parameters passed or default values - if the parameter **?debug=true** is appended to the request for the html page, the script parameters are output as browser console debug messages to validate the correct default or passed values are beingused:



3. It creates an iFrame element, sets the properties based on the parameters, attaches to the current page at the anchor id specified and renders the content - per the example navbar.html it should look something like this:



Notice in this case that the Customizer extension to modify the navigation bar by adding WebEx links has been maintained.

Troubleshooting / Validation

Below are some common troubleshooting issues and how to potentially solve those issues and validate that the embedded navbar is working properly. Additional console log statements could be added to the **get-navbar.js** script to help with debugging.

Issue	**Potential Solutions**
get-navbar.js script not loaded	Script src url incorrect or file not present in /pv-connections/customizations/embed-navbar directory.
iframe content not loaded	Check browser trace for request to /homepage/web/pageHeader is successful. Check script hostname property value - verify in console debug output.
iframe not displayed	Host page does not contain an anchor matching anchorid property or anchorid property value incorrect - verify anchorid in console debug output.

iframe dimensions incorrect	Verify script property values in console debug output.
embed-nav.js & embed-nav.css not loaded in pageHeader response	1. Customizer extension does not exist in app registry. 2. Extension exists but include file paths or filenames incorrect. 3. Files are not present in the directory path referenced by the include statements.

Current list of known issues being worked (general):

1. Apps menu on the search page is showing all apps (its supposed to hide the apps that are NOT available in stand-alone mode in MT) - slack
2. IE11 not working for MT deployment (homepage, profiles, communities etc.) - slack
3. Blogs provisioning API responds with 302 redirect (does not work) on some deployments

Current list of known issues being worked (MSP specific):

1. SAML IDP (AD fs) issue Belsoft

New issues found: