2.13 — End User Interface
## CONTENTS

1. Accessing and Customizing the EUI .................................................. 3
2. Working with the Default EUI Templates .............................................. 6
3. Using Default Searches ................................................................. 26
4. Permissions and Visibility in the EUI .................................................. 28
5. Macros, Methods, and Conditionals .................................................... 36
6. EUI Template Reference ................................................................. 40
7. Text and Localization in the EUI ....................................................... 43
Accessing and Customizing the EUI

End users use an end user license and access a special interface called the End User Interface (EUI). All end users belong to an end user group, which is defined in the Group Permissions wizard, and end users in different groups may see different EUIs. For instance, external customers are part of one group that sees an external-facing EUI, while end users who are company employees see an internal-facing EUI. In either case, the end user license and EUI are designed for users who typically submit records or requests but do not work on them.

The default EUI covers a range of common uses, such as support applications, contract management, and document management. However, the EUI can also be highly customized depending on the needs of your users. You can:

- Construct different screen layouts
- Provide your own style of toolbar
- Adjust the options on the main page
- Add your own fonts, color scheme, logo, and buttons using standard HTML and CSS
- Create a home page for each group, with customized navigation options

You can even direct power users to the EUI if you want to use a more customized interface for them.

Accessing the EUI

End users can access the EUI in several ways. For example, the standard knowledgebase login page offers an End User option for the interface. More commonly, end users access the system through a company web portal or a hotlink stored as a bookmark or sent by email. These methods automatically log the user in to the appropriate interface, without requiring the user to select the EUI option or manually enter login credentials.

When a user enters the EUI, the default setup shows tables and links based on their group membership. For example, users in the Customer group see links to the Support Cases they've submitted, which can be accessed on the Home tab and the View drop-down menu.

![Custom Portal](image)

Depending on their permissions and your setup, end users may also use the EUI to complete other tasks, such as the following:

- Search, view, and print FAQs
- Submit new items in any table
- Edit items defined as their own in any table
- View items in any table, as well as run and save simple and complex searches
- Edit their user profile

**Directing Power Users to the EUI**

The EUI offers extensive customization options that can be useful for power users as well. In that case, you might design a customized EUI specifically for power users, and you would need to direct power users to that interface automatically. You can automatically send power users to the EUI if necessary, through custom login pages or using Single Sign-on (SSO).

To use custom login pages, work with Support to compose hyperlinks that direct the user to one interface or the other. With this method, you can simply distribute the EUI link, Power User Interface link, or both to each user as appropriate.

To set the EUI as the default interface option with SSO, open the user record and set the Preferred Interface field to "EUI." The "Based on Permission" value points to the setting on the General tab of the user's group, where the group is configured as a Power User or End User group. Users with at least one Power User group are shown the Power User Interface unless their Preferred Interface is set to "EUI."

If you don't have the Preferred Interface field in your system:

1. Create a new Choice field in the People table.
2. Name it Preferred Interface.
3. Assign it a new choice list with the options "Based on Permission" and "EUI."
4. Set the default value to "Based on Permission."

If you need to test this method without setting up an SSO-friendly test user, you can use this hyperlink syntax to pass the EUI interface preference, replacing `KBName`, `login` and `password`, but make sure to encrypt the URL if you need to send it to an external party:

https://[KBName].agiloft.com/gui2/login.jsp?keyID=0&kb=[KBName]&user=[login]&password=[password]&State=Main&gui=enduser&euiurl=/eui2template/main.php

**Customizing the EUI**

The EUI is customized by editing EUI templates, which determine how different parts of the EUI look. For example, one template determines how the navigation menu looks and what options are included in the menu; another template determines the items in the search area. Not all templates stand alone, and some are used by other templates or included in other pages. For instance, the `header.htm`, `footer.htm`, and `menu.htm` templates are included in every page of the EUI.

© 2021 Agiloft Inc.
In addition to the HTML-based EUI that uses EUI templates, Agiloft offers a Legacy EUI with limited functionality.

Editing EUI Templates

All EUI templates are stored as records in the EUI Templates table, and editing the templates allows you to customize different parts of the EUI. However, before editing anything in the EUI, we strongly recommend that you have a basic understanding of CSS and HTML. W3 Schools is a helpful online resource for learning about those topics. Once you have a grasp of HTML and CSS, you’ll find it easier to make changes to individual EUI templates. One you understand CSS in particular, you can make global formatting changes by editing the style.css template file.

When you make changes to any record in the EUI Templates table, save a copy of the text so that you can revert the code back to its original state if necessary.

For more information about how to customize some of the main EUI templates, see Working with the Default EUI Templates. For general information about the EUI templates, see EUI Template Reference.

Macros

When you're customizing the EUI, understanding how macros work is important. The main template for the home page is main.php, but this template doesn't directly contain the elements displayed on the home page. Instead, it uses the macro #ew_include() to pull in the contents of three other template records: header.htm, home.htm, and footer.htm. In turn, header.htm uses the same macro to include the template menu.htm, and home.htm includes search-form.htm. The header.htm file also uses macros to reference to the logo, plus user information and a logout link in the upper-right corner. For more information on macros, see Macros, Methods, and Conditionals.

© 2021 Agiloft Inc.
Working with the Default EUI Templates

This topic provides instructions on how to control which tables are displayed on the home page of the End User Interface (EUI), as well as which tables and tabs appear in the navigation menu. The tables and tabs that the EUI displays are controlled by records in the EUI Templates table, which also control the functionality and appearance of the EUI. For information on changing any text that appears on the EUI, see Text and Localization in the EUI.

In addition to working with specific EUI templates, use these pages to learn more about other aspects of the EUI that involve working with templates:

- Creating a New Page Tutorial: Learn how to manually add a new record submission form for a table.
- EUI Portal Tutorial: Learn to create a very basic EUI portal from scratch.
- Using Quick Links to Prepopulate Records in the EUI: Learn how to use parameters and macros in the EUI to create records that are populated with content. This page requires knowledge of web development and JavaScript.
- EUI Setup Tips: Learn about some additional setup tips regarding single sign-on, drop-down menus, language selection, and browser pop-ups.

For more information on getting started with the EUI, and to complete interactive practice exercises, we highly recommend signing up for the online training and completing the EUI chapter in Unit 2.

Prerequisites

Before making changes to the EUI, we recommend that you have a basic understanding of the following topics:

- HTML
- CSS
- Macros, methods, and conditionals, as they’re used in the EUI

For more information on HTML and CSS, visit W3 Schools.

Working with Home Pages

The EUI has three predefined home page template files that determine the layout of the home page. Two of the template files are mutually exclusive and display multiple tables to the user, whereas the third file is used when the user only has permission to view a single table. This third file is used because the other two template files are not optimized for viewing a single table. The system automatically calculates the tables available to the user and displays the appropriate home page.
home.htm—Default Layout

The default home.htm template file provides a dynamic two-column display. The system dynamically calculates the layout to list the available tables in order from left to right and top to down. Available links to create or view items appear under each table heading.

home-1column.htm—Alternate Layout

The alternate home-1column.htm template file arranges each table on its own row with the links in a second column to the right.
home-1table.htm—Default Single-Table Layout

The home-1table.htm template uses a similar layout to the home-1column.htm template. The primary differences are larger icons and text in home-1table.htm, as well as an additional block of text to fill space on the page.

Welcome to the Agiloft Custom Portal
You are on the Custom Portal Homepage. From here you can create new records, View or edit existing records, and depending on your permissions and your company setup, you may be able to browse frequently asked questions by clicking FAQs and view or update your user account information using My Profile.

- Support Cases
- Create a Support Case
- View my Support Cases
- FAQs

Change the Home Page

The home page template file that the system uses is configured in the translation.properties file, which is located in the EUI Templates table. By changing one or more variables in translation.properties, you can choose the home page that the EUI uses. The translation.properties file has three variables near the top that define the home page template files:

- **global.home.file**: Defines the main home page file. The default value is home.htm. If you prefer the alternate layout, change the variable to home-1column.htm. If neither of the predefined home pages suit your needs, you can also create your own home page and enter the template name here.
- **global.home.one.table.file**: Defines a third home page file that's displayed if the user who logs in only has access to a single table. The default value is home-1table.htm.
- **global.search.file**: Defines the file that controls the tables displayed in the global search area. The default value is home-multiple-search.htm.

Taken together, the template files defined by these variables determine which tables are shown to users, the ordering of the tables, and other formatting. Although some page components are static, such as the Logout link, other links and headings are controlled by a combination of user permissions, variables defined in translations.properties, and code in the template files.

To change the default home page:
1. If you want to use an entirely new home page, copy and edit one of the default home page files in the EUI Templates table, or create a new template record. Skip this step if you want to change the home page to one of the predefined template files.
   a. Make any changes to the code as desired.
   b. Save the file.
2. Open translation.properties.
3. Update one or both of the global.home and global.home.one.table variables to point to the new home page file.
4. Click Save.

The system counts the number of tables available to a user by using code in the menu.htm template file. If the table count results in a total value of 1, the main.php template file displays the single-table home page. Otherwise, the main home page file is displayed.

Change the Order of Tables

Each table included in the home page has its own block of code. In the default home page, each block of table code looks like the following:

```plaintext
#if ($ewText.get("show.incidents")=="yes" & & ($ewPermission.
table("incident","view_own") | | $ewPermission.table("incident","v_own_with_ssearch") | | $ewPermission.table("incident","v_oth_with_ssearch")))
  #set($rowcount=$rowcount+1)
  ...
#end
```

Notice that the code begins with an #if directive and ends with an #end directive. The order of these code sections determines the order in which tables are displayed to users. The alternate home page follows a similar logic, but the code looks slightly different.

To change the order of the tables on the home page:
1. Open the desired home page file.

Before making changes to any code, create a copy of the existing text so that you can revert the code back to its original state if necessary.

2. Cut and paste one table's section above or below another section. Be careful to include all of the section related to the table. Pay special attention to nested #if and #end directives so that you cut and paste in the correct location.

3. Click Save.

For more information on how permissions are used in the home page to display tables to users, see Permissions and Visibility in the EUI.

Review the Home Page Code

The following section is a snippet of code for a single table in the default home.htm template. It contains added comments that explain the purpose of the code, so if you want to make manual adjustments to your home page, use the comments to help you know how each block of code affects the page.

```
Support Cases

Create a Support Case
View My Support Cases
FAQs

Support Cases Table

#if ($ewText.get("show.support.cases")="yes" &&
(ewPermission.table("case","view_own") ||
ewPermission.table("case","v_own_with_ssearch") || ewPermission.table("case","view faq") || ewPermission.table("case","view faq_with_ss")))

#set($rowcount=$rowcount+1)
<td class="home-section">
<div class="home-cont">
<div>
<img src="#ew_image('/gif/light_orange_red_20/master_light_orange_red_20_57.png')" style="vertical-align: bottom" width="20" height="20" alt="/"&nbsp;&nbsp; $ewText.get('home.sc.h3') </div>
</div>
</div>
```

© 2021 Agiloft Inc.
If the show.support.cases variable is set to Yes and the user’s record-level permissions include view own, view own with saved search, view FAQs, or view FAQs with saved search, then the code sets the row count and displays the table heading.

```html
#if($ewPermission.table("case","create"))
<img src="/gif/light_orange_red_16/master_light_orange_red_16_96.png" alt=""/>
<a href="#ew_forward("new_supportcase.html")">$ewText.get('home.sc.new.link')</a>
<br />
#end
```

If the user has create permissions, the Create a Support Case link is displayed.

```html
#if ($ewPermission.table("case","view_own") || $ewPermission.table("case","v_own_with_ssearch"))
<img src="/gif/light_orange_red_16/master_light_orange_red_16_02.png" alt=""/>
<a href="#ew_forward("my_supportcases.html")">$ewText.get('home.sc.my.link')</a><br />
#end
```

If the user has view own or view own with saved search record permissions, the View My Support Cases link is displayed.

```html
#if ($ewText.get("show.sc.faqs")=="yes" && ($ewPermission.table("case","view_faq") || $ewPermission.table("case","view_faq_with_ss")))
<img src="/gif/light_orange_red_16/master_light_orange_red_16_157.png" alt=""/>
<a href="#ew_forward("supportcases-faq.html")">$ewText.get('home.sc.faq.link')</a>
</a>
#end
```

If the show.sc.faqs variable is set to Yes and the user has view FAQs or view FAQs with saved search permissions, the FAQs link is displayed.

```html
</div>
</div>
</td>

#if ($rowcount%2==0)
</tr>
<tr>
<td class="h-line"></td>
<td></td>
<td class="h-line"></td>
</tr>
<tr>
<td class="v-line"></td>
#end
#end
```
Depending on the table's location in the code and the row count, this defines whether it's placed in the first or second column and inserts the appropriate vertical or horizontal lines.

Working with the Navigation Menu

The navigation menu refers to series of tabs and drop-downs in the top-right of the default home page. Its contents are defined in the `menu.htm` template and `translation.properties` files.

![Navigation Menu](image)

Change the Navigation Menu Tabs

The tabs on the navigation menu are controlled by `menu.htm` and variables in `translation.properties`. You can control whether the New, View, FAQs, and My Profile tabs appear at all or only to certain users.

To control which tabs can appear at all:

1. In the EUI Templates table, open `translation.properties`.
2. Find the note beginning with "#These determine whether the menu bar shows..." and notice the variables immediately below the note.
3. If you do or do not want to display a tab, change the corresponding variable to Yes or No, respectively. For example, if you don't want to display the View tab, the variable would look like the following:
   ```
   show.menu.view=no
   ```
4. Click Save.

   ⚠️ If the variable for a tab is set to Yes, users might not see the tab if you've applied conditional visibility to it in `menu.htm`. See the steps below for more information.

To control which tabs are visible depending on a user's group, team, or permissions:

1. In the EUI Templates table, open `translation.properties`.

© 2021 Agiloft Inc.
2. Find the note beginning with "#These determine whether the menu bar shows..." and notice the variables immediately below the note.

3. For tabs you want to display to at least some users, make sure the corresponding variable is set to Yes. If the variable is set to No, users will not be able to see the tab, regardless of any conditions you create. For example, if you want to display the New, View, and My Profile tabs to some users, but you don’t want to display the FAQs tab to any users, the variables would look like this:

   show.menu.new=yes
   show.menu.view=yes
   show.document.faqs=no
   show.myprofile=yes

4. Click Save.

5. Open menu.htm.

6. Find the beginning line that corresponds to the tab you want to make conditionally visible. For example, the New tab begins with the following line:

   #if ($ewText.get("show.menu.new")=="yes")

7. On the same line, add a logical operator and an appropriate method to define the type of conditional visibility you want to use. For example, if you only want users in the Contract Creator group to see the New tab, your code would look like this:

   #if ($ewText.get("show.menu.new")=="yes" && $ewUser.isInGroup("Contract Creator"))

For more information on using conditionals and methods, see Macros, Methods, and Conditionals and EUI Method Reference.

8. Click Save.

Change the Links under the Navigation Menu Tabs

Like the navigation menu tabs themselves, the links they contain are controlled by menu.htm and variables in translation.properties.
You can control whether a link appears at all or only to certain users. In most cases, it's better to make the links appear to only certain users by creating conditions that check a user's team, group, or table permissions. This is because the method to control whether a link can appear at all also controls whether a table may appear at all in the EUI. For example, if you prevent the Support Case link from appearing at all by setting the `show.support.cases` variable to No, you also prevent the Support Cases table from appearing on the EUI. However, if a table is no longer in use or end users shouldn't have access to it, it can sometimes be useful to completely remove the table and its associated links from the EUI.

To control which links can appear at all:

1. In the EUI Templates table, open `translation.properties`.
2. Find the note beginning with "#These values determine whether a table..." and notice the variables immediately below the note.
3. If you do or do not want to display a link, change the corresponding variable to Yes or No, respectively. Each variable may control multiple links. For example, if you set the `show.support.cases` variable to No, both the Support Case and My Support Cases links are removed.
4. Click Save.

⚠️ If the variable for a table is set to Yes, users might still not see any related links if you've applied conditional visibility in `menu.htm`. See the steps below for more information.

To control which links are visible depending on a user's group, team, or permissions:

1. In the EUI Templates table, open `translation.properties`.
2. Find the note beginning with "#These values determine whether a table..." and notice the variables immediately below the note.
3. For links you want to display to any users, make sure the corresponding variable is set to Yes. If the variable is set to No, users will not be able to view the link, regardless of the condition.
4. Click Save.
5. Open `menu.htm`.
6. Find the link that you want to make conditionally visible. Each link uses a few lines of code, and it's placed within a section of code for the corresponding tab. For example, lines 4 - 6 of the code below refer to the Service Request link, and the rest of the code indicates that it's within the New tab on the navigation menu:

```html
#if ($ewText.get("show.menu.new")=='yes')
  <li id="new" class="submenu"><a>$ewText.get('menu.new')</a></li>
  <ul>
    #if ($ewText.get("show.service.requests")=='yes' && $ewPermission.table("helpdesk_case","create"))
      <li><a href='#ew_forward("new_servicerequest.html")'>$ewText.get('menu.new.sr')</a></li>
    #end
  #end
  ...
</ul>
<li>
#end
</li>
```
7. Edit the existing conditional to define the type of conditional visibility you want to use. For example, the above example requires that the `show.service.requests` variable in `translation.properties` is set to Yes and that the user has permission in the Service Requests table to create records. If you also want to require that the user is in the Service Manager group, your code would look like this:

```html
#if ($ewText.get("show.menu.new")=='yes')
  <li id="new" class="submenu"><a>$ewText.get('menu.new')</a>
    <ul>
      #if ($ewText.get("show.service.requests")=='yes' && $ewPermission.table("helpdesk_case","create") && $ewUser.isInGroup("Service Manager"))
        <li><a href='#ew_forward("new_servicerequest.html")'>$ewText.get('menu.new.sr')</a></li>
      #end
    </ul>
  </li>
#end
```

For more information on creating conditionals, see [Macros, Methods, and Conditionals](#).

8. Click Save.

### Working with the Header

The header refers to the top area of the EUI that includes the company logo, language drop-down, user information, and logout link. It’s controlled by `header.htm` and variables in `translation.properties`. The navigation menu is not considered part of the header and is controlled separately with `menu.htm`.

### Customize the Logo

One of the first changes you may want to make is to replace the existing logo in the EUI with your company logo. You can do this by editing the `#ew_image` macro and changing its file path.

1. You first need to add the image to the server, so go to **Setup > Look and Feel > End User Schemes** and edit any existing scheme.
2. Upload the new logo on the Icons tab of the Look and Feel wizard and click Finish. This makes it available in the image directory so that it can be referenced by the #ew_image macro.

3. Go to the EUI Templates table and edit header.htm.

4. In the Body field, locate the following line:

```
<a href="#ew_forward("main.php")"><img src="#ew_image('/gif/agiloft-logo.png')" width="185" height="75" alt="Agiloft" style="vertical-align: middle" /></a>
```

5. To change the logo, replace the reference to agiloft-logo.png in the #ew_image macro with the new file name.
   
a. Change the alt attribute value from "Agiloft" to the new logo name. This is the text that screen readers use and that appears when you hover over the image.
   
b. Change the width and height properties to match the new logo dimensions, or leave out the dimensions to display it at its full size.
   
For example, your new code may look like the following:

```
<a href="#ew_forward("main.php")"><img src="#ew_image('/gif/sample-logo.png')" width="225" height="100" alt="Company logo" style="vertical-align: middle" /></a>
```

6. Click Save. The new logo will be visible when you log into the EUI.

---

**Customize the Logo with Conditional Branding**

Suppose you want to display a different logo to users in a particular group or team. You can achieve this by adjusting the code in the previous section.

1. Follow the steps in the previous section to upload a second image.

2. Go to the EUI Templates table and edit header.htm.

3. In the Body field, locate the line you edited earlier. From the previous example, it looks like this:

```
<a href="#ew_forward("main.php")"><img src="#ew_image('/gif/sample-logo.png')" width="225" height="100" alt="Company logo" style="vertical-align: middle" /></a>
```

4. Create a conditional that displays each logo to the appropriate users:
   
   - If you want to display one logo to users in a certain group and the other logo to everyone else, your code might look like this:
If you want to display one logo to users in a certain team and the other logo to everyone else, your code might look like this:

```html
#if ($ewUser.isInGroup("Contract Creator"))
<img src="#ew_image('/gif/sample-logo.png')" width="225" height="100" alt="Company logo" style="vertical-align: middle" />
#else
<img src="#ew_image('/gif/new-logo.png')" width="220" height="150" alt="New logo" style="vertical-align: middle" />
#end
```

For more information on creating conditionals, see Macros, Methods, and Conditionals.

5. Click Save.

Working with Tables

The EUI comes with several default tables already included. Each table has individual template files for each of its pages in the EUI, as well as variables in the `translation.properties` template file that these and other pages use. For example, the Support Cases table has template files that allow users to create new support cases, view support cases, and search support cases. The text that these pages use, such as the header or search label, is defined with variables in `translation.properties`.

You can edit the template files to change the formatting and content of individual EUI pages. For example, you can move or resize a page's header, or you can apply a different saved search. Some changes, like resizing the header, require only a basic knowledge of HTML. Other changes, like applying a different saved search, require an understanding of the macros used in the EUI. For more information on applying a new saved search to a page, see Using Default Searches. For more information on macros in general, see Macros, Methods, and Conditionals.

Add a New Table

If you want to add a table to the EUI that is not included in the default setup, you must update the EUI templates to provide this access. To do so, you can either add the code manually or generate the code using the New EUI Table Excel template. We strongly recommend that you use the Excel template, which you can download here. Using the template is much easier and much less likely to result in errors than entering the code manually. The steps below describe how to use the Excel template.
Before you start, go to Setup [Table Name] for the table you’re adding and note the following information on the General tab:

- The plural table label
- The logical table name

Update translation.properties

The first step to adding a new table is to create the variables used in the translation.properties file. If you have a multilingual knowledgebase, you need to update the translation.properties file for each language, using the same variables created below but with appropriately translated values.

1. Download the New EUI Table Excel template.
2. Click the Table Name sheet, located on the bottom of the Excel window.
3. Enter the table information, including the name; plural label; a unique, three-letter abbreviation; and the logical table name. You can use any three-letter abbreviation you want, but it must be unique and not match any existing table abbreviations in translation.properties.

<table>
<thead>
<tr>
<th>New table name:</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plural:</td>
<td>Products</td>
</tr>
<tr>
<td>Abbreviated (lower case):</td>
<td>prd</td>
</tr>
<tr>
<td>Logical table name:</td>
<td>product</td>
</tr>
</tbody>
</table>

4. Click the translation.properties sheet and copy the code from Column N. Don't close the spreadsheet until this entire process is complete because you still need code from the other Excel sheets.
5. In your knowledgebase, go to the EUI Templates table and open translation.properties.
6. Paste the code from Column N into the bottom of the Body field.

⚠️ The code in translation.properties is separated into sections and listed in alphabetical order, but this is only for readability. The code works anywhere you place it, but it's typically good practice to keep code organized.

7. Click Save.

Create the New Template Files

Next, create the template files for the pages that the new table will use.
1. Create the first template file:
   a. Go to the EUI Templates table and click New. This template will be for displaying all records in the table.
   b. In the Name field, enter a name like all_records.html or all_widgets.html.
   c. Provide a Description, but leave the Body field blank for now.
   d. Click Save.
2. Repeat the previous steps to create pages for creating new records, viewing the user's own records, and using searches on the table. For example, you might create pages titled new_record.html, my_records.html, and my_records-search.htm. The search page must end in ".htm" and the other three pages must end in ".html" for the table to be correctly added.

   Although the tables can be titled anything you like, we recommend that you follow a consistent naming convention throughout the EUI.

3. Go to the table in your knowledgebase and create a search called "My [Plural Table Label]" that finds all records assigned to the currently logged in user. For example, you might name it My Records or My Widgets.
4. Retrieve the code for the first template file:
   a. In the Excel spreadsheet, click the EUI Templates_import sheet.
   b. Copy the code from cell D2 and paste it into a Word document or other word processor. You must paste it into a word processor before pasting it into the template file. Otherwise, Excel automatically includes quotation marks that prevent the HTML from working properly.
   c. Copy the code from the Word document, go the the EUI Templates table, and open the all_records.html page you created.
   d. Paste the code into the Body field and click Save.
5. Repeat the previous steps and use cells D3, D4, and D5 to retrieve and insert the correct code for the respective pages.

   Tip
   To define the view for a specific template file, add the View Name (not View Label) to the #ew_table line, in the blank expression after the table name.

Add the Table to the Home Page

After you've created the new template files, edit the home page template file so that the new table appears on the home page.
If you are using the alternate home page or a custom home page, you need to manually enter the code in the template file. To help prevent errors, use the code for existing tables as a model when you add the new table.

1. In the Excel spreadsheet, click the home.htm sheet.
2. Copy the code from cell B1 and paste it into a Word document or other word processor. As before, you must paste it into a word processor before pasting it into the template file.
3. Copy the code from the Word document, go to the EUI Templates table, and open home.htm.
4. Scroll down to find the last two #end directives that are directly after one another, and place your cursor after the second #end directive.

```
<i class="h-line"></td>
</tr>
<tr>
    #else
    <td class="v-line"></td>
    #end
</tr>
</table>
</div>
<!--/Home Links-->
</div>
<if($isInclude('global.search.file'))
    #ew_include('quicklinks.htm')
</if>
```

5. Press Enter and paste the code from the Word document. Make sure the code is placed after the second #end directive but before the </tr> tag.
6. Click Save.

Create Links from the Navigation Menu

Now that users can view the table on the home page, add some links to the navigation menu. This lets users create new records in the table and view records that they own, assuming they have the appropriate permissions.

1. In the Excel spreadsheet, click the menu.htm sheet.
2. Copy the code in cell B1 and paste it into a Word document.
   a. Copy the code from the Word document, go to the EUI Templates table, and open menu.htm.
b. Find the last \#end directive in the first unordered list. An unordered list begins with a <ul> tag and ends with a matching </ul> tag.

```html
#end
```

b. Place your cursor directly after the \#end directive and press Enter.

d. Paste the code from the Word document. Make sure the code is placed after the \#end directive and before the </ul> tag. Don't close the record yet.

3. Go back to the Excel spreadsheet, copy the code in cell B2, and paste it into a Word document.

   a. Copy the code from the Word document and go back to menu.htm.

   b. Find the last \#end directive in the second unordered list, which is near the bottom of the field.

```html
#end
```

b. Place your cursor directly after the \#end directive, press Enter, and paste the code. Make sure the code is placed after the \#end directive and before the </ul> tag.

d. Click Save.

Add the Table to the Single-Table Home Page

The next step is to add the table to the single-table home page, which is displayed in cases where the user only has access to this table.

1. In the Excel spreadsheet, click the home-1table.htm sheet.
2. Copy the code in cell B1 and paste it into a Word document.
3. Copy the code from the Word document, go to the EUI Templates table, and open `home-1table.htm`.
4. Find the last `#end` directive in the table element. It’s near the bottom of the page and preceded by two `<div>` tags.

```
<a href="#ew_forward("my_opentasks.html"
</div>
#end
</td>
</tr>
</table>
<!--Home Links-->
</div>
<div class="sidebar">
#ew_include("$ewText.get('global.search.f
```

5. Place your cursor directly after the `#end` directive and press Enter.
6. Paste the code from the Word document. Make sure the code is placed after the `#end` directive and before the `</td>` tag.
7. Click Save.

---

**Make the Table Searchable from the Global Search Block**

The final step is to add the table to the global search block so that users can search it. This step involves editing three template files.

**Edit home-multiple-search.htm**

1. In the Excel spreadsheet, click the `home-multiple-search.htm` sheet.
2. Copy the code in cell B1 and paste it into a Word document.
3. Copy the code from the Word document, go to the EUI Templates table, and open `home-multiple-search.htm`.
4. Find the last `#end` directive in the div element. It's near the bottom of the page and preceded by a `<br>` tag.

```html
#end
#if ($(sourcetext).get("show.contracts") == "yes") & & $(sourcetext).get("show.contracts") & & $(sourcetext).get("show.contracts")
<input name="tables" type="checkbox" value="contract">
<br>
#end
</div>
</form>
</div>
</div>
<!--/Search form-->
```

5. Place your cursor directly after the `#end` directive and press Enter.

6. Paste the code from the Word document. Make sure the code is placed after the `#end` directive and before the `</div>` tag.

7. Click Save.

**Edit multiple-search.htm**

1. In the EUI Templates table, open `multiple-search.htm`. Notice the `#if` directives and the `my_table` parameter that they each contain.

2. Find the last `#if` directive and make note of the number that follows its `my_table` parameter. For example, it might be `my_table7`.

```html
#end
    #ew_query("document" "my_table7" "queryId"
    "showToolBar=false&showNavigation=true&showfastsearch=false")
#end
```

3. In the Excel spreadsheet, click the `multiple-search.htm` sheet.

4. In cell B3, edit the `my_table` parameter so that it uses the next table number in sequence. For example, if the last `#end` directive in `multiple-search.htm` contained `my_table7`, you would enter `my_table8` in cell B3.

5. Copy the updated code in cell B1 and paste it into a Word document.

6. Go back to `multiple-search.htm` in the EUI Templates table.
7. Find the last `#end` directive in the div element. It's near the bottom of the page and preceded by several visualization parameters.

```
  #ew_query("document" "my_table?" "queryId"
"showToolBar=false&showNavigation=true&showfastsearch=false")
  #end
  
</div>
<!--Search form-->
```

8. Place your cursor directly after the `#end` directive and press Enter.

9. Paste the code from the Word document. Make sure the code is placed after the `#end` directive and before the `ibrator` characters.

10. Click Save.

**Edit multiple-search-result.html**

1. In the Excel spreadsheet, click the multiple-search-result.html sheet.

2. Copy the code in cell B1 and paste it into a Word document.
   a. Copy the code from the Word document, go to the EUI Templates table, and open `multiple-search-result.html`.
   b. Find the last `#end` directive in the first part of the body element. It's preceded by a `</div>` tag and immediately followed by a `</script language="JavaScript">` element.

```
  #ew_table("document" "settings"
"showToolBar=false&showNavigation=true&showfastsearch=false")
  #end
  
<\/div>
  
<script language="JavaScript">
  !--
  var queryId = '$esc.jsavscript($request);
  var requesqueryId = $request; $getParameter;
```

   c. Place your cursor directly after the `#end` directive and press Enter.
   d. Paste the code from the Word document. Make sure the code is placed after the `#end` directive and before the `</script language="JavaScript">` element. Don't close the record yet.

3. Go back to the Excel spreadsheet, copy the code from cell B2 on the multiple-search-result.html sheet, and paste it into a Word document.
a. Go back to `multiple-search-result.html` in the EUI Templates table, and find the last `#end` directive in the second part of the body element. It’s near the bottom of the page and preceded by several visualization parameters.

```html
#end

    #if ($queryTables.contains('doc'))
    #ew_query("document" "my_table7" "queryId"
"showToolBar=false&showNavigation=false&showfastsearch=false")
    #end ←–

    }
}

// →–
```

b. Place your cursor directly after the `#end` directive and press Enter.

c. Paste the code from the Word document. Make sure the code is placed after the `#end` directive and before the `)` character.

d. Click Save.

The new table is now added to the EUI and searchable from the global search block. If you want to adjust how the table appears on the home page or edit the links on the navigation menu, see [Working with Home Pages](#) and [Working with the Navigation Menu](#), respectively.
Using Default Searches

Default searches can be configured on specific pages in the EUI to determine the records that are displayed. For example, when a user clicks a page, you might only want to display records that are assigned to them.

1. Navigate to the table that contains the records on which you want to run the search.
2. Hover over Search to open the drop-down. Note the name of the search you'd like to use as the new default search, or click New to create a new saved search that you want to use.
3. Navigate to the EUI Templates table.
4. Edit the .html record that corresponds to the page in the EUI on which you'd like the search to run.

Updating the Default Search

To update the default search, you can use an existing saved search or create a new saved search.

1. Navigate to the table that contains the records on which you want to run the search.
2. Hover over Search to open the drop-down. Note the name of the search you'd like to use as the new default search, or click New to create a new saved search that you want to use.
3. Navigate to the EUI Templates table.
4. Edit the .html record that corresponds to the page in the EUI on which you'd like the search to run.

Prerequisites

Before making changes to the EUI, we recommend that you have a basic understanding of the following topics:

- HTML
- CSS
- Macros, methods, and conditionals, as they're used in the EUI

For more information on HTML and CSS, visit W3 Schools.
By default, the .html records you need to edit are named after the page titles in the EUI. For example, the My Service Requests page is my_servicerequests.html, the My Incidents page is my_incidents.html, and so on.

5. Locate the line of code beginning with #set that lists the current default search:

```html
#set ($defaultsearch="My Incidents")
     <div class="wrapper">
       <div class="top-space"></div>
       <div class="main">
         #ew_include("header.htm")
       </div>
     </div>
     <div class="content">
       <div style="float: left"><h2>$ewText.get('myinc.header')</h2>
         <div class="title-desc"></div></div>
       #ew_include("my_incidents-search.htm")
       #ew_status()
     </div>
     <div id="main_frame">
       #ew_table("incident" "" $defaultsearch ""
       "showToolBar=false&showNavigation=true&showfastsearch=false" "my_table" "")
     </div>
```

6. Following $defaultsearch= in the first line, replace the name in quotation marks with the name of the new default search.

7. In the last line of the code block shown above, ensure $defaultsearch is listed in the second set of quotation marks after #ew_table.

8. Click Save. The default search is now updated.
Permissions and Visibility in the EUI

User permissions determine which elements on the EUI are visible to each user. Oftentimes, you may have several groups of users, and you want the different groups to see different elements on the interface. In other cases, you may want the same user to see different parts of a record based on the the page they're viewing. For example, when a user is submitting a support case, you might not show the Created By field because it isn't relevant. In either case, understanding how permissions control visibility in the EUI can help you create an interface that suits your needs.

Prerequisites

Before making changes to the EUI, we recommend that you have a basic understanding of the following topics:

- HTML
- CSS
- Macros, methods, and conditionals, as they're used in the EUI

For more information on HTML and CSS, visit W3 Schools.

Controlling Visibility

The interface that a user sees depends on two factors:

- Their group permissions.
- The status of a table or menu tab in the translation.properties file. Tables and menu tabs can be turned on if their value is Yes and turned off if their value is No. If a table or menu tab is not turned on, end users will not see it, regardless of their permissions.

To illustrate this, the three images below show the home page contents for three different users: one in the Admin group, one in the Document Creator group, and one in the Customer group. Each user sees a different interface based on their permissions. Notice that not even the user in the Admin group sees the Change Requests or Tasks tables, which are turned off in these examples.

A user in the Admin group sees six tables and sets of links:
A user in the Document Creator group sees four tables and sets of links:

- **Service Requests**
  - Create a Service Request
  - View My Service Requests

- **Incidents**
  - Create an Incident
  - View My Incidents
  - View Open Incidents

- **Purchase Requests**
  - Create a Purchase Request
  - View My Purchase Requests

- **Support Cases**
  - Create a Support Case
  - View My Support Cases

- **Contracts**
  - Create a Contract Request
  - View My Contracts
  - View All Contracts

- **Documents / FAQs**
  - Create a Document
  - View My Documents
  - All Documents
  - Search Published Documents / FAQs

A user in the Customer group sees two tables and sets of links:

- **Service Requests**
  - Create a Service Request
  - View My Service Requests

- **Incidents**
  - Create an Incident
  - View My Incidents
  - View Open Incidents

- **Purchase Requests**
  - Create a Purchase Request
  - View My Purchase Requests

- **Documents / FAQs**
  - Create a Document
  - View My Documents
  - All Documents
  - Search Published Documents / FAQs

© 2021 Agiloft Inc.
Control Visibility with Permissions

When users log in, the system checks their group permissions and uses the $ewPermission.table method to determine which EUI elements they see. This method requires two parameters, the logical table name and the permission type, which take the form $ewPermission.table("table","permission"). For example, $ewPermission.table("case","view_own") returns “true” if the currently logged in user user has permission to view their own records in the Support Cases table.

You can insert the $ewPermission.table method in a conditional that surrounds an element, such as a link or menu item. Conditionals are used throughout the records in the EUI Templates table to dynamically control element visibility based on group permissions. For example, consider the following conditional:

```#
if ($ewPermission.table("case","view_own"))
   <li><a href='#ew_forward("my_supportcases.html")'>$ewText.get('menu.view.mysc')</a></li>
#end```

If the user has permission to view their own records in the Support Cases table, the system shows the user a menu item to view a page containing support cases that they own.

To control which permissions control the display of an element:

1. In the EUI Templates table, open the desired template file.
2. Find the method that relates to the relevant table and element.
3. Edit the permission in the second parameter, or add a logical operator to expand the conditional. For a complete list of permission types, see Permission Parameters below. For more information on conditionals and the available logical operators, see Macros, Methods, and Conditionals.
4. Click Save.
5. View the EUI page with the relevant user to verify that the element visibility is working as intended.

The logical table name might be different from the table name that's displayed. To find the logical table name, open the Table wizard for the desired table and check the General tab.
Control Visibility with translation.properties

The translation.properties file in the EUI Templates table controls the global visibility for tables and menu tabs by turning on or turning off specific tables and tabs. For users to see a table or menu tab in the EUI, it must be turned on in translation.properties. To prevent all users from seeing a table or menu tab, you can turn it off.

Table visibility is controlled near the top of the template file, and menu tab visibility is controlled just below the table visibility section. If the show.[table or tab] key is set to Yes for a given table or tab, then it’s included in the interface for those users with permission to view it. If show.[item] is set to No, it’s not shown on the home page or navigation menu to any users, regardless of their permissions.

To control global visibility for tables and tabs:

1. In the EUI Templates table, open translation.properties.
2. Set the relevant table and menu tab keys to Yes or No.
3. Click Save.
4. If other languages are in use, open any other versions of translation.properties and repeat the process.
5. View the EUI to confirm that changes have taken effect.

Translation file has alternate versions for other languages, which allow for a multilingual environment. If your KB is available in multiple languages, be sure to update the keys in each translation file when making changes. For more information on localization and the EUI, see Text and Localization in the EUI.

Table Visibility

Menu Tab Visibility
Keep in mind that whether a user can actually view a table or the items within a tab still depends on the user’s group permissions.

Permission Parameters

The table below lists the most common permission-based parameters for use with the `setPermission` method. Each of these values corresponds to a permission setting in the Table Permissions wizard.

<table>
<thead>
<tr>
<th>Permission Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>create</td>
<td>Create records in the table</td>
</tr>
<tr>
<td>edit_own</td>
<td>Edit their own records in the table</td>
</tr>
<tr>
<td>e_own_with_ssearch</td>
<td>Edit only their own records matching a saved search</td>
</tr>
<tr>
<td>delete_own</td>
<td>Delete their own records in the table</td>
</tr>
<tr>
<td>view_own</td>
<td>View their own records in the table</td>
</tr>
<tr>
<td>v_own_with_ssearch</td>
<td>View only their own records matching a saved search</td>
</tr>
<tr>
<td>view_others</td>
<td>View other people's records in the table</td>
</tr>
<tr>
<td>v_oth_with_ssearch</td>
<td>View only others' records matching a saved search</td>
</tr>
<tr>
<td>view_faq</td>
<td>View FAQs from the table</td>
</tr>
<tr>
<td>view_faq_with_ss</td>
<td>View only FAQs matching a saved search</td>
</tr>
</tbody>
</table>

The table below lists the complete list of permission-based parameters.

<table>
<thead>
<tr>
<th>Permission Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>create</td>
<td>Allow user to create a new record</td>
</tr>
<tr>
<td>delete_own</td>
<td>Allow user to delete own records</td>
</tr>
<tr>
<td>delete_others</td>
<td>Allow user to delete others’ records</td>
</tr>
<tr>
<td>mass_delete</td>
<td>Allow user to delete multiple records</td>
</tr>
<tr>
<td>dele_oth_if_pub_y</td>
<td>Delete others’ records if published field is yes</td>
</tr>
<tr>
<td>massedit</td>
<td>Allow user to perform mass edit</td>
</tr>
<tr>
<td>import_multiple</td>
<td>Allows import of multiple records from a file</td>
</tr>
<tr>
<td>export_multiple</td>
<td>Allows export of multiple records to a file</td>
</tr>
</tbody>
</table>

© 2021 Agiloft Inc.
<table>
<thead>
<tr>
<th>Permission</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>link_multiple_recs</strong></td>
<td>Link multiple records using the Link menu</td>
</tr>
<tr>
<td><strong>print_recs</strong></td>
<td>Print records using the printer icon</td>
</tr>
<tr>
<td><strong>View and Edit Permissions</strong></td>
<td></td>
</tr>
<tr>
<td><strong>view_own</strong></td>
<td>Allow user to view own records</td>
</tr>
<tr>
<td><strong>view_others</strong></td>
<td>Allow user to view others’ records</td>
</tr>
<tr>
<td><strong>view_own_if_pub_y</strong></td>
<td>View own records if published field is yes.</td>
</tr>
<tr>
<td><strong>view_oth_if_pub_y</strong></td>
<td>View others’ records if published field is yes.</td>
</tr>
<tr>
<td><strong>edit_own</strong></td>
<td>Allow user to edit own records</td>
</tr>
<tr>
<td><strong>edit_others</strong></td>
<td>Allow user to edit others’ records</td>
</tr>
<tr>
<td><strong>Saved Search Permissions</strong></td>
<td></td>
</tr>
<tr>
<td><strong>mod_own_s_searches</strong></td>
<td>Allow user to modify/delete saved searches that they created</td>
</tr>
<tr>
<td><strong>mod_all_s_searches</strong></td>
<td>Allow user to modify/delete all saved searches</td>
</tr>
<tr>
<td><strong>mod_own_s_views</strong></td>
<td>Allow user to modify/delete saved views that they created</td>
</tr>
<tr>
<td><strong>mod_all_s_views</strong></td>
<td>Allow user to modify/delete all saved views</td>
</tr>
<tr>
<td><strong>mod_own_s_reports</strong></td>
<td>Allow user to modify/delete saved reports that they created</td>
</tr>
<tr>
<td><strong>mod_all_s_reports</strong></td>
<td>Allow user to modify/delete all saved reports</td>
</tr>
<tr>
<td><strong>pub_s_reports</strong></td>
<td>Allow user to publish saved reports</td>
</tr>
<tr>
<td><strong>v_own_with_ssearch</strong></td>
<td>View own entries that satisfy the saved search</td>
</tr>
<tr>
<td><strong>v_oth_with_ssearch</strong></td>
<td>View others’ entries that satisfy the saved search</td>
</tr>
<tr>
<td><strong>e_own_with_ssearch</strong></td>
<td>Edit own entries that satisfy the saved search</td>
</tr>
<tr>
<td><strong>e_oth_with_ssearch</strong></td>
<td>Edit others’ entries that satisfy the saved search</td>
</tr>
<tr>
<td><strong>d_oth_with_ssearch</strong></td>
<td>Delete others’ entries that satisfy the saved search</td>
</tr>
<tr>
<td><strong>acc_oth_ss</strong></td>
<td>Allow access to others’ saved searches</td>
</tr>
<tr>
<td><strong>acc_oth_ss_py</strong></td>
<td>Allow access to others’ saved searches if Published=Yes</td>
</tr>
<tr>
<td><strong>FAQ Permissions</strong></td>
<td></td>
</tr>
<tr>
<td><strong>view_faq</strong></td>
<td>Allow user to view FAQs</td>
</tr>
<tr>
<td><strong>view_faq_with_ss</strong></td>
<td>View FAQs that satisfy the saved search</td>
</tr>
<tr>
<td><strong>Table Permissions</strong></td>
<td></td>
</tr>
<tr>
<td>Feature</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ed_rec_tab_view</td>
<td>Edit records from table view</td>
</tr>
<tr>
<td>omit_table_label</td>
<td>Allow user to set show label on view/editing on General tab</td>
</tr>
<tr>
<td>copy_recs</td>
<td>Copy $table-label-plural</td>
</tr>
<tr>
<td>show_in_main_tabs</td>
<td>Allow user to see it in main tabs</td>
</tr>
<tr>
<td>allow_sel_ed_flds</td>
<td>Allow user to select editable fields in view they can edit</td>
</tr>
<tr>
<td>apply_to_subtabs</td>
<td>Apply changes to subtables (in groups wizard)</td>
</tr>
<tr>
<td>apply_all_to_subtabs</td>
<td>Apply everything to subtables (in groups wizard)</td>
</tr>
<tr>
<td><strong>Custom Report Permissions</strong></td>
<td></td>
</tr>
<tr>
<td>create_calendar</td>
<td>Create calendar records on specific subtype</td>
</tr>
<tr>
<td>c_cal_with_ssearch</td>
<td>Create calendars on records that satisfy the saved search</td>
</tr>
<tr>
<td>view_own_calendar</td>
<td>View own calendar entries</td>
</tr>
<tr>
<td>v_own_cal_with_ssearch</td>
<td>View own entries that satisfy the saved search</td>
</tr>
<tr>
<td>view_others_calendar</td>
<td>View other peoples’ calendar entries</td>
</tr>
<tr>
<td>others_cal_with_ssearch</td>
<td>View other peoples’ entries that satisfy the saved search</td>
</tr>
<tr>
<td>edit_own_calendar</td>
<td>Edit own calendar entries</td>
</tr>
<tr>
<td>e_own_cal_with_ssearch</td>
<td>Edit own entries that satisfy the saved search</td>
</tr>
<tr>
<td>edit_others_calendar</td>
<td>Edit other peoples’ calendar entries</td>
</tr>
<tr>
<td>e_others_cal_with_ssearch</td>
<td>Edit other peoples’ entries that satisfy the saved search</td>
</tr>
<tr>
<td>delete_own_calendar</td>
<td>Delete own calendar entries</td>
</tr>
<tr>
<td>d_own_cal_with_ssearch</td>
<td>Delete own entries that satisfy the saved search</td>
</tr>
<tr>
<td>delete_others_calendar</td>
<td>Delete other peoples’ calendar entries</td>
</tr>
<tr>
<td>d_others_cal_with_ssearch</td>
<td>Delete other peoples’ entries that satisfy the saved search</td>
</tr>
<tr>
<td><strong>Email Permissions</strong></td>
<td></td>
</tr>
<tr>
<td>view_comm_from</td>
<td>View outgoing email address FROM</td>
</tr>
<tr>
<td>edit_comm_from</td>
<td>Edit outgoing email address FROM</td>
</tr>
<tr>
<td>view_comm_replyto</td>
<td>View send-back email address REPLY-TO</td>
</tr>
<tr>
<td>edit_comm_replyto</td>
<td>Edit send-back email address REPLY-TO</td>
</tr>
<tr>
<td>create_comm</td>
<td>Create a communication record linked to the given subtype</td>
</tr>
</tbody>
</table>
## Email Template Permissions

<table>
<thead>
<tr>
<th>Permission</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mod_own_eml_tmpl</td>
<td>Allow users to modify/delete own email templates</td>
</tr>
<tr>
<td>mod_all_eml_tmpl</td>
<td>Allow users to modify/delete all email templates</td>
</tr>
<tr>
<td>viewsend_own_eml_tmpl</td>
<td>Allow viewing/sending only their own email templates</td>
</tr>
<tr>
<td>viewsend_ownpublished_eml_tmpl</td>
<td>Allow viewing/sending own and published email templates</td>
</tr>
<tr>
<td>viewsend_published_eml_tmpl</td>
<td>Allow viewing/sending own and published email templates</td>
</tr>
<tr>
<td>viewsend_all_eml_tmpl</td>
<td>Allow viewing/sending of all email templates</td>
</tr>
<tr>
<td>publish_eml_tmpl</td>
<td>Allow user to publish email templates</td>
</tr>
</tbody>
</table>

## Print Template Permissions

<table>
<thead>
<tr>
<th>Permission</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cr_mod_own_ptmpl</td>
<td>Create/modify own print templates</td>
</tr>
<tr>
<td>cr_mod_all_ptmpl</td>
<td>Create/modify all print templates</td>
</tr>
<tr>
<td>use_all_ptmpl</td>
<td>Use all print templates</td>
</tr>
<tr>
<td>use_ownpub_ptmpl</td>
<td>Use/own/publish print templates</td>
</tr>
<tr>
<td>can_pub_ptmpl</td>
<td>User can publish print templates</td>
</tr>
</tbody>
</table>
Macros, Methods, and Conditionals

Macros are short, reusable pieces of code that represent a longer set of instructions for the software or browser to run. Macros allow developers to save time and simplify their code. In the Agiloft EUI, macros are shorthand instructions for linking to pages, requesting data from the system, creating table views, and other operations.

Macros are often combined with methods and conditional statements, or conditionals, to control the visibility of a macro’s content. Methods typically function as the actual condition of a conditional, and the conditional determines what the user sees on the EUI. For example, consider two macros that each display an image. A conditional might tell the system to use one macro to show image A if the user is in group A and use the other macro to display image B if the user is in group B. In this case, the method checks the user’s group, and the conditional is the entire statement that uses the user’s group membership to determine which image is displayed.

Understanding how macros, methods, and conditionals work in the EUI is helpful if you plan to make any customizations to the interface. Each of these elements is an essential part to how the EUI functions, so knowing how they work can help you create an EUI that suits your needs.

Macros

Agiloft uses the Velocity Template Language (VTL) template engine to process macros, which uses a special syntax:

```
#macro_name("$param1" "$param2" "$param3" ...)
```

Macros begin with a "#" sign followed by the name of the macro, and many macros also require one or more parameters, which function as additional instructions. Parameters immediately follow the macro name and are enclosed in parentheses. Parameters are each further enclosed in single or double quotes, depending on context. Although the actual name of a parameter includes a dollar sign, in practice the parameter does not typically use a dollar sign when it’s replaced by a value.
Macros with Multiple Parameters

When a macro uses multiple parameters, the order of the parameters is important. The system uses the parameter order to determine which parameter you're setting. Even if you're not setting a specific parameter, or if you're leaving it at its default value, you need to leave a space for it in the code so that you retain the correct parameter order. Parameters that aren't set or that are left at their default value are indicated with a pair of empty quotation marks.

In practice, a macro with several parameters may look like this:

```html
#ew_table("contract" "" $defaultsearch "")
"showToolBar=false&showNavigation=true&showfastsearch=false" "my_table" ")
```

This macro contains seven parameters. One parameter, $defaultsearch, is a special variable parameter that references a saved search and uses a dollar sign but doesn't require quotation marks. See Using Default Searches for more information on how this parameter works. You can find the other six parameters by counting the pairs of quotation marks. Notice that three parameters are empty, meaning that they use a default value. For more information on this macro and its parameters, as well as other macros, see EUI Macro Reference.

Macros and Page Types

In the default EUI setup, each of the basic page types corresponds to an Agiloft macro that uses parameters to reference the table, record, or form. The macros can also apply search filters or set additional parameters.

- Table views, such as the My Support Cases or All Documents pages, use the #ew_table macro, which creates an iframe and displays an Agiloft table inside it.
- Record editing pages, such as the My Profile page, use the #ew_edit_record macro.
- Submission forms, such as the Create a Support Case page, use the #ew_create_record macro.
- FAQ pages, such as the Document / FAQs page, use the #ew_faq macro.

For more information on these macros, see EUI Macro Reference.
Conditionals and Methods

In addition to specialized macros, the Velocity Template Language (VTL) also has prebuilt directives for creating conditional statements, or conditionals. You can use conditionals within the body of an EUI template to show or hide HTML elements and macros. Conditionals are typically combined with methods, which are ways of checking the conditions, making decisions, and retrieving information. Our discussion only covers the basics, so reference the VTL user manual if you need more information.

Using Conditionals

Before including methods, let's first consider conditionals on their own:

```
#if(condition A is true)
<p>Welcome!</p>
#end
```

This conditional displays the text "Welcome!" only if the condition is true; otherwise, nothing happens. Conditionals start with the `#if` directive and end with the `#end` directive, like in this example. However, they can also be combined with `#else` and `#elseif` directives, as well as logical operators, to create more complex conditionals. For instance, consider this example:

```
#if([condition A is true] && [condition B is true])
<p>Welcome!</p>
#else
<p>Hello!</p>
#end
```

This conditional displays the text "Welcome!" if both the conditions are true. Otherwise, it displays the text "Hello!" Similarly, consider this example:

```
#if([condition A is true] || [condition B is true])
<p>Welcome!</p>
#else
<p>Hello!</p>
#end
```

This conditional displays the same "Welcome!" text, but this time the text is displayed if either condition is true. If neither condition is true, it displays the text "Hello!"
Logical Operators

These are the logical operators you can use when comparing conditions within a conditional:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>&amp;&amp;</td>
<td>And</td>
</tr>
<tr>
<td>!</td>
<td>Not</td>
</tr>
<tr>
<td>&lt;</td>
<td>Less than</td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater than</td>
</tr>
<tr>
<td>==</td>
<td>Is equal to</td>
</tr>
</tbody>
</table>

Using Methods

Methods often function as the actual condition of a conditional. For instance, some common methods check a user's group membership, team membership, and permission settings, and the conditional uses the result to determine what the user sees. Methods begin with a dollar sign and, like macros, contain one or more parameters. Parameters immediately follow the method name and are enclosed in parentheses, and each parameter is further enclosed in double quotes and separated by commas. For instance, a method might look like this:

```$ewUser.isInGroup("group name")```

This method checks the current user's group membership to see if they're a member of the specified group. Notice that this method only has a single parameter, which contains the group name. When a method is used with a conditional and a macro, it might look like this:

```#if ($ewUser.isInGroup("Contract Creator") || $ewUser.isInGroup("Contract Manager"))
<img src="#ew_image('/gif/contract-team-logo.png')" />
#else
<img src="#ew_image('/gif/main-logo.png')" />
#end```

This conditional uses the method to check if the current user is in the Contract Creator or Contract Manager groups. If they are, the system uses some HTML with a macro to display `contract-team-logo.png`. If the user is not in either of these groups, the system displays `sample-logo.png`.

For a list of common methods used in the EUI, see EUI Method Reference.
EUI Template Reference

The Agiloft EUI is managed by a set of HTML files stored as records in the EUI Templates table. Using HTML, you can redesign the interface in any number of ways to match your branding and web design. This article contains a description for each of the main EUI templates. To see the code for each template, go to the EUI Templates table in your knowledgebase and open the desired template. For more detailed instructions on editing the templates, see Working with Default EUI Templates.

Each page of the EUI has a corresponding record in the EUI Templates table. The Name field shows the full template name including the file extension, such as .htm, .html, .css, .php, or .properties.

<table>
<thead>
<tr>
<th>Edit</th>
<th>ID</th>
<th>Name</th>
<th>Date Updated</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>main.php</td>
<td>Mar 20 2020 06:37</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>new_servicerequest.html</td>
<td>May 03 2019 04:36</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>style.css</td>
<td>Oct 19 2020 23:15</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>footer.htm</td>
<td>Dec 18 2019 11:10</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>header.htm</td>
<td>Dec 14 2018 17:25</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>home.htm</td>
<td>Oct 23 2020 07:03</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>login.vm</td>
<td>Oct 12 2017 07:45</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>menu.htm</td>
<td>Oct 23 2020 06:53</td>
</tr>
</tbody>
</table>

The Body field holds the code, and modifying the code makes changes to the EUI. For this reason, we recommend making a copy before editing any template records so that you can revert to the old code if necessary.

Some EUI templates don't stand alone as separate pages, but are used by other templates or included in other pages. For instance, the templates header.htm, footer.htm and menu.htm are included in every page of the EUI. Here are other examples of these "helper" files:

- **iresize.js**: Contains a piece of JavaScript
- **search-form.htm**: Contains the search block found on every home page
- **meta.htm**: Holds meta data included in every page

### main.php

The standard login pages provided by Agiloft are set to direct a user to main.php when logging into the custom portal. The following wireframe image shows how the elements contained in the main.php page fit together:
The **main.php** file does not directly contain the elements that are displayed to the user on the page. Instead, **main.php** uses the `#ew_include()` **macro** to pull in the contents of three other HTML files:

- `header.htm`
- `home.htm`
- `footer.htm`

`Header.htm` in turn includes the file `menu.htm`, and `home.htm` pulls in `search-form.htm`. `Header.htm` also contains the reference to the logo, plus user information and a logout link in the upper right corner.

Although a login page does have to specify which page a user is taken to, it does not have to be the **main.php** page. Different login pages for the same project could even direct users to different landing pages.

### home.htm

The **home.htm** template file is the main part of the page that users see when they log in to the EUI. It makes heavy use of conditionals to show different tables to different user groups. An HTML table lines up the Agiloft tables in columns and rows. On the EUI, the horizontal and vertical lines are table cells with particular CSS classes designated. Using conditionals with this type of layout provides a relatively compact presentation when there are only a few options for each user.

You do not have to use **home.htm** if it is not suitable for your needs. In **main.php**, you can replace the reference to **home.htm** with a different EUI template to serve as your main home page. See [Working with Home Pages](#) for more information.

### menu.htm

The **menu.htm** template file creates the navigation menu, or the tabs across the top of the screen.
Each tab of the menu is held in an unordered HTML list in `menu.htm`. The template file also contains a script near the bottom that highlights the current tab. If you add any extra tabs to the EUI, ensure that you give each new item a unique ID in the script.

The area used by the tabs expands to accommodate any extra tabs added. If the area used by the tabs becomes larger than the area available to the menu, it creates a second line, which generally looks bad. For this reason, we recommend that you limit the number of tabs you add, use shorter tab names, or add line breaks into the tab names so that users see a limited number of tabs. You can also create drop-down menus for the extra items. For more information on how to do this, see the EUI Setup Tips page.

**header.htm**

The `header.htm` template file controls the appearance of the EUI header and also references `menu.htm` to display the navigation menu. `Header.htm` contains the reference to the logo, plus user information and a logout link in the upper right corner. The `$ewText.get` method uses the `userinfo.name.label` and `userinfo.logout.link` variables to retrieve the words "User:" and "Logout," respectively, from the `translation.properties` file.

**translation.properties**

The `translation.properties` file is actually part of a family of files. The default Agiloft setup includes `translation.properties`, `translation_ru.properties`, and `translation_zh.properties`. These records hold the text retrieved by the `$ewText.get` method for English, Russian, and Chinese, respectively. For more information about `translation.properties`, localization, and the EUI, see Text and Localization in the EUI.

**style.css**

The `style.css` file contains the CSS code for the EUI, which defines the page formatting and can be updated as desired. By default, `style.css` is already referenced on each page by including the `meta.htm` file, which calls `style.css` along with several other standard scripts. You can use this same shortcut on pages you create.

⚠️ When editing the CSS, keep in mind that by default the `<divs>` in `home.htm` have their location determined by the HTML table and do not use relative positioning. This means that the location of a particular `<div>` on the page is determined by its placement within the HTML table. If you want to update the `style.css` to use positioning, then you must remove the HTML table in `home.htm` or your positioning will not work as intended.
Text and Localization in the EUI

The EUI uses the $ewText.get method, along with key and value pairs in translation.properties, to display text on the interface. When the $ewText.get method contains a particular key, the key's corresponding value is inserted wherever that key appears. This allows the EUI to provide support for multiple languages and facilitate localization. By using these key and value pairs, a full translation of the EUI can be created by simply copying the translation.properties file and translating the value for each key.

Displaying text in the EUI with key and value pairs also creates consistency and allows you to easily make changes. If you need to make a terminology change, it's easier when the text values are contained in a single file that each page can reference. For example, if you decide to change the name of a table from Support Cases to Support Tickets, you can go to translation.properties and make that change in one place to propagate the change anywhere the table name appears.

Prerequisites

Before making changes to the EUI, we recommend that you have a basic understanding of the following topics:

- HTML
- CSS
- Macros, methods, and conditionals, as they’re used in the EUI

For more information on HTML and CSS, visit W3 Schools.

Working with Text

If you want to change text on the EUI, you need to edit the key to which the text corresponds in translation.properties. Although it is possible to enter text directly on an EUI page, we recommend using the key-value pairs in translation.properties for consistency across the EUI templates, as well as to facilitate any future localization efforts.

To change text on the interface:

1. Open translation.properties and search for the text you want to change.
   
   Key-value pairs are organized by function and then by table.

2. Make the change to the text. Be careful not to remove the equals sign (=) or the text that precedes it.
3. Click Save.
4. Open the corresponding EUI page and confirm that the text is updated. Remember, if the key to which the text corresponds is used in more than one place, the updated text will appear in the EUI in more than one place too.

Localization with ewText.get and translation.properties

The purpose of the `ewText.get()` method and the `translation.properties` file is to allow for a multilingual environment. This is possible because `translation.properties` is actually a part of a family of files, with the default setup also including `translation_ru.properties` and `translation_zh.properties`. These latter two files hold the text for Russian and Chinese, respectively, and `translation.properties` holds the text for English.

When the system sets the locale for each user session, the `ewText.get` method checks the session locale and uses that to determine which `translation.properties` file to use. For instance, consider this key and text value pair from `translation.properties`:

```plaintext
home.hc.my.link=My
```

The key is `home.hc.my.link`, which is followed by an equals sign (`=`), which is then followed by the text value "My Cases." Whenever `ewText.get("home.hc.my.link")` appears in an EUI template, "My Cases" is displayed in the EUI when English is the locale. However, if the locale is Russian or Chinese, the value displayed in the EUI is whatever translation of "My Cases" is in `translation_ru.properties` or `translation_zh.properties`, respectively.

⚠️ If you add other languages to the system, the corresponding language file must be named `translation_XX.properties`, where XX is a proper Java locale code. You can find a list of supported Java locale codes [here](#). For more information on the localization process, see [Localization](#).

Set the Session Locale

There are a few common ways to set the session locale:

- Add `lang=XX` to the login URL.
- Use the `#ew_languages` macro in `main.php` to display a list of available languages for the user.
- Use the `#ew_locale` macro in `main.php` to set the locale.

If a locale is not set, the default `translation.properties` file is used, which is typically English but could in principle be other languages. For more information on Agiloft's localization-related macros, see [EUI Macro Reference](#).