Oracle® Communications Convergence

Customization Guide

Release 2

July 2015

ORACLE

Oracle Communications Convergence Customization Guide, Release 2

Contents

Chapter 1. Adding a New Language

Adding a New Language

> **Note**
> To add new languages in Convergence 1.x, see Adding a New Language in Convergence 1.x.

You can customize Convergence to make it available in multiple languages in addition to those supported by default. Moreover, any individual domain can be customized to display a particular language to users in that domain.

This article includes the following topics:

Overview of Tasks

To add a new language in Convergence, you must perform these tasks:

- Add your own resources for the new language. The resource file contains the localized text for labels, names, and other text that appears in the UI.
- Enable end users to select the language by adding it to the drop-down list of languages displayed in the Global Options menu.

To create custom l10n (i18n) resources, you must use the dojo i18n directory infrastructure. Place your custom resource file in the `iwc_static/c11n/<domain>/nls` subdirectory. *Domain* is the domain where the customized languages will be available. Use the `allDomain` directory to apply to all domains in your deployment.

To Add a New Language in Convergence

The following steps outline how to customize the UI to support and display a new language:

1. Enable customization in the Convergence Server.

Before you can customize the UI, the Convergence Server must be enabled for customization. Use the command-line utility, `iwcadmin`, to set the `client.enablecustomization` parameter to `true`. For example:

```
./iwcadmin -W <password file> -o client.enablecustomization -v true
```

2. Create the customization directory, `/c11n`.

All customizations must be located under this directory.

```
/iwc_static/c11n
```

3. Edit the configuration file, `config.js`, by setting the `i18nEnabled` flag to `true`:

```
/c11n/config.js
```

You can create this file by copying an example of the `config.js` file from the sample customization directory:

```
/iwc_static/c11n_sample
```

In the `config.js` configuration file, set the `i18nEnabled` flag to `true`.

4. Create an `nls`/*new language* subdirectory for the l10n module:

```
iwc_static/c11n/<domain>/nls/<new language>
```

where *domain* is the name of the domain in which the new language will be available

and *new language* is the subdirectory in which the new language's resource file is located. For example, to add Vietnamese as a new language, you could create this path:

```
iwc_static/c11n/allDomain/nls/vi
```

5. Create a default resource file named `resources.js`.

```
iwc_static/c11n/<domain>/nls/<new language>/resources.js
```

For example, to add an l10n resource file that adds Vietnamese to all domains, you could create this file:

```
iwc_static/c11n/allDomain/nls/vi/resources.js
```

Since a new language requires the entire resource file translation, you can copy the Convergence English resource file (iwc_static/js/iwc/i18n/nls/resources.js) to begin with. Note that each language added to each domain would have its own l10n resources file (`resources.js`). Depending on the end-user's language locale, Convergence loads that locale's resources file.

Note: the directory structure should be as shown in the following example. In this example, the language "vi" is added to the `allDomain` directory. (Thus, it applies to all domains.)

```
iwc_static/c11n/allDomain/nls/
iwc_static/c11n/allDomain/nls/resources.js (Default resources.js file
for all customized languages in this domain.)
iwc_static/c11n/allDomain/nls/vi/
iwc_static/c11n/allDomain/nls/vi/resources.js  (This resources.js file
contains the localization required for the newly added language, "vi".)
```

You can also create this file by copying an example of the `resources.js` file from the sample customization directory, `/iwc_static/c11n_sample`.

Workaround to Add a New Language that Does Not Currently Exist in the Dojo Toolkit

Because Convergence uses dojo resources for language strings, the language has to be manually added to Convergence if the language is not supported in dojo. The the list of supported dojo languages is in the following directory: `iwc_static/js/dojotoolkit/dojo/nls`.

Use this procedure to add a new language to Convergence that either does not currently exist in the Dojo toolkit or is not complete, as is the case with calendar data formats for Vietnamese (vi). You follow the instructions in To Add a New Language in Convergence but with the following additions.

1. To customize the localized calendar, refer to {{ iwc_static/js/dojotoolkit/dojo/cldr/nls/gregorian.js}} as the base.
2. Copy the English language version from `iwc_static/js/dojotoolkit/dojo/cldr/nls/en/gregorian.js` to `nls/vi/gregorian.js`.
3. Translate the nls/vi/gregorian.js to the localized language (Vietnamese in this example).
4. Make sure you also have a `resources.js file` in the `nls` directory (see bug 6745757). For example:

   ```
   {
       last: ""
   }
   ```

5. Make sure the above file is called from */docroot* `/iwc_static/c11n/allDomain/js/customize.js`, for example, at the end:

   ```
   // adding new language will need localization of dojo calendar
   string
   dojo["requireLocalization"]("c11n.allDomain", "gregorian");
   dojo.date.locale.addCustomFormats("c11n.allDomain","gregorian");
   ```

6. Restart GlassFish Server and clear the browser cache to view the change.
7. Then log in to Convergence.

Sample Custom l10n Resource File

The following example shows a sample `resources.js` file which shows a few labels localized into Vietnamese:

```
{
  compose_tab: "Son th",
  last: ""
}
```

The Convergence i18n service uses dojo l10n modules. For details about customizing languages, consult the dojo l10n documentation.

To Add a Label for the New Language to the Global Options Language Menu

Once you have created a localized version of the resources file, you need to make the new language available to end users. Users can select a language from the Options menu by selecting **Global**, then **General**.

The Global Options - General panel displays the languages supported by default in a drop-down list. You can add the new language to this list by creating and editing the `option.General.js` widget.

Take these steps:

1. Enable customization in the Convergence Server.

Before you can customize the UI, the Convergence Server must be enabled for customization. Use the command-line utility, `iwcadmin`, to set the `client.enablecustomization` parameter to `true`. For example:

```
./iwcadmin -W <password file> -o client.enablecustomization -v true
```

2. Set the `jsEnabled` flag to `true` in the `config.js` file.

```
/c11n/config.js
```

(If you have not already done so, you must create the `/c11n` directory and `config.js` file. For details, see Steps 2 and 3 in CommSuite:To Add a New Language in Convergence.)

3. Create the following subdirectory under the customization (`c11n`) directory:

```
iwc_static/c11n/<domain>/js
```

where *domain* is the domain in which the customized language will be available.

4. Create a `customize.js` file, as follows:

```
iwc_static/c11n/<domain>/js/customize.js
```

You can create this file by copying an example of the `customize.js` file from the sample customization directory, `/iwc_static/c11n_sample`. For example, you could copy the file from

```
iwc_static/c11n_sample/allDomain/js
```

At runtime, this file is responsible for loading the javascript customizations (the custom widget files) in this subdirectory. For details, see How the `customize.js` File Manages Customized Widgets.

5. In the `customize.js` file, uncomment the following line:

```
dojo["require"]("c11n.allDomain.js.widget.option.General");
```

6. Create a subdirectory to contain the custom widgets:

```
iwc_static/c11n/<domain>/js/widget
```

7. Copy the sample version of the `option/General.js` file from the sample customization directory to your live customization directory:

Copy from

```
iwc_static/c11n_sample/allDomain/js/widget/option/General.js
```

to

```
iwc_static/c11n/<domain>/js/widget/option/General.js
```

For example, the following directory structure is created when you customize the `option.General.js` widget in the `allDomain` directory:

```
/c11n/config.js
  /c11n/allDomain/js/customize.js
  /c11n/allDomain/js/widget/option/General.js
```

8. Edit the `option/General.js` file, adding your language.

The sample file adds Vietnamese, as follows:

```
dojo.provide("c11n.allDomain.js.widget.option.General");

dojo["require"]("iwc.widget.option.General");

dojo.declare("iwc.widget.option.General",
 iwc.widget.option.General,
 {
  buildRendering: function() {
   this.inherited(arguments);

   // add your new languages here
   // value: language code - e.g., "en", "zh-TW"
   // label: display name, use UTF-8 for double bytes
   this.language.addOption({value: "vi", label: "Vietnamese"});

  },

  last: ""
 }
);
```

In the line `this.language.addOption`:

- The `value:` identifies the language code. This should be the l10n directory you created for your language. For example, the Vietnamese example uses `vi`.
- The `label:` identifies the display name of the language. This name appears in the drop-down list of languages in the **Global Options - General** panel. For example, "Vietnamese."

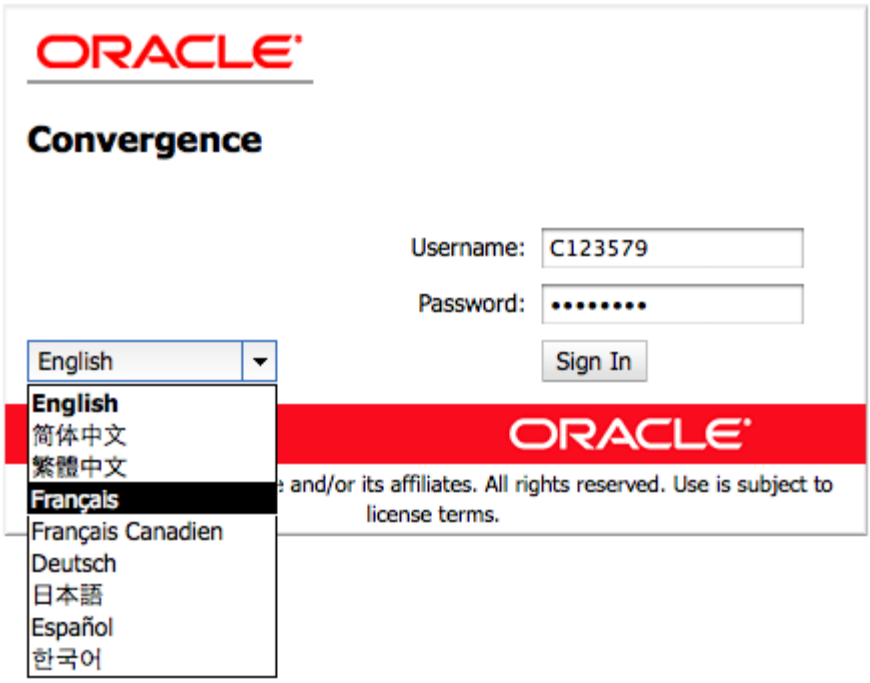
8a. Edit `"vi"` and `"Vietnamese"`, replacing them with your language directory and name. Use UTF-8 for double-byte characters.

(Add additional this.language.addOption lines to add additional languages.)

To Add a Label for the New Language to the Convergence Login Page

The following figure shows the default Convergence login page. The supported languages are listed in the drop-down menu at the bottom of the page.

Figure 1. Convergence Login Page with List of Supported Languages



For this release of Convergence, to add the new language to this list, you must modify the login HTML file, `login.html`.

The `login.html` file is a static component downloaded to the user's browser. Unlike a dojo widget, the HTML file does not dynamically extend the base dojo code. Moreover, `login.html` is one of the layout files in the base code. It cannot be loaded from the `c11n` customization directory.

Therefore, editing the `login.html` file differs significantly from extending a widget to customize the UI.

To add a new label to the login page, review Customizing Layout HTML Pages.

To Add, Remove, and Set Help for Unsupported Locales in the Convergence Banner

If the user's preferred language or the browser's language is not supported in Convergence, clicking the Help link on the Convergence Banner results in a File Not Found (404) error.

Supported languages are: `de, en, es, fr, fr-ca, ja, ko, zh-cn, zh-tw`.

To properly set the locale for the help, be sure to copy the following code from `c11n_sample/allDomain/js/customize.js` file into your customization's `c11n/allDomain/js/customize.js` file:

```
// Locale Help: Set the default help locale
// Uncomment the following code to have non-supported locale use 'es'
instead of
// the default provided 'en'.
/*
iwc.api.setDefaultHelpLocale("es");
*/
//
// Locale Help: Add a new locale help file
// Uncomment the following code to allow locale 'EN_US' to use the help
file
// located at'help/en/toc.html'
/*
iwc.api.addHelpLocale("EN_US", "help/en/toc.html");
*/

// Locale Help: Remove a help locale
// Uncomment the following code to have the locale 'fr-ca' use the
default help
// locale instead.
/*
iwc.api.removeHelpLocale("fr-ca");
*/
```

To Import or Export Address Book Information in a Custom Language

See Knowledge Base Article: How to Export and Import Address Book in Custom Language (Doc ID 1465260.1)

Note
Configuring Convergence for Address Book import in custom language can only be performed beginning with Convergence 2 Patch 4.

Chapter 2. Changing Names and Labels in the Convergence UI

Changing Names and Labels in the Convergence UI

You can change the labels for icons, tabs, menus, and other elements in the Convergence user interface (UI). You create your own names for these UI widgets by changing their definitions in the `resources.js` file.

Moreover, the label or name can be customized in a particular domain or can be applied to all domains. Use the `allDomain` directory to apply to all domains in your deployment.

To Change a Convergence UI Element's Label or Name

The following steps outline how to customize the Convergence UI to display a new label or name for a widget:

1. Enable customization in the Convergence Server.

Before you can customize the UI, the Convergence Server must be enabled for customization. Use the command-line utility, `iwcadmin`, to set the `client.enablecustomization` parameter to `true`. For example:

```
./iwcadmin -W <password file> -o client.enablecustomization -v true
```

2. Create the customization directory, `/c11n`.

All customizations must be located under this directory.

```
/iwc_static/c11n
```

3. Create the configuration file, `config.js`, for the customization.

```
/c11n/config.js
```

You can create this file by copying an example of the `config.js` file from the sample customization directory:

```
/iwc_static/c11n_sample
```

4. In the `config.js` configuration file, set the `i18nEnabled` flag to `true`.

5. Create an `nls` subdirectory for the l10n module:

```
iwc_static/c11n/<domain>/nls
```

where *domain* is the name of the domain for which you want to customize the UI.

At runtime, for each domain that has `i18nEnabled` enabled, Convergence loads that domain's l10n resource file, named `resources.js`. To apply a customization to all domains, use the `allDomain` as your domain name.

6. Create a default resource file named `resources.js`.

```
iwc_static/c11n/<domain>/nls/resources.js
```

7. In the `resources.js` file, add a text string that identifies the UI widget for which you want a customized label, and add the new label in double quotes.

Add a new text line for each widget which will have a custom label.

This customized resources file (under the `c11n` directory) inherits the default values in the standard resources file and extends them with the new, custom values.

At runtime, the `resources.js` file in the default directory is loaded first; then the `resources.js` in this customization directory is loaded. Thus, Convergence first loads the standard values (including UI widget labels) in the default resources file; it then overrides those values with any customizations in this resources file.

8. To change a label in a specific language supported by Convergence, edit the `resources.js` file in the directory for that language.

For example, to provide a customized label in French, add the new text in the `resources.js` file in the subdirectory containing the French version:

```
iwc_static/c11n/<domain>/nls/fr/resources.js
```

The following section shows an example of a custom label.

Sample Customized Label in the Resource File

The following example shows a sample `resources.js` file which identifies a new label for the "Compose" tab in the mail service. This example also shows a new theme with its new icon to be displayed.

```
{
  compose_tab: "New Mail Message",
  prius_green_theme: "Prius Green"
}
```

The following figures show examples of custom labels for the Compose tab in Mail. The label **New Mail Message** has replaced the default label, **[No Subject]**.

Figure 1: Convergence 1.x Compose Tab with a Customized Label ("New Mail Message")

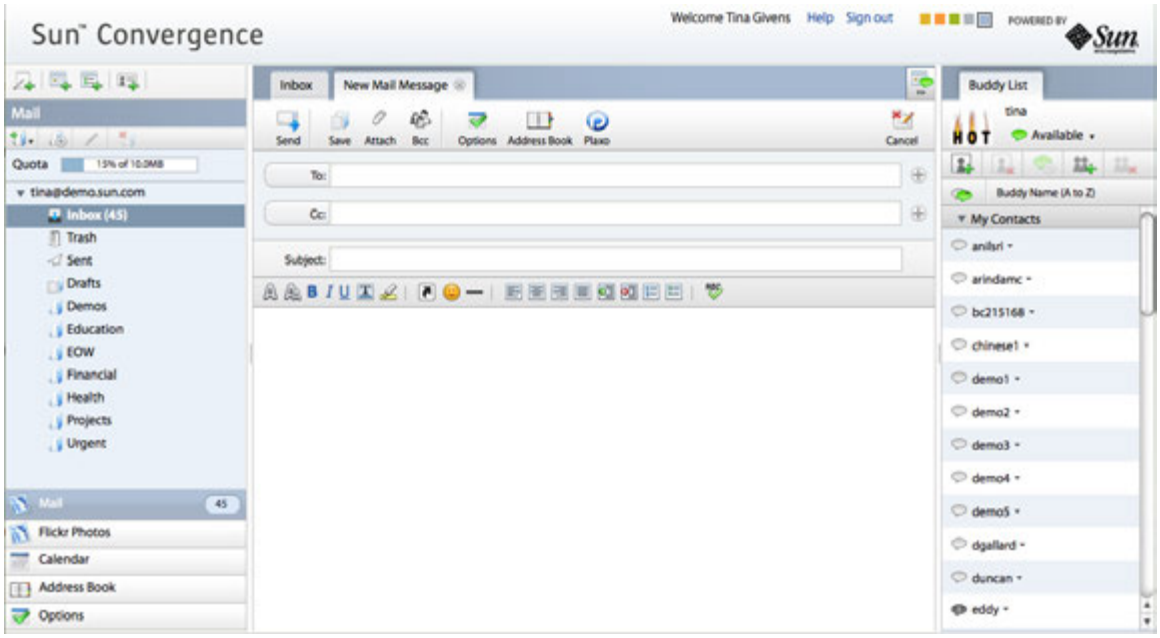
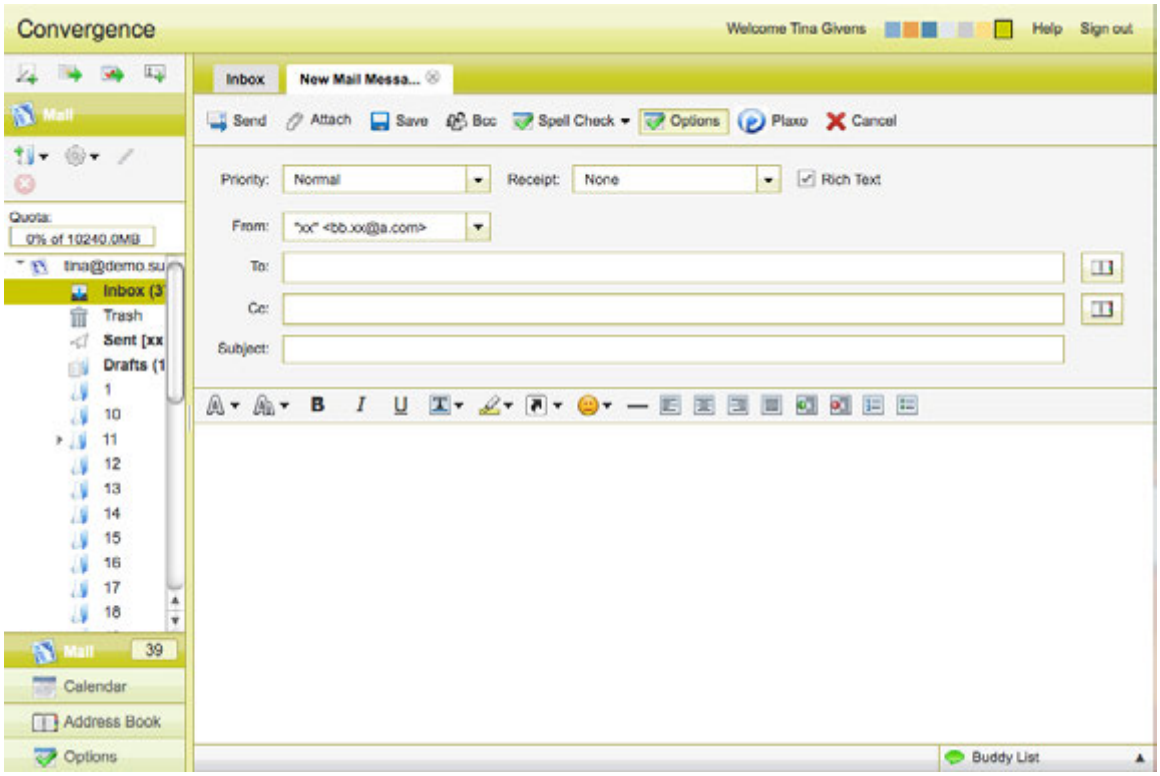


Figure 2: Convergence 2 Compose Tab with a Customized Label ("New Mail Message")



The Convergence i18n service uses dojo l10n modules. For details about customizing languages, consult the dojo l10n documentation.

Sample Directory Structure for i18n

The following example illustrates the directory structure of the `allDomain` directory:

```
/c11n/config.js
  /c11n/allDomain/nls/resources.js
  /c11n/allDomain/nls/<lang>/resources.js
```

In the preceding example, the variable <*lang*> is one of the supported languages.

Chapter 3. Adding Additional Spell Check Dictionaries

Customization Example - Adding Additional Spell Check Dictionaries

This Convergence example assumes that the reader has thoroughly read the Convergence customization documentation, Convergence Customization Guide.

Comms Community Verified Contribution
This article came from the Comms Community and has been verified by the Oracle Communications Unified Communications Suite product team.

Note
To add additional spell check dictionaries in Convergence 1.x, see Customization Example - Adding Additional Spell Check Dictionaries in Convergence 1.x.

When a user clicks on the spell-check button in the Convergence email compose window, the email data to be checked is sent to the configured Messaging Server system, which then scans and detects incorrectly spelt words and offer alternate suggestions back to Convergence.

The user can then select between these alternate suggestions and fix incorrectly spelt words that were detected.

The user is also able to select between several pre-configured Messaging Server dictionaries (English, Spanish, German, French). Additional dictionaries can be added in a two step process.

- Generating a new **ispell** formatted dictionary hash which is used by Messaging Server to spell check new emails.
- Modifying the existing Convergence Spell Check option to include the new dictionary as an option for the end-user.

Although it is possible to create **ispell** formatted dictionary files from scratch, the process is very time-consuming and is not-recommended if a pre-existing dictionary can be found and modified as required.

The following external website has a list of a number of common language **ispell** formatted dictionaries which can be used as a template for your own custom dictionary:

http://fmg-www.cs.ucla.edu/fmg-members/geoff/ispell-dictionaries.html

For this example we will add an Italian (**it**) dictionary to Messaging Server and Convergence.

1. Download the Italian **ispell** dictionary and affix file archive (ispell-it2001.tgz) from the following website
   http://members.xoom.it/trasforma/ispell
   The required files from this archive are *italian.aff* and *italian.words*
2. Generate the Messaging Server **ispell** dictionary hash file

Note
The **ispell** dictionary hash needs to be created on the system configured in the following Convergence option:
./iwcadmin -o mail.host

```
cd <msg_svr_base>/lib
./buildhash <dictionary_file> <affix_file> <language_name>.hash

e.g.

./buildhash /tmp/italian.words /tmp/italian.aff ./it.hash

Verify that the newly created <language_name>.hash file has the
same permissions and ownership as the existing
 <language_name>.hash files in the <msg_svr_base>/lib directory.
```

3. Add new language code to the supported languages list in Messaging Server
   Run the following command to see the existing list of supported languages.

```
./configutil -o local.supportedlanguages
```

   If you don't see the <language_name> listed in the above output, add it to the list e.g.

```
./configutil -o local.supportedlanguages -v "[............,it]"
```

4. Restart the Messaging Server mshttpd process

```
./stop-msg http;./start-msg http
```

5. Enable Convergence client customizations

```
./iwcadmin -o client.enablecustomization -v true
```

6. Create the following directory structure under <appserver domain root>/docroot/iwc_static/ to hold the required customization files.

```
c11n/
c11n/allDomain
c11n/allDomain/js
c11n/allDomain/js/widget
c11n/allDomain/nls
```

7. Generate the general customization configuration file.
   This configuration file enables customizations (jsEnabled: true) to the JavaScript code that generates the list of dictionaries that can be used by the Spell Check functionality and is applied to all user domain (allDomain:):

c11n/config.js

```
dojo.provide("c11n.config");

c11n.config = {

      // allDomain configuration
      allDomain: {
      module: "allDomain", // module name
      themeEnabled: false, // true if theme is customized
      i18nEnabled: true, // i18n must be customized to display the
localized label
      jsEnabled: true // true if js is customized

      // the last entry must not end with comma
      }
}
```

8. Create a `resources.js` file with the content shown below to define the localized label for the "Italian" dictionary.

c11n/allDomain/nls/resources.js

```
{
        options_global_it : "Italian",  // Added Italian ("it") to
the list of available dictionaries.
                                        // The "it" language code
must match the {{it.hash}} file added to Messaging Server.
        last: ""
}
```

9. (Optional) To make the label different for each locale, you can take these steps:
   a. Add a localization directory for the locale. For example:

```
c11n/allDomain/nls/it
```

   b. Add another `resources.js` file in the specified locale to display the label. For example:

c11n/allDomain/nls/it/resources.js

```
{
        options_global_it : "Italiano",  // Added Italian
("it") to the list of available dictionaries.
                                            // The "it" language
code must match the {{it.hash}} file added to Messaging
Server.
        last: ""
}
```

10. Restart GlassFish Server (formerly Application Server) and clear the browser cache to see the change.

Chapter 4. Consolidating Convergence Customizations to Preserve Client Performance

Consolidating Convergence Customizations to Preserve Client Performance

This Convergence example assumes that the reader has thoroughly read the Convergence Customization Guide.

Note
To consolidate customizations in order to preserve client performance in Convergence 1.x, see Consolidating Convergence 1.x Customizations to Preserve Client Performance.

Comms Community Verified Contribution
This article came from the Comms Community and has been verified by the Oracle Communications Unified Communications Suite product team.

If you create a lot of dojo customizations you will notice that the browser will need to download all of your customization files individually. Convergence already requires a lot of files to be downloaded by the browser, so increasing the number of files by another 10 or 20 or so will only make the initial load time worse.

Instead of creating the customizations as inidividual files, you can add the customizations directly to `customize.js`. This means that you can have an unlimited number of customizations, but still only one file will be required.

For example, instead of:

c11n/allDomain/js/widget/mail/option/VacationMessage.js

```
dojo.provide("c11n.allDomain.js.widget.mail.option.VacationMessage");
dojo.require("iwc.widget.mail.option.VacationMessage");
dojo.declare("iwc.widget.mail.option.VacationMessage",
iwc.widget.mail.option.VacationMessage, {
    constructor: function() {
        // your customizations to this widget
    },
    last: ""
});
```

and

c11n/allDomain/js/customize.js

```
dojo.require("c11n.allDomain.js.widget.mail.option.VacationMessage");
```

you can consolidate it all into one file. Furthermore, pobrien78 correctly points out that you can get rid of the last dojo.require() line:

c11n/allDomain/js/customize.js

```
dojo.provide("c11n.allDomain.js.widget.mail.option.VacationMessage");
dojo.require("iwc.widget.mail.option.VacationMessage");
dojo.declare("iwc.widget.mail.option.VacationMessage",
iwc.widget.mail.option.VacationMessage, {
    constructor: function() {
        // your customizations to this widget
    last: ""
});
```

Finally, you might want to consider running this file through a JavaScript compressor such as ShrinkSafe to reduce the size, and therefore latency, of the JavaScript by browsers.

Chapter 5. Customization Example - Categorizing Calendar Events By Using Different Text Colors or Background Colors

Customization Example - Categorizing Calendar Events By Using Different Text Colors or Background Colors

This Convergence customization example describes how to differentiate calendar events with text colors or background colors to represent different event categories. For example, you can designate specific text colors or background colors in an event to indicate when an event is a business meeting, a conference call, or a vacation category. This example assumes that users only have one calendar.

1. Enable client customizations.

```
./iwcadmin -o client.enablecustomization -v true
```

2. Create the following directory structure under the `appserver-domain-root/docroot/iwc_static/` directory to hold the customization files.

```
c11n/
c11n/allDomain
c11n/allDomain/js
c11n/allDomain/js/widget
c11n/allDomain/js/widget/calendar
```

3. Generate the general customization configuration file.
   This configuration file enables JavaScript customization (`jsEnabled: true`) across all domains:

c11n/config.js (module: "allDomain")

```
dojo.provide("c11n.config");

c11n.config = {

        // allDomain configuration
        allDomain: {
                module: "allDomain",           // module name
                themeEnabled: false,           // true if theme is
customized
                i18nEnabled: false,            // true if i18n is
customized
                jsEnabled: true                // true if js is
customized

                // the last entry must not end with comma

        }
}
```

4. Create and add to the domain-specific customization file.

c11n/allDomain/js/customize.js

```
dojo.provide("c11n.allDomain.js.customize");

dojo.require("c11n.allDomain.js.widget.calendar.Event");
```

- Uncomment the following section in the `c11n/allDomain/js/customize.js` file.

c11n/allDomain/js/customize.js

```
var loadCustomizedCssFile = function(url, id) {
        if (!id){
                id = url.replace(/../gm, "").replace(///gm,
"_").replace(/./gm, "_");
        }
        var link = window.document.getElementById(id);
        if (! link) {
                link = window.document.createElement("link");
                link.setAttribute("id", id);
                link.setAttribute("type", "text/css");
                link.setAttribute("rel", "stylesheet");
                var head =
window.document.getElementsByTagName("head")[0];
                head.appendChild(link);
        }

        link.setAttribute("href", url + "?" +
djConfig.cacheBust);
}
loadCustomizedCssFile("../c11n/allDomain/layout/css/c11n.css")
```

5. Add the following code to the `c11n/allDomain/js/widget/calendar/Event.js` file. In this example, red text events are designated as business events, blue text events are designated as vacation events, and purple text events are designated as conference calls:

c11n/allDomain/js/widget/calendar/Event.js

```
dojo.provide("c11n.allDomain.js.widget.calendar.Event");

dojo["require"]("iwc.widget.calendar.Event");

dojo.declare("iwc.widget.calendar.Event",
iwc.widget.calendar.Event, {
        categoryColor: {
                "business": "redCategory",
                "vacation": "blueCategory",
                "conference call": "purpleCategory"
        },

        initDisplay: function() {
                this.inherited(arguments);

                var s = this.calendarService;
                try {
                        // category can be string or array,
depending on save to server or get from server
                        var category = s.getValue(this.item,
"CATEGORIES");
                        if (dojo.isArray(category)) category =
category[0];
                        category = category.toLowerCase();
                        var color = this.categoryColor[category];
                        if (color) {
                                dojo.addClass(this.domNode, color);
                        }
                } catch(e) {};
        },

        last: ""

});
```

6. Edit `c11n/allDomain/layout/css/c11n.css` by adding the following code to define each category with text color or background color.

> **Note**
> To change background color, refer to the Convergence event colors which are located in the `iwc_static/layout/images/calendar` directory.

To change text color:

c11n/allDomain/layout/css/c11n.css

```
.redCategory {
        color: red;
}
```

To change background color:

c11n/allDomain/layout/css/c11n.css

```
.redCategory {
        background-color: #e72d20;
}

.CalendarViewerContainer .CalendarEvent.redCategory
.CalendarEvent-container
{
        background-image:
url("../../../../layout/images/calendar/eventBoxTRRed.png");
}

.CalendarViewerContainer .CalendarEvent.redCategory
.CalendarEvent-header
{
        background-image:
url("../../../../layout/images/calendar/eventBoxTLRed.png");
        background-color: #e72d20;
}

.CalendarViewerContainer .CalendarEvent.redCategory
.CalendarEvent-body
{
        background-color: #e75248;
        border-color: #e72d20;
}

.CalendarViewerContainer .CalendarEvent.redCategory
.CalendarEvent-footer
{
        background-image:
url("../../../../layout/images/calendar/eventBoxBLRed.png");
}

.CalendarViewerContainer .CalendarEvent.redCategory
.CalendarEvent-footer .CalendarEvent-footerText
{
        background-image:
url("../../../../layout/images/calendar/eventBoxBarRed.png");
}
```

7. Restart GlassFish Server and clear the browser cache to see the change.

Chapter 6. Customization Example - Disabling External POP Account Access

Customization Example - Disabling External POP Account Access

> **Note**
> This Convergence example assumes that the reader has thoroughly read the Convergence Customization Guide.

You can use this sample code to remove the Change Password option, Remove Mail Vacation Message option, remove Calendar Notifications option, and remove External Accounts option.

> **Note**
> If you don't want to disable external POP account access, but you want to control what appears in the "From" pull-down menu for addressing, a customization example has been added to only allow outgoing mail to use the parameters in the local account identity. See: Customization Example - Restricting Outgoing Mail by Only Allowing Local Account Identity Parameters.

1. Enable client customizations.

```
./iwcadmin -o client.enablecustomization -v true
```

2. Create the following directory structure under `<appserver domain root>/docroot/iwc_static/` to hold the customization files.

```
c11n/
c11n/allDomain
c11n/allDomain/js
c11n/allDomain/js/widget
c11n/allDomain/js/widget/option
```

3. Generate the general customization configuration file.
   This configuration file enables JavaScript customization (`jsEnabled: true`) across all domains (`module: "allDomain"`).

c11n/config.js

```
dojo.provide("c11n.config");

c11n.config = {

        // allDomain configuration
        allDomain: {
                module: "allDomain",           // module name
                themeEnabled: false,           // true if theme is
customized
                i18nEnabled: false,            // true if i18n is
customized
                jsEnabled: true                // true if js is
customized

                // the last entry must not end with comma

        }
}
```

4. Create the domain specific customization file (if it does not already exist).

c11n/allDomain/js/customize.js

```
dojo.provide("c11n.allDomain.js.customize");

dojo.require("c11n.allDomain.js.widget.option.Navigator");
```

5. To remove the External Accounts, edit the `iwc.api.removeOption` in `Navigator.js` JavaScript file:

c11n/allDomain/js/widget/option/Navigator.js

```
dojo.provide("c11n.allDomain.js.widget.option.Navigator");

dojo["require"]("iwc.api");
dojo["require"]("iwc.widget.option.Navigator");

dojo.declare("iwc.widget.option.Navigator",
iwc.widget.option.Navigator,
        {
                postCreate: function() {
                        // call the superclass postCreate
                        this.inherited(arguments);

                        // remove the change password option
                        iwc.api.removeOption("/Global/Change
Password");

                        // remove the Mail vacation message option
                        iwc.api.removeOption("/Mail/Local
Account/Vacation Message");

                        // remove the Calendar Notifications option


iwc.api.removeOption("/Calendar/Notifications");

                        // remove the External Accounts
                        iwc.api.removeOption("/Mail/External
Accounts");

                },

                last: ""
        }
);
```

6. Restart GlassFish Server and clear the browser cache to see the change.

Chapter 7. Convergence Customization Index

Convergence Customization Index

The following dynamically generates a list of Convergence (2.x and later) customization pages on this wiki:

Page: Removing the Language Selection Pull-Down Menu and Removing Languages from the Options Page
Page: Removing the Move Button in the Mail Open Folder
Page: Removing the SMS option from the Calendar Notification Options and the New Event Reminder Dialog
Page: ThemeJsonReference
Page: Using the Convergence Customization Example
Page: Writing a Custom Authentication Module Beginning with Convergence 3.0.0.0.0
Page: Writing a Custom Authentication Module for Convergence
Page: Writing a Pluggable SSO Module for Convergence

Chapter 8. Customization Example - Adding a Logo to the Right Side of the Banner

Customization Example - Adding a Logo to the Right Side of the Banner

While there are a number of theme and banner customization scenarios outlined in Customizing Themes and Banner in Convergence 2.x, this example explains describes how to add an Oracle logo to the right side of the banner.

As with other Convergence elements, you can customize the banner in a particular domain, or it can be applied to all domains. The following example uses the `allDomain` directory to apply the logo to the right side of the banner in all domains of the deployment:

1. Enable customization in the Convergence Server.
   Before you can customize the UI, the Convergence Server must be enabled for customization. Use the command-line utility, `iwcadmin`, to set the `client.enablecustomization` parameter to `true`. For example:

   ```
   ./iwcadmin -W <password file> -o client.enablecustomization -v true
   ```

2. Create the following directory structure under the *appserver-domain-root* `/docroot/iwc_static/` directory to hold the customization files.

   ```
   c11n/
   c11n/allDomain
   c11n/allDomain/js
   c11n/allDomain/js/widget
   ```

3. Create the general customization configuration file, `config.js`. (You can copy this file from the `c11n_sample` directory.)
4. In the `config.js` configuration file, set the `jsEnabled` flag to `true` across all domains (`"allDomain"`). For example:

c11n/config.js

```
dojo.provide("c11n.config");

c11n.config = {

        // allDomain configuration
        allDomain: {
                module: "allDomain",            // module name
                themeEnabled: false,            // true if theme is
customized
                i18nEnabled: false,             // true if i18n is
customized
                jsEnabled: true                 // true if js is
customized

                // the last entry must not end with comma

        }
}
```

5. Create the domain specific customization file.

c11n/allDomain/js/customize.js

```
dojo.provide("c11n.allDomain.js.customize");

//Enable the Banner.js to add the logo to the right of the banner.
dojo.require("c11n.allDomain.js.widget.Banner");
```

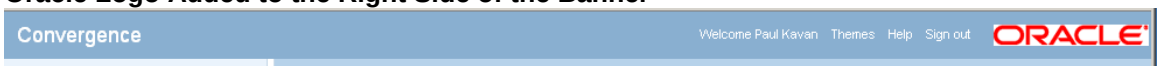
6. Create the JavaScript file that adds the Oracle logo (`smallOracleLogo.gif`) to the right side of the banner. When doing so, be sure to do the following:
    a. The `float:right` parameter aligns the dom node to the right side of the banner.
    b. Adjust the image height, width and image location to fit the logo in the banner.
        - height: change the style `"height"` from `"40px"` to the height of the logo space.
        - width: change the style `"width"` from `"138px"` to the width of the logo space.
        - image location: change the style `"background"` URL from `'url("../c11n/allDomain/layout/images/smallOracleLogo.gif")'`, to the logo location of the logo you want to add to the banner. The URL should be relative to `main.html`.

c11n/allDomain/js/widget/Banner.js

```
dojo.provide("c11n.allDomain.js.widget.Banner");
dojo["require"]("iwc.widget.Banner");

dojo.declare("iwc.widget.Banner", iwc.widget.Banner,
  {
  // Overwriting iwc.widget.Banner postCreate
  // Purpose: Add additional Dom Node for logo on the
right
  // How to Style: Modify newLogoProperties to provide
inline styles
  postCreate: function(){
    //console.debug(this.id+"::postCreate");
    this.inherited(arguments);

    // new Logo dom node properties
    var newLogoProperties = {
 "style": {
  "float": 'right',
  "height": '40px',
  "width": '138px',
  "background": 'transparent
url("../c11n/allDomain/layout/images/smallOracleLogo.gif")
no-repeat center center'
  }
      };

    // Create new Dom Node before everything else
    var newLogoDomNode = dojo.create("div",
newLogoProperties, this.domNode,"first");
  }
});
```

7. Restart GlassFish Server and clear the browser cache to see the change.

Oracle Logo Added to the Right Side of the Banner



Chapter 9. Customization Example - Adding and Removing Fonts in the Editor Menu

Customization Example - Adding and Removing Fonts in the Editor Menu

> **Note**
> This example needs a minimum patch level of Convergence 2 Patch 3.

The following example describes how to add and remove fonts in the Editor menu.

1. Enable client customizations.

```
./iwcadmin -o client.enablecustomization -v true
```

2. Create the following directory structure under `<appserver domain root>/docroot/iwc_static/` to hold the customization files.

```
c11n/
c11n/allDomain
c11n/allDomain/js
```

   This configuration file enables customization (`jsEnabled: true`) across all domains (`module: "allDomain"`).

c11n/config.js

```
dojo.provide("c11n.config");

c11n.config = {

        // allDomain configuration
        allDomain: {
                module: "allDomain",           // module name
                themeEnabled: false,           // true if theme is
customized
                i18nEnabled: false,            // true if i18n is
customized
                jsEnabled: true                // true if js is
customized

                // the last entry must not end with comma

        }
}
```

3. Create the domain specific customization file (if it does not already exist).

c11n/allDomain/js/customize.js

```
dojo.provide("c11n.allDomain.js.customize");

dojo.require("c11n.allDomain.js.editor.plugins.FontChoice");
```

4. To add new Japanese fonts to the Editor font pull-down menu, use the following sample:

c11n/allDomain/js/editor/plugins/FontChoice.js

```
dojo.provide("c11n.allDomain.js.editor.plugins.FontChoice");

   dojo["require"]("iwc.editor.plugins.FontChoice");

   dojo.declare("iwc.editor.plugins.FontChoice",
    iwc.editor.plugins.FontChoice,
      {
           custom: {
                    fontName: [
                              "Ms pgothic",          // sans-serif
                              "Arial",               // sans-serif
                              "Comic Sans MS",       // cursive
                              "Courier",             // monospace
                              "Courier New",         // monospace
                              "Georgia",             //
transitional-serif
                              "Helvetica",           // sans-serif
                              "Lucida Console",      // sans-serif
                              "Tahoma",              //sans-serif
                              "Times",               // serif
                              "Times New Roman",     // serif
                              "Trebuchet MS",        // sans-serif
                              "Verdana"              // sans-serif
                               ],
                     fontSize: [1,2,3,4,5,6,7] // sizes according
to the old HTML FONT SIZE
                     },
          last: ""
  });
```

5. Restart GlassFish Server and clear the browser cache to see the change.

Note
To translate the font name "ms pgothic" in the editor font choice pull-down menu, see: Translating the Font Name "ms pgothic" in the Editor Font Choice Pull-down Menu.

Chapter 10. Customization Example - Allowing Users to Edit their Identity Settings

Customization Example - Allowing Users to Edit their Identity Settings

The following example enables the ability for end users to edit name, email and Reply-To: settings for any external identities found in the Options Tab --> Mail --> Local Account --> Identities in the Mail Personal Account Settings.

1. Enable client customizations.

```
./iwcadmin -o client.enablecustomization -v true
```

2. Create the following directory structure under the *appserver-domain-root* `/docroot/iwc_static/` directory to hold the customization files.

```
c11n/
c11n/allDomain
c11n/allDomain/js
c11n/allDomain/js/widget
```

3. Generate the general customization configuration file.
   This configuration file enables JavaScript customization (`jsEnabled: true`) across all domains (`module: "allDomain"`).

c11n/config.js

```
dojo.provide("c11n.config");

c11n.config = {

        // allDomain configuration
        allDomain: {
                module: "allDomain",           // module name
                themeEnabled: false,           // true if theme is
customized
                i18nEnabled: false,            // true if i18n is
customized
                jsEnabled: true                // true if js is
customized

                // the last entry must not end with comma

        }
}
```

4. Create the domain specific customization file (if it doesn't already exist).

c11n/allDomain/js/customize.js

```
dojo.provide("c11n.allDomain.js.customize");

// Enable Identity Settings
dojo.require("c11n.allDomain.js.widget.mail.option.Identity");
```

5. Create the JavaScript file that enables Identity settings and sets user preferences.

c11n/allDomain/js/widget/mail/option/Identity.js

```
dojo.provide("c11n.allDomain.js.widget.mail.option.Identity");
dojo["require"]("iwc.widget.mail.option.Identity");

dojo.declare("iwc.widget.mail.option.Identity",
iwc.widget.mail.option.Identity, {

         postCreate: function () {
                 this.inherited(arguments);
                 this.senderName.setDisabled(false);
                 this.senderEmail.setDisabled(false);
                 this.replyTo.setDisabled(false);

         },

         onSubmit: function() {
                var data = [];

data.push({name:'general.cn',value:this.senderName.attr('value')});
                var deferred = iwc.protocol.iwcp.setUserPrefs(data,
true /* always sync */);
                deferred.addCallback(this, function() {
                     //success
                 });
                this.inherited(arguments);
        },

         last: ""

});
```

6. To create mappings for custom attributes, modify the `custom-useroptions.properties` file in the `/var/opt/sun/comms/iwc/config/` directory by adding `custom.name=cn` to the file.
7. Restart GlassFish Server and clear the browser cache to see the change.
8. After users edit any of their identity settings (edit name, email and Reply-To: settings) for the first time, be sure that they re-login to Convergence in order to enable their changes.

Chapter 11. Customization Example - Changing an Icon in the Service Selector

Customization Example - Changing an Icon in the Service Selector

This example describes how to change the Mail, Address Book, Instant Messaging, or Options icon in the service selector.

Default Service Selector Icons
The default service selector icons are circled in the following screenshot:

As with other Convergence elements, your customization can apply to a particular domain, or it can be applied to all domains. The following example uses the `allDomain` directory to customize the service icons in all domains in the deployment:

1. Enable customization in the Convergence Server.
   Before you can customize the UI, the Convergence Server must be enabled for customization. Use the command-line utility, `iwcadmin`, to set the `client.enablecustomization` parameter to `true`. For example:

   ```
   ./iwcadmin -W <password file> -o client.enablecustomization -v true
   ```

2. Create the following directory structure under the *appserver-domain-root* `/docroot/iwc_static/` directory to hold the customization files.

```
c11n/
c11n/allDomain
c11n/allDomain/js
c11n/allDomain/layout
c11n/allDomain/layout/css
```

3. Create the general customization configuration file, `config.js`. (You can copy this file from the `c11n_sample` directory.)
4. In the `config.js` configuration file, set the `jsEnabled` flag to `true` across all domains (`"allDomain"`). For example:

c11n/config.js

```
dojo.provide("c11n.config");

c11n.config = {

        // allDomain configuration
        allDomain: {
                module: "allDomain",            // module name
                themeEnabled: false,            // true if theme is
customized
                i18nEnabled: false,             // true if i18n is
customized
                jsEnabled: true                 // true if js is
customized

                // the last entry must not end with comma

        }
}
```

5. Create the domain specific customization file.

c11n/allDomain/js/customize.js

```
dojo.provide("c11n.allDomain.js.customize");

// Load Customized CSS File Helper
var loadCustomizedCssFile = function(url, id) {
        if (!id){
                id = url.replace(/\.\./gm, "").replace(/\//gm,
"_").replace(/\./gm, "_");
        }
        var link = window.document.getElementById(id);
        if (! link) {
                link = window.document.createElement("link");
                link.setAttribute("id", id);
                link.setAttribute("type", "text/css");
                link.setAttribute("rel", "stylesheet");
                var head =
window.document.getElementsByTagName("head")[0];
                head.appendChild(link);
        }

        link.setAttribute("href", url + "?" + djConfig.cacheBust);
}

// Load customized css file c11n_icons
loadCustomizedCssFile("../c11n/allDomain/layout/css/c11n_icons.css");
```

6. Add new icons or icon sprite (a single image that contains multiple icons) to the `c11n/allDomain/layout/images/` directory. For example, `new_services_icon_sprite.png` replaces the existing icon sprite:
   Existing Image Sprite
   
   New Image Sprite
   
7. To change all the service selector icons, create a new css file: `clln_icons.css` in the `c11n/allDomain/layout/css/` directory. In this example, the `background-image` points to the new icon sprite in `c11n/allDomain/layout/images/`:

c11n/allDomain/layout/css/c11n_icons.css

```
/* Service Icons */
/* Mail Service Icon */
 .mail .serviceIcon {
  background-image: url("../images/new_services_icons_sprite.png");
  background-repeat: no-repeat;
  background-position: 0px center;
  background-color: transparent;
}

/* Calendar Service Icon */
 .calendar .serviceIcon {
  background-image: url("../images/new_services_icons_sprite.png");
  background-repeat: no-repeat;
  background-position: -60px center;
  background-color: transparent;
}

/* Address Book Service Icon */
 .abs .serviceIcon {
  background-image: url("../images/new_services_icons_sprite.png");
  background-repeat: no-repeat;
  background-position: -30px center;
  background-color: transparent;
}

/* Options Service Icon */
 .options .serviceIcon {
  background-image: url("../images/new_services_icons_sprite.png");
  background-repeat: no-repeat;
  background-position: -88px center;
  background-color: transparent;
}
```

8. To change just the Mail service selector icon, you can still use an icon sprite, only modifying the Mail service in the `c11n_icons.css` file:

c11n/allDomain/layout/css/c11n_icons.css

```
/* Service Icons */
/* Mail Service Icon */
 .mail .serviceIcon {
  background-image: url("../images/new_services_icons_sprite.png");
  background-repeat: no-repeat;
  background-position: 0px center;
  background-color: transparent;
}
```

9. To use an individual image instead of an image sprite, point the `background-image` to the icon image (in the `c11n/allDomain/layout/images/` directory) of the individual service. For example, in the Mail service:

c11n/allDomain/layout/css/c11n_icons.css

```
/* Service Icons */
/* Mail Service Icon */
 .mail .serviceIcon {
  background-image: url("../images/new_mail_service_icon.png");
  background-repeat: no-repeat;
  background-position: 0px center;
  background-color: transparent;
}
```

10. Restart GlassFish Server and clear the browser cache to see the change.

After Changing the Icons in the Service Selector

This screenshot shows what the service selector looks like when all icons are modified by using the new icon sprite referenced in this example:

Chapter 12. Customization Example - Disabling Identity Settings

Customization Example - Disabling Identity Settings

The following example disables the ability for end users to edit name, email and Reply-To: settings for any external identities found in the Options Tab --> Mail --> Local Account --> Identities in the Mail Personal Account Settings.

1. Enable client customizations.

```
./iwcadmin -o client.enablecustomization -v true
```

2. Create the following directory structure under the *appserver-domain-root* `/docroot/iwc_static/` directory to hold the customization files.

```
c11n/
c11n/allDomain
c11n/allDomain/js
c11n/allDomain/js/widget
```

3. Generate the general customization configuration file.
   This configuration file enables JavaScript customization (`jsEnabled: true`) across all domains (`module: "allDomain"`).

c11n/config.js

```
dojo.provide("c11n.config");

c11n.config = {

        // allDomain configuration
        allDomain: {
                module: "allDomain",           // module name
                themeEnabled: false,           // true if theme is
customized
                i18nEnabled: false,            // true if i18n is
customized
                jsEnabled: true                // true if js is
customized

                // the last entry must not end with comma

        }
}
```

4. Create the domain specific customization file (if it doesn't already exist).

c11n/allDomain/js/customize.js

```
dojo.provide("c11n.allDomain.js.customize");

// Disable Identity Settings
dojo.require("c11n.allDomain.js.widget.mail.option.Identity");
```

5. Create the JavaScript file that disables Identity settings.

c11n/allDomain/js/widget/mail/option/Identity.js

```
dojo.provide("c11n.allDomain.js.widget.mail.option.Identity");
dojo["require"]("iwc.widget.mail.option.Identity");
dojo.declare("iwc.widget.mail.option.Identity",
iwc.widget.mail.option.Identity, {

        postCreate: function () {
                this.inherited(arguments);
                this.senderName.setDisabled(true);
                this.senderEmail.setDisabled(true);
                this.replyTo.setDisabled(true);

        },

        last: ""

});
```

6. Restart GlassFish Server and clear the browser cache to see the change.

Chapter 13. Customization Example - Hiding the Quick Actions Menu

Customization Example – Hiding the Quick Actions Menu

The following example hides the Quick Actions menu illustrated in Common Widgets. In this example, the Quick Actions menu is being removed in all domains.

1. Enable client customizations.

```
./iwcadmin -o client.enablecustomization -v true
```

2. Create the following directory structure under the *appserver-domain-root* `/docroot/iwc_static/` directory to hold the customization files.

```
c11n/
c11n/allDomain
c11n/allDomain/js
c11n/allDomain/js/widget
```

3. Generate the general customization configuration file.
   This configuration file enables JavaScript customization (`jsEnabled: true`) across all domains (`module: "allDomain"`).

c11n/config.js

```
dojo.provide("c11n.config");

c11n.config = {

        // allDomain configuration
        allDomain: {
                module: "allDomain",           // module name
                themeEnabled: false,           // true if theme is
customized
                i18nEnabled: false,            // true if i18n is
customized
                jsEnabled: true                // true if js is
customized

                // the last entry must not end with comma

        }
}
```

4. Uncomment the following code in `customize.js`:

c11n/allDomain/js/customize.js

```
var loadCustomizedCssFile = function(url, id) {
        if (!id){
                id = url.replace(/\.\./gm, "").replace(/\//gm,
"_").replace(/\./gm, "_");
        }
        var link = window.document.getElementById(id);
        if (! link) {
                link = window.document.createElement("link");
                link.setAttribute("id", id);
                link.setAttribute("type", "text/css");
                link.setAttribute("rel", "stylesheet");
                var head =
window.document.getElementsByTagName("head")[0];
                head.appendChild(link);
        }

        link.setAttribute("href", url + "?" + djConfig.cacheBust);
}
loadCustomizedCssFile("../c11n/allDomain/layout/css/c11n.css")
```

5. Create and and add the following to the `c11n.css`:

iwc_static/c11n/allDomain/layout/css/c11n.css

```
.QuickActions {
        display: none;
}
```

6. Restart GlassFish Server and clear the browser cache to see the change.

Chapter 14. Customization Example - Displaying the Japanese Yen Symbol

Customization Example - Displaying the Japanese Yen Symbol (¥)

While the the ASCII equivalent of the backslash keyboard entry (\) in a Japanese environment should display the Japanese Yen (¥) symbol, on some fields in the Convergence UI (such as the Subject or Body of a Mail message), the backslash (\) is mistakenly displayed. To fix this issue, follow these steps:

> **Note**
> This example needs a minimum patch level of Convergence 2 Patch 3.

Topics:

To Display the Japanese Yen Symbol in the Subject, Calendar Title, and Calendar Description

1. Enable client customizations.

```
./iwcadmin -o client.enablecustomization -v true
```

2. Create the domain specific customization file and add the following code:

c11n/allDomain/js/customize.js

```
loadCustomizedCssFile("../c11n/allDomain/layout/css/c11n.css");
```

- Uncomment or add (if it doesn't already exist) the following code in `customize.js`:

c11n/allDomain/js/customize.js

```
var loadCustomizedCssFile = function(url, id) {
        if (!id){
                id = url.replace(/\.\./gm, "").replace(/\//gm,
"_").replace(/\./gm, "_");
        }
        var link = window.document.getElementById(id);
        if (! link) {
                link = window.document.createElement("link");
                link.setAttribute("id", id);
                link.setAttribute("type", "text/css");
                link.setAttribute("rel", "stylesheet");
                var head =
window.document.getElementsByTagName("head")[0];
                head.appendChild(link);
        }

        link.setAttribute("href", url + "?" +
djConfig.cacheBust);
}
loadCustomizedCssFile("../c11n/allDomain/layout/css/c11n.css")
```

3. To enable the Subject, Calendar title, Calendar description to accept the Japanese Yen symbol, edit the `c11n/allDomain/layout/css/c11n.css`, by removing the sample data and adding the following code:

c11n/allDomain/layout/css/c11n.css

```
body {
            font-family: "ms pgothic", arial, helvetica, sans-serif;
        }

    .dj_ie body * {
            font-family: "ms pgothic", arial, helvetica, sans-serif;
        }

    .dj_ie .FormSimpleTextarea .FormSimpleTextarea-inputText {
            font-family: "ms pgothic", arial, helvetica, sans-serif
!important;
        }
```

4. Restart GlassFish Server and clear the browser cache to see the change.

To Display the Japanese Yen Symbol in the Message Viewer

1. Enable client customizations.

```
./iwcadmin -o client.enablecustomization -v true
```

2. Create the following directory structure under the *appserver-domain-root* `/docroot/iwc_static/` directory to hold the customization files.

```
c11n/
c11n/allDomain
c11n/allDomain/js
c11n/allDomain/js/widget
```

3. Generate the general customization configuration file.
   This configuration file enables JavaScript customization (`jsEnabled: true`) across all domains (`module: "allDomain"`).

c11n/config.js

```
dojo.provide("c11n.config");

c11n.config = {

        // allDomain configuration
        allDomain: {
                module: "allDomain",           // module name
                themeEnabled: false,           // true if theme is
customized
                i18nEnabled: false,            // true if i18n is
customized
                jsEnabled: true                // true if js is
customized

                // the last entry must not end with comma

        }
}
```

4. To enable the Message Viewer to display the Japanese Yen symbol, create a new file: `c11n/allDomain/js/widget/mail/IframeMessagePane.js`. Add the following code:

c11n/allDomain/js/widget/mail/IframeMessagePane.js

```
dojo.provide("c11n.allDomain.js.widget.mail.IframeMessagePane");


dojo["require"]("c11n.allDomain.js.widget.mail.IframeMessagePane");

     iwc.widget.mail.IframeMessagePane.prototype.customCssUrl =
        "/iwc_static/c11n/allDomain/layout/css/MessagePane.css";
```

5. Create `/iwc_static/c11n/allDomain/layout/css/MessagePane.css` and add the following style:

iwc_static/c11n/allDomain/layout/css/MessagePane.css

```
.MailMessagePane body {
          font-family: "ms pgothic", arial, helvetica, sans-serif;
    }
```

6. To enable the Editor to use the Japanese Yen symbol, create a new file:

`c11n/allDomain/js/editor/Editor.js`. Add the following code:

c11n/allDomain/js/editor/Editor.js

```
dojo.provide("c11n.allDomain.js.editor.Editor");

     dojo["require"]("iwc.editor.Editor");

     iwc.editor.Editor.prototype.customCssUrl =
        "/iwc_static/c11n/allDomain/layout/css/Editor.css";
```

7. Create `/iwc_static/c11n/allDomain/layout/css/Editor.css` and add the following style:

/iwc_static/c11n/allDomain/layout/css/Editor.css

```
body {
          font-family: "ms pgothic", arial, helvetica, sans-serif;
   }
```

8. Add the following lines to `c11n/allDomain/js/customize.js`:

c11n/allDomain/js/customize.js

```
dojo.require("c11n.allDomain.js.editor.Editor");
   dojo.require("c11n.allDomain.js.editor.plugins.FontChoice");
   dojo.require("c11n.allDomain.js.widget.mail.IframeMessagePane");
```

9. Restart GlassFish Server and clear the browser cache to see the change.

To Add a New Font to the Editor Font Toolbar Pull-down Menu

Use the following steps to setup a default font for the editor:

1. Enable client customizations.

```
./iwcadmin -o client.enablecustomization -v true
```

2. Create the following directory structure under `<appserver domain root>/docroot/iwc_static/` to hold the customization files.

```
c11n/
c11n/allDomain
c11n/allDomain/js
```

3. Generate the general customization configuration file.
   This configuration file enables JavaScript customization (`jsEnabled: true`) across all domains (`module: "allDomain"`).

c11n/config.js

```
dojo.provide("c11n.config");

c11n.config = {

        // allDomain configuration
        allDomain: {
                module: "allDomain",           // module name
                themeEnabled: false,           // true if theme is
customized
                i18nEnabled: false,            // true if i18n is
customized
                jsEnabled: true                // true if js is
customized

                // the last entry must not end with comma

        }
}
```

4. Create the domain specific customization file (if it does not already exist).

c11n/allDomain/js/customize.js

```
dojo.provide("c11n.allDomain.js.customize");

dojo.require("c11n.allDomain.js.editor.plugins.FontChoice");
```

5. To add new Japanese fonts to the Editor font pull-down menu, use the following sample:

c11n/allDomain/js/editor/plugins/FontChoice.js

```
dojo.provide("c11n.allDomain.js.editor.plugins.FontChoice");

   dojo["require"]("iwc.editor.plugins.FontChoice");

   dojo.declare("iwc.editor.plugins.FontChoice",
    iwc.editor.plugins.FontChoice,
      {
           custom: {
                    fontName: [
                              "Ms pgothic",          // sans-serif
                              "Arial",               // sans-serif
                              "Comic Sans MS",       // cursive
                              "Courier",             // monospace
                              "Courier New",         // monospace
                              "Georgia",             //
transitional-serif
                              "Helvetica",           // sans-serif
                              "Lucida Console",      // sans-serif
                              "Tahoma",              //sans-serif
                              "Times",               // serif
                              "Times New Roman",     // serif
                              "Trebuchet MS",        // sans-serif
                              "Verdana"              // sans-serif
                               ],
                     fontSize: [1,2,3,4,5,6,7] // sizes according
to the old HTML FONT SIZE
                     },
          last: ""
  });
```

6. Restart GlassFish Server and clear the browser cache to see the change.

Alternatively, you can also translate the font name "ms gothic" to your localized language. See: To Translate the Font Name "ms pgothic" in the Editor Font Choice Pull-down Menu.

To Translate the Font Name "ms pgothic" in the Editor Font Choice Pull-down Menu

To translate the font name "ms pgothic" in the editor font choice pull-down menu, follow these steps:

1. Turn on the `nls` translation by editing the `c11n/config.js`:
   This configuration file enables i18n customization (`i18nEnabled: true`) across all domains (`module: "allDomain"`).

c11n/config.js

```
dojo.provide("c11n.config");

c11n.config = {

        // allDomain configuration
        allDomain: {
                module: "allDomain",           // module name
                themeEnabled: false,           // true if theme is
customized
                i18nEnabled: true,            // true if i18n is
customized
                jsEnabled: false                // true if js is
customized

                // the last entry must not end with comma

        }
}
```

2. Edit the file `c11n/allDomain/nls/resources.js`, add the following:

c11n/allDomain/nls/resources.js

```
{
       "ms pgothic": "your l10n font name",
        last: ""
   }
```

3. Restart GlassFish Server and clear the browser cache to see the change.

Chapter 15. Customization Example - Disabling Quick Parsing Calendar Capability

Customization Example - Disabling Quick Parsing Calendar Capability

If you click or drag within a Calendar view, an event balloon displays. If quick parsing is enabled when a user enters "1PM Lunch at The Counter" in the text box of the event balloon, an event titled "Lunch" with location "The Counter" at 1PM for a duration of one hour is created. If, however, quick parsing is disabled, an event with title "1PM Lunch at the Counter" is created at the clicked time. The following example disables the quick parsing Calendar capability.

> **Note**
> This example works with versions beginning with Convergence 2 Patch 4.

1. Enable client customizations.

```
./iwcadmin -o client.enablecustomization -v true
```

2. Create the following directory structure under the `appserver-domain-root/docroot/iwc_static/` directory to hold the customization files.

```
c11n/
c11n/allDomain
c11n/allDomain/js
c11n/allDomain/js/widget
c11n/allDomain/js/widget/calendar
```

3. Generate the general customization configuration file.
   This configuration file enables JavaScript customization (`jsEnabled: true`) across all domains:

c11n/config.js (module: "allDomain")

```
dojo.provide("c11n.config");

c11n.config = {

        // allDomain configuration
        allDomain: {
                module: "allDomain",           // module name
                themeEnabled: false,           // true if theme is
customized
                i18nEnabled: false,            // true if i18n is
customized
                jsEnabled: true                // true if js is
customized

                // the last entry must not end with comma

        }
}
```

4. Create and add to the domain specific customization file.

c11n/allDomain/js/customize.js

```
dojo.provide("c11n.allDomain.js.customize");

// Disable quick parsing in calendar
dojo.require("c11n.allDomain.js.widget.calendar.Balloon");
```

5. Create the JavaScript file that disables quick parsing feature in calendar:

c11n/allDomain/js/widget/calendar/Balloon.js

```
dojo.provide("c11n.allDomain.js.widget.calendar.Balloon");

dojo["require"]("iwc.widget.calendar.Balloon");

dojo.declare("iwc.widget.calendar.QuickEventBalloon",
iwc.widget.calendar.QuickEventBalloon, {
quickParsingEnabled: false ,
         last: ""
});
```

6. Restart GlassFish Server and clear the browser cache to view the change.

Chapter 16. Customization Example - Changing Default Folder Mappings for Sent and Deleted Messages

Customization Example - Changing Default Folder Mappings for Sent and Deleted Messages

> **Note**
> This example requires a minimum Convergence patch level of Convergence 2-2.01 (2 patch 2).

By default, copies of sent messages are mapped to a Sent folder and deleted messages are mapped to a Trash folder in the Convergence UI. You can change the default folder mappings: 'Place a copy in: Sent' and 'Move messages to: Trash' in the Options --> Mail --> General tab. To do so, use the following customization example where the Sent and Trash folder default mappings are changed to Sent Messages and Deleted Messages:

1. Enable client customizations.

```
./iwcadmin -o client.enablecustomization -v true
```

2. Create the following directory structure under the *appserver-domain-root* `/docroot/iwc_static/` directory to hold the customization files.

```
c11n/
c11n/allDomain
c11n/allDomain/js
c11n/allDomain/js/widget
```

3. Create the domain specific customization file (if it doesn't already exist) and add the following code:

c11n/allDomain/js/customize.js

```
dojo.provide("c11n.allDomain.js.customize");

// Change the default folder mappings from Sent and Trash to Sent
Messages and Deleted Messages.
iwc.systemPrefs.mail.defaultImapSystemFolders.trash = "Deleted
Messages";
iwc.systemPrefs.mail.defaultImapSystemFolders.sent = "Sent
Messages";
```

4. Generate and edit the general customization configuration file.
This configuration file enables i18n customization (`i18nEnabled: true`) across all domains (`module: "allDomain"`).

c11n/config.js

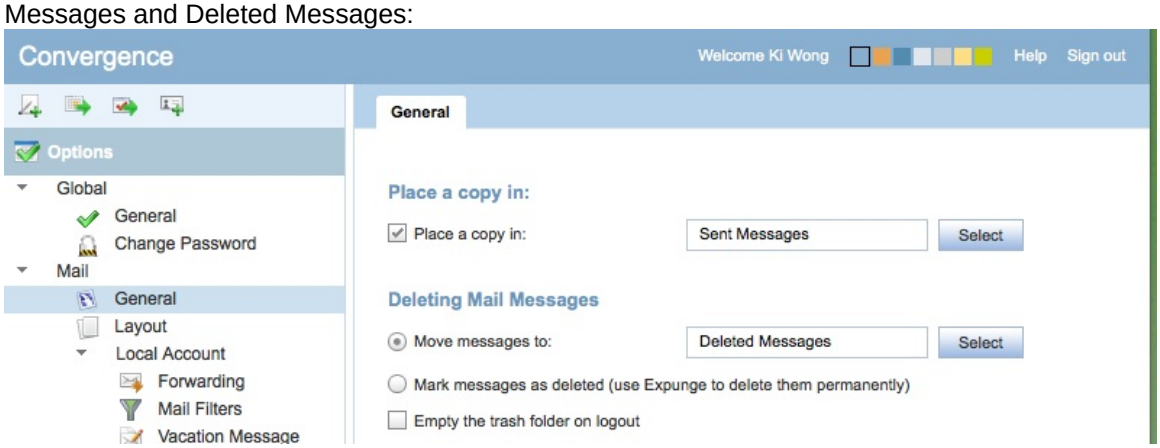
```
dojo.provide("c11n.config");

c11n.config = {

        // allDomain configuration
        allDomain: {
                module: "allDomain",           // module name
                themeEnabled: false,           // true if theme is
customized
                i18nEnabled: true,            // true if i18n is
customized
                jsEnabled: false                // true if js is
customized

                // the last entry must not end with comma

        }
}
```

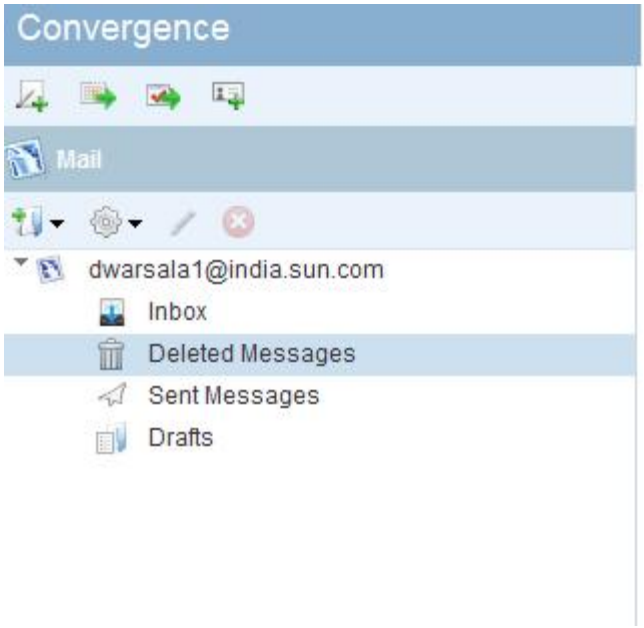
5. Edit `resources.js` in `c11n/allDomain/nls/` by deleting the sample content and adding the following code:

c11n/allDomain/nls/resources.js

```
{
        "trash": "Deleted Messages",
        "sent_items": "Sent Messages",

        last: ""
}
```

6. Restart GlassFish Server and clear the browser cache to see the change.
   In the following figures, the default Sent and Trash folder mappings are changed to Sent Messages and Deleted Messages:




Convergence
Mail
dwarsala1@india.sun.com
Inbox
Deleted Messages
Sent Messages
Drafts

Chapter 17. Customization Example - Modifying the Document Title and the Convergence Text in the Banner

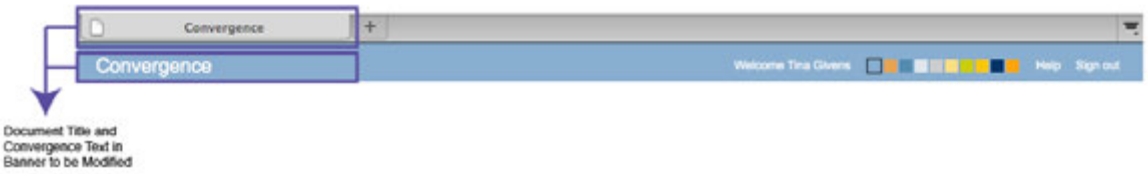
Customization Example - Modifying the Document Title and the Convergence Text in the Banner

This example provides an easy method to modify the Convergence text in the document title and on the banner in the `main.html` page.

> **Note**
> You can only do this type of customization on a per domain basis. In other words, doing such modifications will impact the single domain you're modifying, not all of your domains.

> **Note**
> To customize the theme and banner for all domains, or to do extensive theme and banner customization (such as adding new themes, new colors, or logos), refer to Customizing Themes and Banner.

The following illustration shows the Convergence tab and banner text prior to modification:



1. Enable client customizations.

```
./iwcadmin -o client.enablecustomization -v true
```

2. Create the following directory structure under the *appserver-domain-root* `/docroot/iwc_static/` directory to hold the customization files. In this example, `mydomain_org` is the single domain that's being modified.

```
c11n/
c11n/mydomain_org
c11n/mydomain_org/js
c11n/mydomain_org/js/widget
```

3. Generate and edit the general customization configuration file.
   This configuration file enables i18n customization (`i18nEnabled: true`) for the `"mydomain_org"` domain.

c11n/config.js

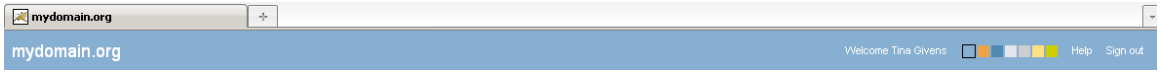
```
dojo.provide("c11n.config");

c11n.config = {

        // allDomain configuration
        allDomain: {
                module: "allDomain",           // module name
                themeEnabled: false,           // true if theme is
customized
                i18nEnabled: false,            // true if i18n is
customized
                jsEnabled: false               // true if js is
customized

                // the last entry must not end with comma

        },

        // mydomain_org configuration
        mydomain_org: {
                module: "mydomain_org",         // module name
                themeEnabled: false,            // true if theme is
customized
                i18nEnabled: true,              // true if i18n is
customized
                jsEnabled: false                // true if js is
customized

                // the last entry must not end with comma

        }
}
```

4. Create the `resources.js` file in the `c11n/mydomain_org/nls` directory, and add the desired text to the `login_product_name` for the domain, as shown in this step with `mydomain.org`:

c11n/mydomain_org/nls

```
{
       login_product_name: "mydomain.org",
       last: ""
}
```

5. Restart GlassFish Server and clear the browser cache to see the change.
   In the following illustration, `mydomain.org` has been added to both the document title and the Convergence text in the banner:



Chapter 18. Customization Example - Hiding User-created Folder Names

Customization Example - Hiding User-created Folder Names

> **Note**
> This needs a minimum patch level of Convergence 2 Patch 4.

If users change default folders (such as Sent, Drafts, Spam, or Trash) to different user-created folders, the folder name is changed from the default to the user-created folder name. If you want to hide user-created folder names and only display the default names, the following customization hides the user-created folder names in the mail folder tree and in the options page (Options > Mail > General):

1. Enable customization in the Convergence Server.
   Before you can customize the UI, the Convergence Server must be enabled for customization. Use the command-line utility, `iwcadmin`, to set the `client.enablecustomization` parameter to `true`. For example:

   ```
   ./iwcadmin -W <password file> -o client.enablecustomization -v true
   ```

2. Create the customization directory, `/c11n`.
   All customizations must be located under this directory.

   ```
   /iwc_static/c11n
   ```

3. Create the configuration file, `config.js`, for the customization.

   ```
   /c11n/config.js
   ```

   You can create this file by copying an example of the `config.js` file from the sample customization directory:

   ```
   /iwc_static/c11n_sample
   ```

4. In the `config.js` configuration file, set the `i18nEnabled` flag to `true`.
5. Create an `nls` subdirectory for the l10n module:

   ```
   iwc_static/c11n/<domain>/nls
   ```

   where *domain* is the name of the domain for which you want to customize the UI.
   At runtime, for each domain that has `i18nEnabled` enabled, Convergence loads that domain's l10n resource file, named `resources.js`. To apply a customization to all domains, use the `allDomain` as your domain name.
6. Create a default resource file named `resources.js`.

```
iwc_static/c11n/<domain>/nls/resources.js
```

7. In the `resources.js` file, add the following text strings:

iwc_static/c11n/allDomain/nls/resources.js

```
custom_folder_drafts: "Drafts",
custom_folder_trash: "Trash",
custom_folder_sent: "Sent",
custom_folder_spam: "Spam",
```

This customized resources file (under the `c11n` directory) inherits the default values in the standard resources file and extends them with the new, custom values.
At runtime, the `resources.js` file in the default directory is loaded first; then the `resources.js` in this customization directory is loaded. Thus, Convergence first loads the standard values (including UI widget labels) in the default resources file; it then overrides those values with any customizations in this resources file.

8. Similarly, for each language you want to hide the user-created folders, create a language directory and its own `resources.js`. For example, to hide the user-created folders for French language, create the `fr` sub-directory in the `nls` directory and add the following strings:

iwc_static/c11n/allDomain/nls/fr/resources.js

```
custom_folder_drafts: "Brouillons",
custom_folder_trash: "Corbeille",
custom_folder_sent: "Envoy\303\251",
custom_folder_spam: "Courrier ind\303\251sirable",
```

Note
Be sure to use a UTF-8 capable text editor to edit your changes.

Chapter 19. Customization Example - Changing the Corporate Directory Name

Customization Example - Changing the Corporate Directory Name

This customization example outlines how change the Corporate Directory name in the Address Book tab (default: Corporate Directory).

For a new user, a user whose corporate entry has previously been deleted from Directory Server, or if new corporate directories are added to the deployment, you can use the `ab.corpdir.[<identifier>].description` and `ab.corpdir.[<identifier>].displayname` configuration parameters to change the name of the Corporate Directory tab in the Convergence Address Book (See: Convergence Reference).

But, if a corporate directory entry already exists for the end user, the `ab.corpdir.[<identifier>].description` and `ab.corpdir.[<identifier>].displayname` configuration parameters do not display the new corporate directory name. Instead, the default name, Corporate Directory, displays. The following instructions explain how to customize the corporate directory name for all users in a domain. You can use this example for both existing as well as new directory entries without having to change the `displayName` entries in `o=PiServerDb` for every user.

As with other Convergence elements, you can customize the product name in a particular domain, or it can be applied to all domains. The following example uses the `allDomain` directory to apply the product name change to all domains in the deployment:

1. Enable customization in the Convergence Server.
   Before you can customize the UI, the Convergence Server must be enabled for customization. Use the command-line utility, `iwcadmin`, to set the `client.enablecustomization` parameter to `true`. For example:

   ```
   ./iwcadmin -W <password file> -o client.enablecustomization -v true
   ```

2. Create the customization directory, `/c11n`.
   All customizations must be located under this directory.

   ```
   /iwc_static/c11n
   ```

3. Create the configuration file, `config.js`, for the customization.

   ```
   /c11n/config.js
   ```

   You can create this file by copying an example of the `config.js` file from the sample customization directory:

```
/iwc_static/c11n_sample
```

4. In the `config.js` configuration file, set the `i18nEnabled` flag to `true` across all domains (`"allDomain"`). For example:

c11n/config.js

```
dojo.provide("c11n.config");

c11n.config = {

        // allDomain configuration
        allDomain: {
                module: "allDomain",            // module name
                themeEnabled: false,            // true if theme is
customized
                i18nEnabled: true,              // true if i18n is
customized
                jsEnabled: false                // true if js is
customized

                // the last entry must not end with comma

        }
}
```

5. Create an `nls` subdirectory.

```
iwc_static/c11n/allDomain/nls
```

6. Create a default resource file named `resources.js`.
   This customized resources file (under the `c11n` directory) inherits the default values in the standard resources file and extends them with the new, custom values. At runtime, for each domain that has `i18nEnabled` enabled, Convergence loads that domain's l10n resource file, named `resources.js`. The `resources.js` file in the default directory is loaded first; then the `resources.js` in this customization directory is loaded. Thus, Convergence first loads the standard values (including labels) in the default resources file; it then overrides those values with any customizations in this resources file.

```
iwc_static/c11n/allDomain/nls/resources.js
```

7. If your `resources.js` file is not new, remove any references to corporate directory (for example, `corp_dir`, `the_corporate_directory`, `corporate_lookup`, or `corp_dir_lookup`.
8. Add the following line `resources.js`:

c11n/allDomain/nls/resources.js

```
{
 corporate_directory: "Example Directory", //where "Example
Directory" is your Corporate Directory's name

 last: ""
}
```

9. Restart GlassFish Server and clear the browser cache to see the change.

Chapter 20. Customization Example - Printing the Japanese Yen Symbol

Customization Example - Displaying and Printing the Japanese Yen (¥) Symbol

> **Note**
> This example needs a minimum patch level of Convergence 2 Patch 3.

The Japanese Yen sign (¥) (specifically, the ASCII equivalent of backslash (\) keyboard entry in a Japanese environment) is not recognized or displaying properly in many parts of the Convergence client, such as:

- Calendar Event Description field
- Calendar Task Note field
- Print Chat Transcript
- Address Book Create Contact
- Address Book Create Contact Notes
- Mail Print
- Calendar Print
- Address Book Print Contact

In order to properly print the Japanese Yen (¥) symbol in the Convergence UI (by using the ASCII equivalent of backslash (\) keyboard entry in the Japanese environment), you need to make the following modification to your customization:

> **Note**
> Since the `Default.css` is part of the Convergence-based codes as opposed to the `c11n` directory, any upgrade updates the `Default.css` and overrides your customization. Therefore, you **must** re-apply these customization changes after each Convergence upgrade.

Edit the `/iwc_static/layout/css/Default.css` by adding the following lines at the end of the file:

/iwc_static/layout/css/Default.css

```
body {
     font-family: "ms pgothic", arial, helvetica, sans-serif;
   }

   .dj_ie body * {
     font-family: "ms pgothic", arial, helvetica, sans-serif;
   }

   .dj_ie .FormSimpleTextarea .FormSimpleTextarea-inputText {
     font-family: "ms pgothic", arial, helvetica, sans-serif !important;
   }
```

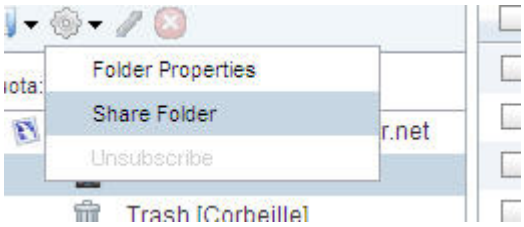
Chapter 21. Customization Example - Removing Folder Sharing and Subscribing Menu Options

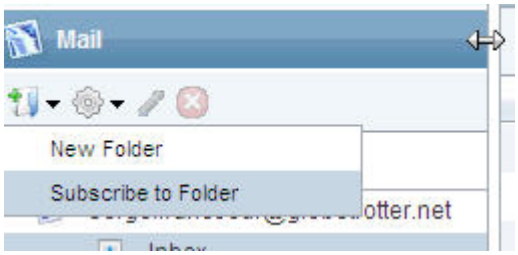
Customization Example - Removing Folder Sharing and Subscribing Menu Options

While you are able to disable folder sharing and folder subscriptions with the following commands:

```
iwcadmin -o mail.restrictanyone -v true
```
and
```
configutil -o store.privatesharedfolders.restrictanyone -v 1,
```

you might choose to take a further step and remove these options from the drop-down menus. The following illustrations display the Share Folder and Subscribe to Folder menu options prior to removing them from the menus:





This example describes how to remove folder sharing and subscribing menu options from the drop-down menus.

1. Enable client customizations.

   ```
   ./iwcadmin -o client.enablecustomization -v true
   ```

2. Create the following directory structure under the *appserver-domain-root* `/docroot/iwc_static/` directory to hold the customization files.

   ```
   c11n/
   c11n/allDomain
   c11n/allDomain/js
   c11n/allDomain/js/widget
   ```

3. Generate the general customization configuration file.
   This configuration file enables JavaScript customization (`jsEnabled: true`) across all domains (`module: "allDomain"`).

c11n/config.js

```
dojo.provide("c11n.config");

c11n.config = {

        // allDomain configuration
        allDomain: {
                module: "allDomain",           // module name
                themeEnabled: false,           // true if theme is
customized
                i18nEnabled: false,            // true if i18n is
customized
                jsEnabled: true                // true if js is
customized

                // the last entry must not end with comma

        }
}
```

4. Create the domain specific customization file.

c11n/allDomain/js/customize.js

```
dojo.provide("c11n.allDomain.js.customize");

// Remove folder sharing and subscribing menu options from the
drop-down menus
dojo.require("c11n.allDomain.js.widget.mail.Navigator");
```

5. Create and edit the JavaScript customization file to remove folder sharing and subscribing menu options from the drop-down menus.

c11n/allDomain/js/widget/mail/Navigator.js

```


dojo.provide("c11n.allDomain.js.widget.mail.Navigator");

dojo["require"]("iwc.widget.mail.Navigator");

dojo.declare("iwc.widget.mail.Navigator",
iwc.widget.mail.Navigator, {
        l10n: iwc.api.getLocalization(),

        postCreate: function() {
                this.inherited(arguments);
                var _this = this;

                // hide the dropdown menu
                dojo.style(this.folderSubscribeButton.domNode,
"display", "none");
                dojo.style(this.folderPropertiesButton.domNode,
"display", "none");
```

```
                // create new button for "New Folder" and
"Properties"
                var btnNewFolder = new dijit.form.Button(
                                        {
                                                label:
this.l10n.create_folder,
                                                onClick:
dojo.hitch(this, "onCreateFolder"),
                                                iconClass:
"FoldersActionMenuNewFolderIcon"
                                        }
                );
                dojo.place(btnNewFolder.domNode,
this.folderSubscribeButton.domNode, "before");

                var btnProperties = new dijit.form.Button(
                                        {
                                                label:
this.l10n.folder_properties,
                                                onClick:
dojo.hitch(this, "onFolderProps"),
                                                iconClass:
"propertiesIcon"
                                        }
                );
                dojo.place(btnProperties.domNode,
this.folderSubscribeButton.domNode, "before");

                // remove any share/subscribe/unsubscribe from
context menu
                dojo.each(this._menuItems,
                        function(o) {
                                _this._menuItems[o.key] =
dojo.filter(o.value,
                                        function(menuItem) {
                                                return
(menuItem.label != _this.l10n.folder_sharing &&

menuItem.label != _this.l10n.unsubscribe_folder);
                                        }
                                );
                        }
                );
        },
```

```
        last: ""

});
```

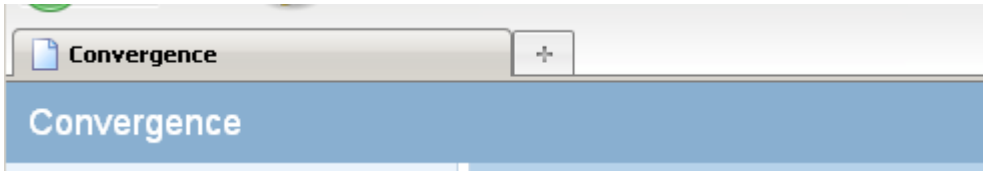
6. Restart GlassFish Server and clear the browser cache to see the change.

Chapter 22. Customization Example - Removing or Changing the Product Name on the Main Page

Customization Example - Removing or Changing the Product Name on the Main Page

This example outlines how to remove or to change the product name (default: Convergence) on the upper left-hand corner of the UI.
The following illustration shows the default product name (Convergence) on the main page of the UI:

Default Product Name



As with other Convergence elements, you can customize the product name in a particular domain, or it can be applied to all domains. The following example uses the `allDomain` directory to apply the product name change to all domains in the deployment:

1. Enable customization in the Convergence Server.
   Before you can customize the UI, the Convergence Server must be enabled for customization. Use the command-line utility, `iwcadmin`, to set the `client.enablecustomization` parameter to `true`. For example:

   ```
   ./iwcadmin -W <password file> -o client.enablecustomization -v true
   ```

2. Create the customization directory, `/c11n`.
   All customizations must be located under this directory.

   ```
   /iwc_static/c11n
   ```

3. Create the configuration file, `config.js`, for the customization.

   ```
   /c11n/config.js
   ```

   You can create this file by copying an example of the `config.js` file from the sample customization directory:

   ```
   /iwc_static/c11n_sample
   ```

4. In the `config.js` configuration file, set the `i18nEnabled` flag to `true` across all domains (`"allDomain"`). For example:

c11n/config.js

```
dojo.provide("c11n.config");

c11n.config = {

        // allDomain configuration
        allDomain: {
                module: "allDomain",            // module name
                themeEnabled: false,            // true if theme is
customized
                i18nEnabled: true,              // true if i18n is
customized
                jsEnabled: false                // true if js is
customized

                // the last entry must not end with comma

        }
}
```

5. Create an `nls` subdirectory.

```
iwc_static/c11n/allDomain/nls
```

6. Create a default resource file named `resources.js`.
   This customized resources file (under the `c11n` directory) inherits the default values in the standard resources file and extends them with the new, custom values. At runtime, for each domain that has `i18nEnabled` enabled, Convergence loads that domain's l10n resource file, named `resources.js`. The `resources.js` file in the default directory is loaded first; then the `resources.js` in this customization directory is loaded. Thus, Convergence first loads the standard values (including labels) in the default resources file; it then overrides those values with any customizations in this resources file.

```
iwc_static/c11n/allDomain/nls/resources.js
```

7. In the `resources.js` file, after the `login_welcome_msg: "Welcome to Convergence"`, add a new line for the **`login_product_name`** label.
   a. To change the product name in the UI, add the new name to the value of the label (in this example: **`"Customized Convergence"`**)

c11n/allDomain/nls/resources.js

```
{
 helloConvergence_btn: "Hello Convergence",
 plaxo_btn: "Plaxo",
 blank123_btn: "123",
 compose_tab: "New Mail Message",
 login_welcome_msg: "Welcome to Convergence",
 login_product_name: "Customized Convergence",

 last: ""
}
```

The following illustration shows the modified product name (Customized Convergence) on the main page of the UI:

Modified Product Name

b. To remove the product name in the UI (default: Convergence), set **`login_product_name`** to an empty string:

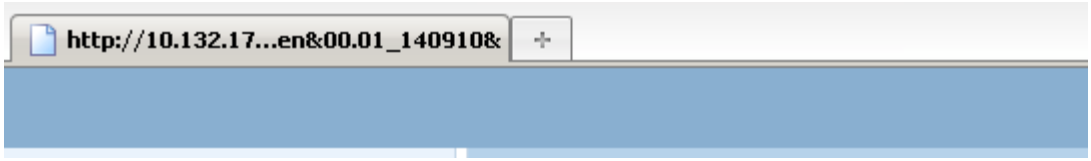
c11n/allDomain/nls/resources.js

```
{
 helloConvergence_btn: "Hello Convergence",
 plaxo_btn: "Plaxo",
 blank123_btn: "123",
 compose_tab: "New Mail Message",
 login_welcome_msg: "Welcome to Convergence",
 login_product_name: "",

 last: ""
}
```

The following illustration shows the main page of the UI without a product name:

Warning
Without a product name, your web browser will likely use the address bar location in the browser tab.

No Product Name



8. Restart GlassFish Server and clear the browser cache to see the change.

Chapter 23. Customization Example - Removing the 'Copy to' Button in the Corporate Address Book

Customization Example - Removing the 'Copy to' Button in the Corporate Address Book

If you want to limit your users' ability to copy Corporate Address Book contacts into their Personal Address Books, you can customize Convergence by removing the 'Copy to' button in the Address Book > Corporate Directory section of the UI:

> **Note**
> This customization limits but does not prevent users from copying Corporate Address Book contacts into their Personal Address Books.

1. Enable client customizations.

```
./iwcadmin -o client.enablecustomization -v true
```

2. Create the following directory structure under `<appserver domain root>/docroot/iwc_static/` to hold the customization files.

```
c11n/
c11n/allDomain
c11n/allDomain/js
c11n/allDomain/js/widget
c11n/allDomain/js/widget/addressBook
```

3. Generate the general customization configuration file.
   This configuration file enables JavaScript customization (`jsEnabled: true`) across all domains (`module: "allDomain"`).

c11n/config.js

```
dojo.provide("c11n.config");

c11n.config = {

        // allDomain configuration
        allDomain: {
                module: "allDomain",           // module name
                themeEnabled: false,           // true if theme is
customized
                i18nEnabled: false,            // true if i18n is
customized
                jsEnabled: true                // true if js is
customized

                // the last entry must not end with comma

        }
}
```

4. Create the domain specific customization file (if it does not already exist) and add the following line:

c11n/allDomain/js/customize.js

```
dojo.provide("c11n.allDomain.js.customize");

dojo["require"]("c11n.allDomain.js.widget.
addressBook._BookBrowserToolbar");
```

5. To limit users' ability to copy Corporate Address Book contacts into their Personal Address Books, add the following code to the `_BookBrowserToolbar.js` file:

c11n/allDomain/js/widget/addressBook/_BookBrowserToolbar.js

```
dojo.provide("c11n.allDomain.js.widget.
addressBook._BookBrowserToolbar");

dojo["require"]("iwc.widget.addressBook._BookBrowserToolbar");


dojo.declare("iwc.widget.addressBook._BookBrowserToolbar",
iwc.widget.addressBook._BookBrowserToolbar, {

        buildRendering: function() {

                // invoke the original buildRendering()

                this.inherited(arguments);

                dojo.style(this.copyToolbarButton.domNode,
"display", "none");       // remove the copyTo button

        },

        last: ""
});
```

6. Restart GlassFish Server and clear the browser cache to see the change.

Chapter 24. Customization Example - Removing the Google Maps Link From the Personal Address Book

Customization Example - Removing the Google Maps Link From the Personal Address Book

This Convergence example assumes that the reader has thoroughly read the Convergence Customization Guide.

To remove the Google Maps link from the Personal Address Book in Convergence 1.x, see Customization Example - Removing the Google Maps Link From the Personal Address Book in Convergence 1.x.

Comms Community Verified Contribution
This article came from the Comms Community and has been verified by the Oracle Communications Unified Communications Suite product team.

The following example removes the Google Maps link from the Personal Address Book when viewing a contact's address details.

1. Enable client customizations.

```
./iwcadmin -o client.enablecustomization -v true
```

2. Create - if not already done - the following directory structure under <appserver domain root>/docroot/iwc_static/ to hold the customization files.

```
c11n/
c11n/allDomain
c11n/allDomain/js
c11n/allDomain/js/service
```

3. Create - if not already done - the general customization configuration file.
   This configuration file enables JavaScript customization (`jsEnabled: true`) across all domains (`module: "allDomain"`).

c11n/config.js

```
dojo.provide("c11n.config");

c11n.config = {

        // allDomain configuration
        allDomain: {
                module: "allDomain",           // module name
                themeEnabled: false,           // true if theme is
customized
                i18nEnabled: false,            // true if i18n is
customized
                jsEnabled: true                // true if js is
customized

                // the last entry must not end with comma

        }
}
```

4. Create the domain specific customization file (if it doesn't already exist).

c11n/allDomain/js/customize.js

```
dojo.provide("c11n.allDomain.js.customize");

// Hide Google Maps link
dojo.require("c11n.allDomain.js.widget.addressBook
._DisplayContactBody");
```

5. Overwrite the JavaScript file `addressBook._DisplayContactBody.js` with the following code. This file must be available at the following location: `c11n/allDomain/js/widget/`.

> **Note**
> In this example, the `_DisplayContactAddressValue` is the actual widget that is being customized. It resides in the `_DisplayContactBody` source file.

c11n/allDomain/js/widget/addressBook._DisplayContactBody.js

```
dojo.provide("c11n.allDomain.js.widget.addressBook._DisplayContactBody");
dojo["require"]("iwc.widget.addressBook._DisplayContactBody");

   dojo.declare("iwc.widget.addressBook._DisplayContactAddressValue",
   iwc.widget.addressBook._DisplayContactAddressValue, {
        postCreate: function() {
            this.inherited(arguments);

            // hide the google maps in the address field
            dojo.addClass(this.mapLinkNode, "dijitHidden");
   },

   last: ""
});
```

6. Restart GlassFish Server and clear the browser cache to see the change.

Chapter 25. Customization Example - Restricting Outgoing Mail by Only Allowing Local Account Identity Parameters

Customization Example - Restricting Outgoing Mail by Only Allowing Local Account Identity Parameters

If you don't want to disable external POP account access, but you want to control what appears in the "From" pull-down menu for addressing, the following customization example only allows outgoing mail to use the parameters in the local account identity:

> **Note**
> To fully disable POP account functionality and to hide External Accounts in the Options panel in Mail, see Disabling External POP Account Access.

1. Enable client customizations.

```
./iwcadmin -o client.enablecustomization -v true
```

2. Create the following directory structure under `<appserver domain root>/docroot/iwc_static/` to hold the customization files.

```
c11n/
c11n/allDomain
c11n/allDomain/js
c11n/allDomain/js/widget
c11n/allDomain/js/widget/option
```

3. Generate the general customization configuration file.
   This configuration file enables JavaScript customization (`jsEnabled: true`) across all domains (`module: "allDomain"`).

c11n/config.js

```
dojo.provide("c11n.config");

c11n.config = {

        // allDomain configuration
        allDomain: {
                module: "allDomain",           // module name
                themeEnabled: false,           // true if theme is
customized
                i18nEnabled: false,            // true if i18n is
customized
                jsEnabled: true                // true if js is
customized

                // the last entry must not end with comma

        }
}
```

4. Create the domain specific customization file (if it does not already exist).

c11n/allDomain/js/customize.js

```
dojo.provide("c11n.allDomain.js.customize");

dojo.require("c11n.allDomain.js.widget.mail.CreateMessage");
```

5. To only allow outgoing mail to use the parameters in the local account identity, add the following code to the `CreateMessage.js` JavaScript file:

c11n/allDomain/js/widget/mail/CreateMessage.js

```
dojo.provide("c11n.allDomain.js.widget.mail.CreateMessage");

dojo["require"]("iwc.widget.mail.CreateMessage");

dojo.declare("iwc.widget.mail.CreateMessage",
iwc.widget.mail.CreateMessage, {
onToggleOptions: function(checked)  {
      this.inherited(arguments);
      dojo.addClass(this.senderIdSelect.domNode, "dijitHidden");
}

});
```

6. Restart GlassFish Server and clear the browser cache to see the change.

Chapter 26. Customizing Address Book Search Capability

Customization Example - Customizing Address Book Search Capability

> **Note**
> This example requires a minimum Convergence patch level of Convergence 2-2.01 (2 patch 2).

Customizing your search capability means that you can change the default search attribute in the Address Book search drop-down menu. For example, when you click the address book icon in Mail, Calendar, or Instant Messaging, the following window displays:

Add from Address Book Drop-down Search Capability Window

You can modify the default search (in this case, Display Name) attribute.

The following example described how to customize your search capability by using the `QuickSearchForm` Address Book widget:

1. Enable client customizations.

   ```
   ./iwcadmin -o client.enablecustomization -v true
   ```

2. Create the following directory structure under the *appserver-domain-root* `/docroot/iwc_static/` directory to hold the customization files.

   ```
   c11n/
   c11n/allDomain
   c11n/allDomain/js
   c11n/allDomain/js/widget
   ```

3. Generate the general customization configuration file.
   This configuration file enables JavaScript customization (`jsEnabled: true`) across all domains (`module: "allDomain"`).

c11n/config.js

```
dojo.provide("c11n.config");

c11n.config = {

        // allDomain configuration
        allDomain: {
                module: "allDomain",           // module name
                themeEnabled: false,           // true if theme is
customized
                i18nEnabled: false,            // true if i18n is
customized
                jsEnabled: true                // true if js is
customized

                // the last entry must not end with comma

        }
}
```

4. Create the domain specific customization file (if it doesn't already exist).

c11n/allDomain/js/customize.js

```
dojo.provide("c11n.allDomain.js.customize");

// Customize Search Capability in Address Book
dojo.require("c11n.allDomain.js.widget.addressBook.QuickSearchForm");
```

5. Create the JavaScript file that allows Address Book search capability.

c11n/allDomain/js/widget/addressBook/QuickSearchForm.js

```
dojo.provide("c11n.allDomain.js.widget.addressBook.QuickSearchForm");
iwc.widget.addressBook.QuickSearchForm, {
        postMixInProperties: function() {
                this.inherited(arguments);

                // "defaultSearch" attr can be one of the
followings:
                // "entry/displayname,email", "entry/displayname",
"email",
                // "person/surname", "person/givenname", "phone"
                this.attr("defaultSearch", "phone");
        }

});
```

6. Restart GlassFish Server and clear the browser cache to see the change.

Chapter 27. Customizing Convergence UI Elements and their Widgets

Customizing Convergence UI Elements and their Widgets

> **Note**
> For Convergence 1.x elements and widgets, refer to Convergence 1.x UI Elements and Javascript Widgets.

This page provides the following information to help you customize Convergence:

- Displays elements and regions in the Convergence UI
- Identifies the Javascript widgets (dojo digits) in the source code that generate, display, and manage these UI elements

This page includes the following topics:

Location of Javascript Widgets

The widgets are located in the following directory in the base code (`iwc_static`):

```
<GlassFish Server base>/docroot/iwc_static/js/iwc/widget
```

For example:

```
/opt/SUNWappserver/domains/domain1/docroot/iwc_static/js/iwc/widget
```

Widgets for each service are located in separate directories:

- Mail widgets: `widget/mail`
- Calendar widgets: `widget/calendar`
- Address Book widgets: `widget/addressBook`
- Instant Messaging widgets: `widget/im`
- Indexing and Search Service widgets: `widget/iss`

Option widgets are located within each service directory:

- Mail Options: `widget/mail/option`
- Calendar Options: `widget/calendar/option`
- Address Book Options: `widget/addressBook/option`

- Instant Messaging Options: `widget/im/option`
- Indexing and Search Service Options: `widget/iss/option`

Common widgets are located within the `widget` directory:

- Common widgets: `widget`
- Common form widgets: `widget/form`
- Common option widgets: `widget/option`

  You create your customized widgets in the customization directory, `c11n`. For example:

  ```
  <GlassFish Server base>/docroot/iwc_static/c11n/<domain>/js/widget
  ```

  where *domain* is the name of the domain where the customizations are applied.
  For example:

  ```
  /opt/SUNWappserver/domains/domain1/docroot/iwc_static/c11n/allDomain/
  ```

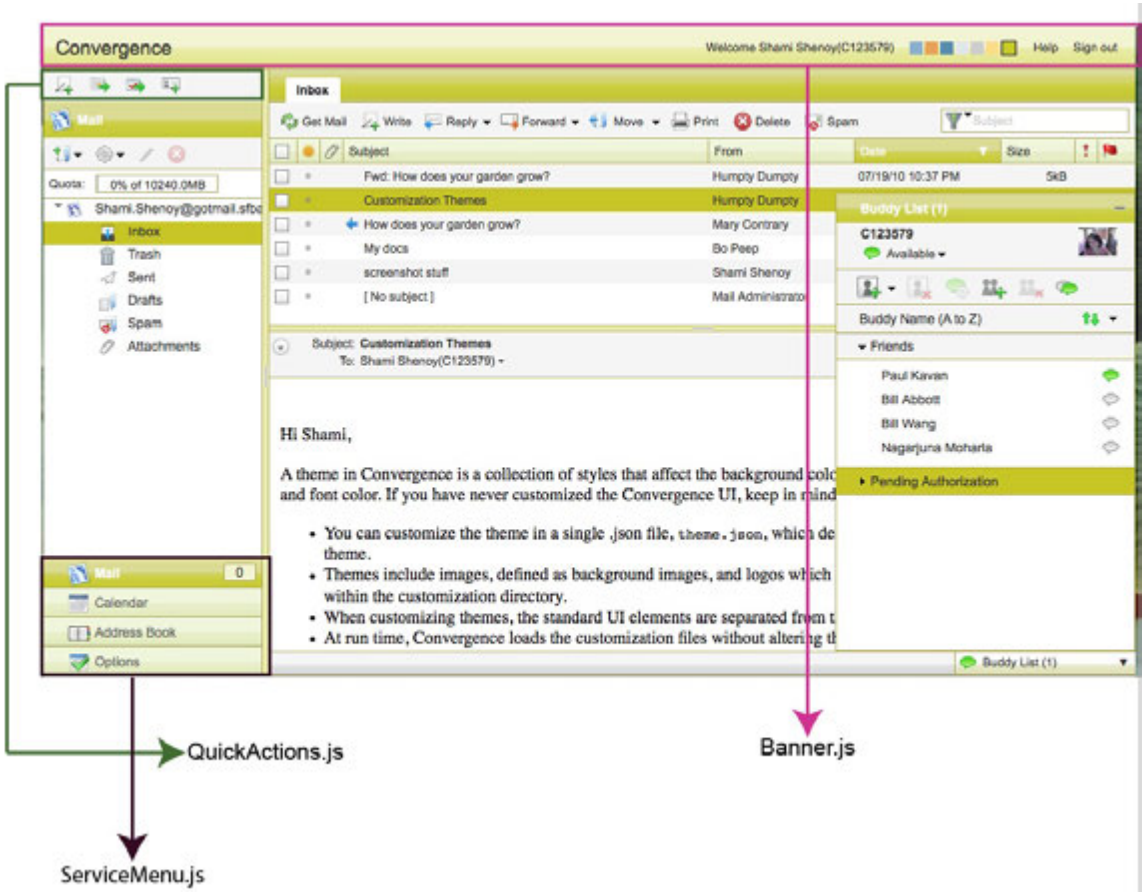
For more information, see Customizing Convergence - Technical Overview.

Common Widgets

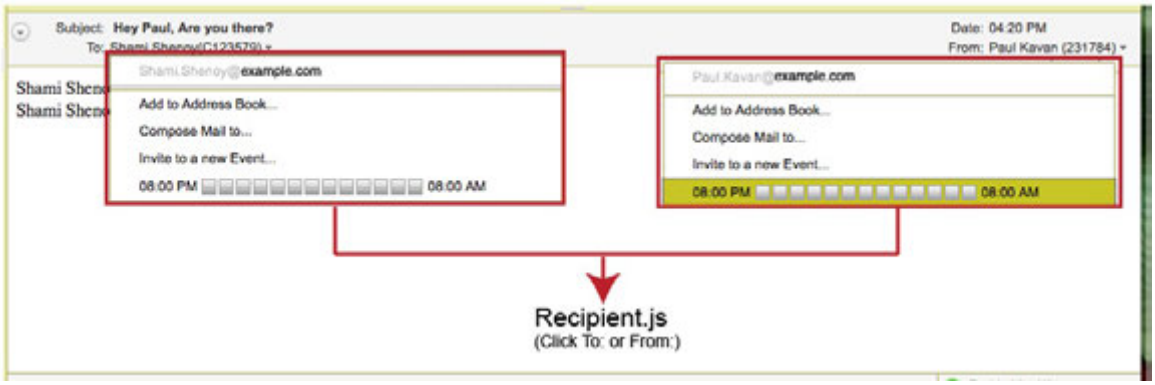
- Banner
- QuickActions
- Recipient
- ServiceMenu

Figure 1: Common Widgets in the UI

The `QuickActions.js`, `Banner.js`, and `ServiceMenu.js` are highlighted in the following illustration:



The `Recipient.js` widget is highlighted in the following illustration:



Mail Widgets

- mail.AdvancedSearch
- mail.CreateMessage
- mail.FolderDialog
- mail.FolderPropertiesDialog
- mail.FolderTree
- mail.Grid
- mail.OpenFolder
- mail.OpenMessage

- mail.MessageViewer
- mail.Navigator
- mail.PrintMessage
- mail.SelectFolderInput
- mail.ViewerContainer

Figure 2: Mail - Folders, Open Folder, Message Viewer, Navigator, Grid

The following mail widgets are highlighted in this illustration: `mail.FolderTree.js`, `mail.Navigator.js`, `mail.Message.Viewer.js`, `mail.OpenFolder.js`, `mail.ViewerContainer.js`, and `mail.Grid.js`.

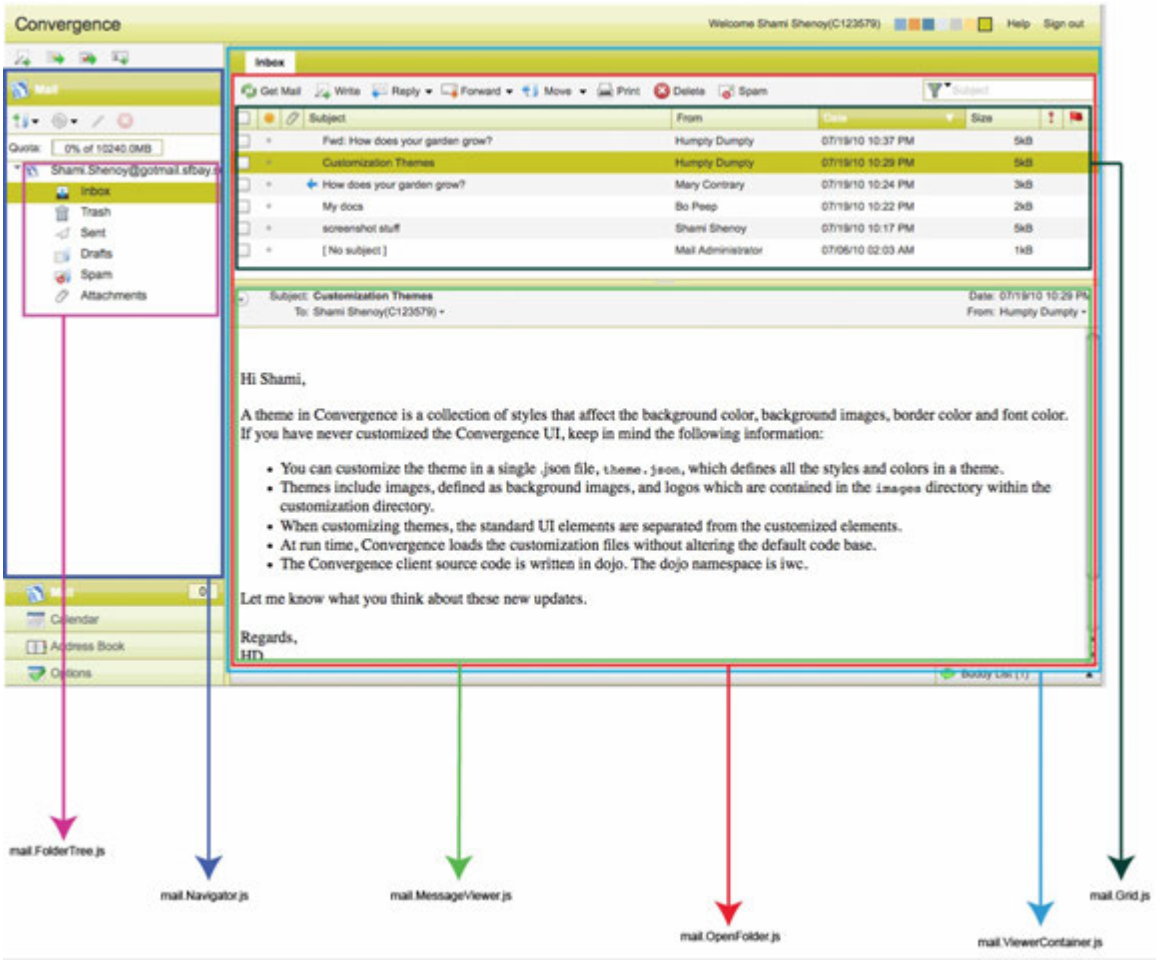


Figure 3: Mail - Create Message

The `mail.CreateMessage.js` and `addressBook.EmailComboTextarea.js` widgets are highlighted in the following illustration:

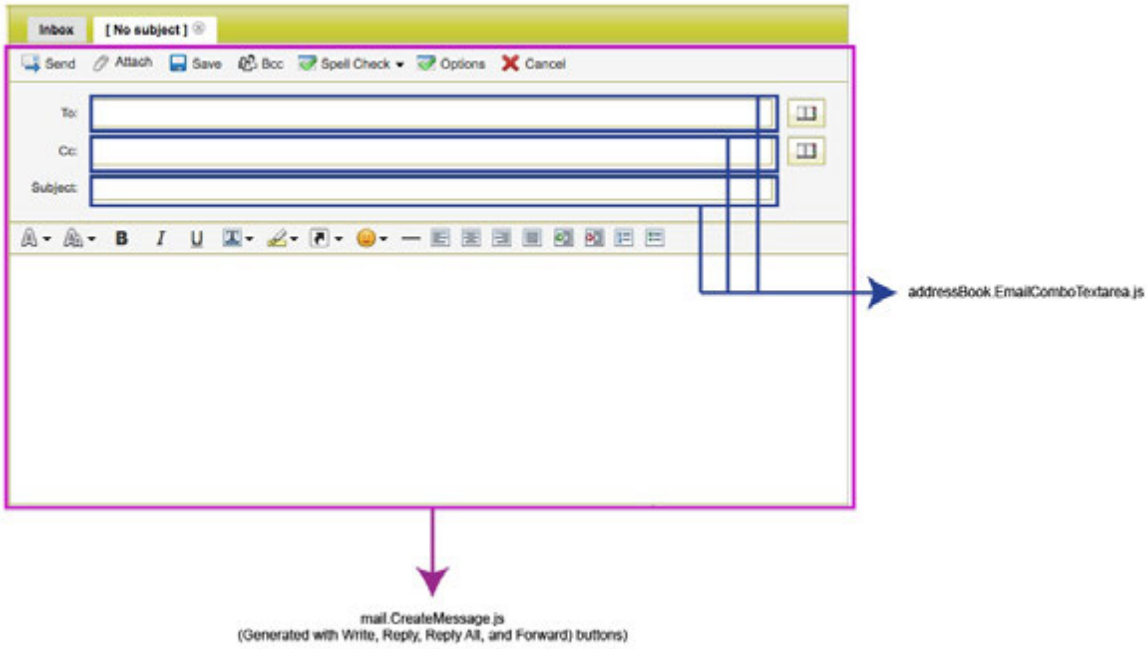


Figure 4: Mail - Open Message

In the following illustration, the `mail.OpenMessage.js` widget is highlighted:

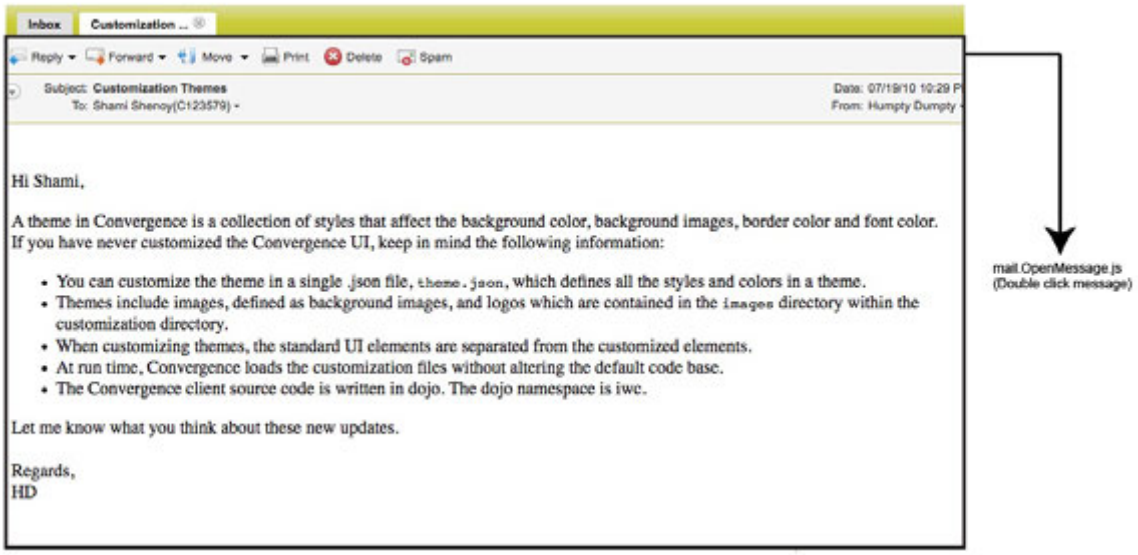


Figure 5: Mail - Advanced Search

The `mail.AdvancedSearch.js` and `mail.SelectFolderInput.js` widgets are highlighted in the following illustration:

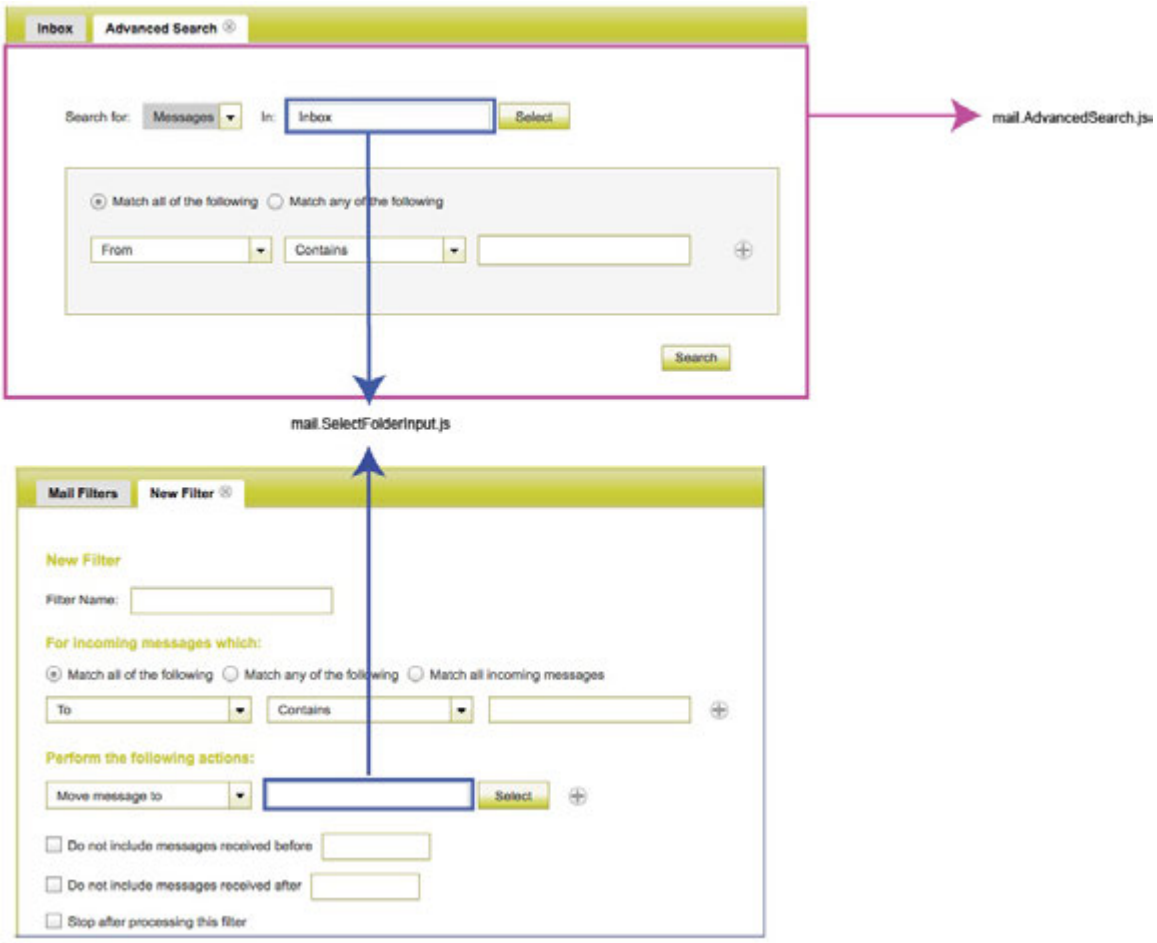


Figure 6: Mail - Print Message

The `mail.PrintMessage.js` widget is highlighted in the following illustration:

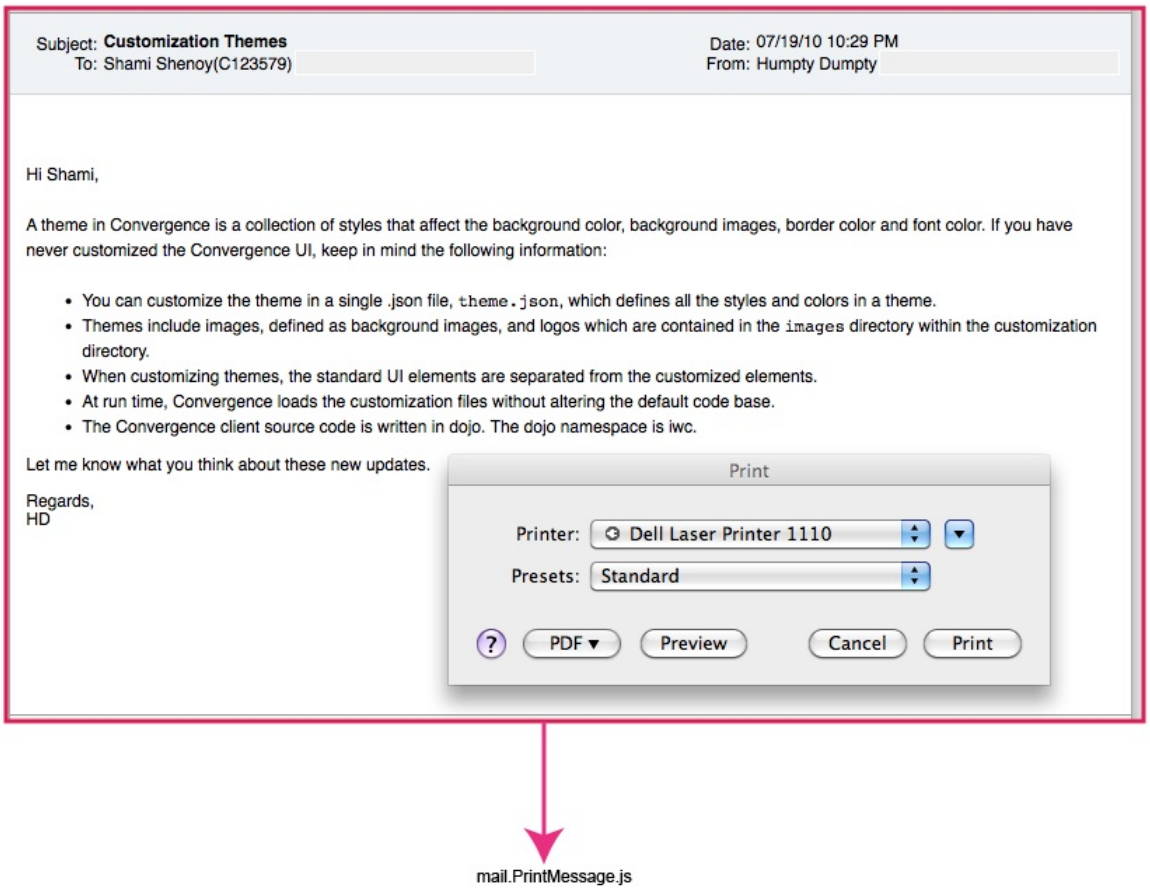


Figure 7: Mail - Folder Properties Dialog

The `mail.FolderPropertiesDialog.js` widget is highlighted in the following illustration:

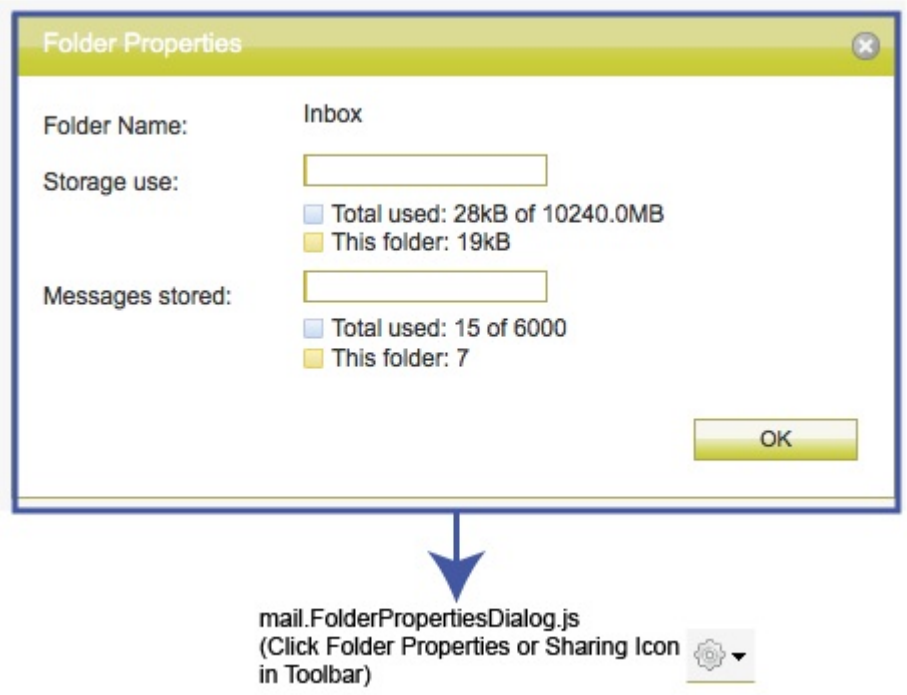
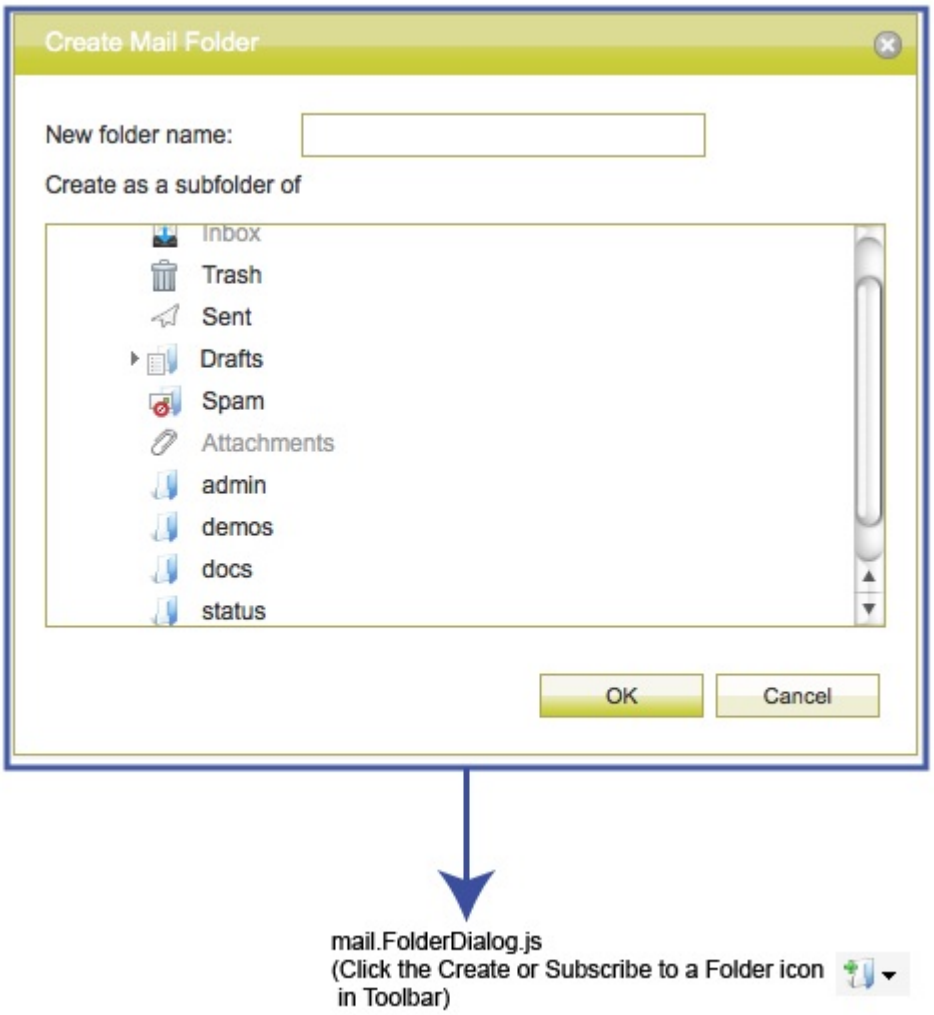


Figure 8: Mail - Create Folder Dialog

The `mail.FolderDialog.js` widget is highlighted in the following illustration:



Address Book Widgets

- addressBook.BookStoreItemSelector
- addressBook.CorporateBookBrowser
- addressBook.CreateContact
- addressBook.CreateContactDialog
- addressBook.CreateGroup
- addressBook.EmailComboTextarea
- addressBook.ExportContactsDialog
- addressBook.ImportContactsDialog
- addressBook.Navigator
- addressBook.PersonalBookBrowser
- addressBook.RenameGroupDialog

- addressBook.ResourceStoreItemSelector
- addressBook.ViewerContainer

Figure 9: Address Book - Open Folder and Navigator

The following Address Book widgets are highlighted in this illustration: `addressBook.Navigator.js`, `addressBook.PersonalBookBrowser.js`, `addressBook.ViewerContainer.js`, and `addressBook._BookBrowserToolbar.js`.

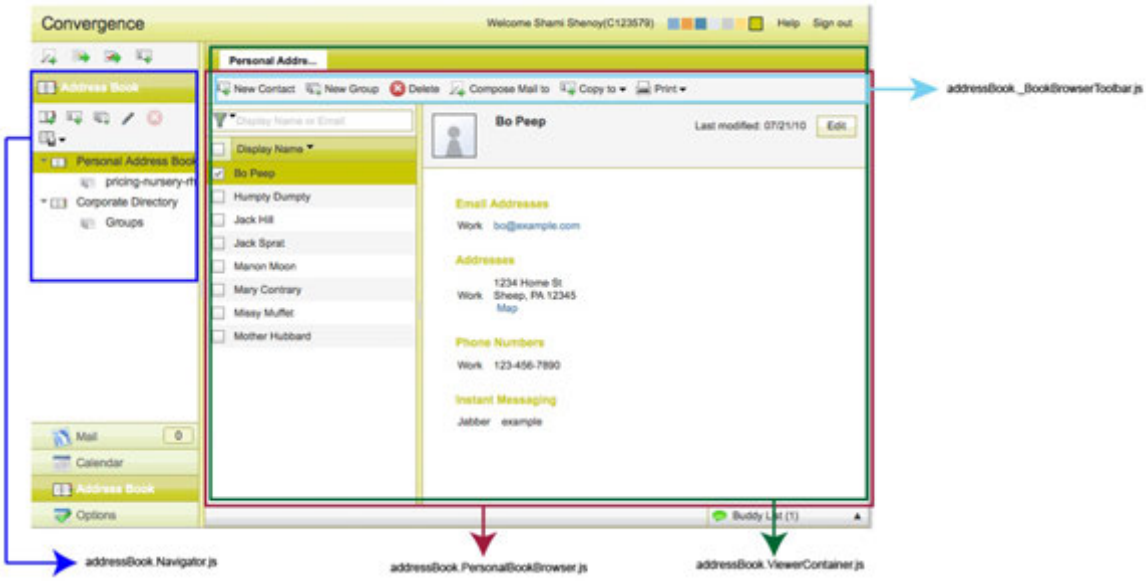


Figure 10: Address Book - Corporate Lookup

The `addressBook.CorporateBookBrowser.js` widget is highlighted in the following illustration:

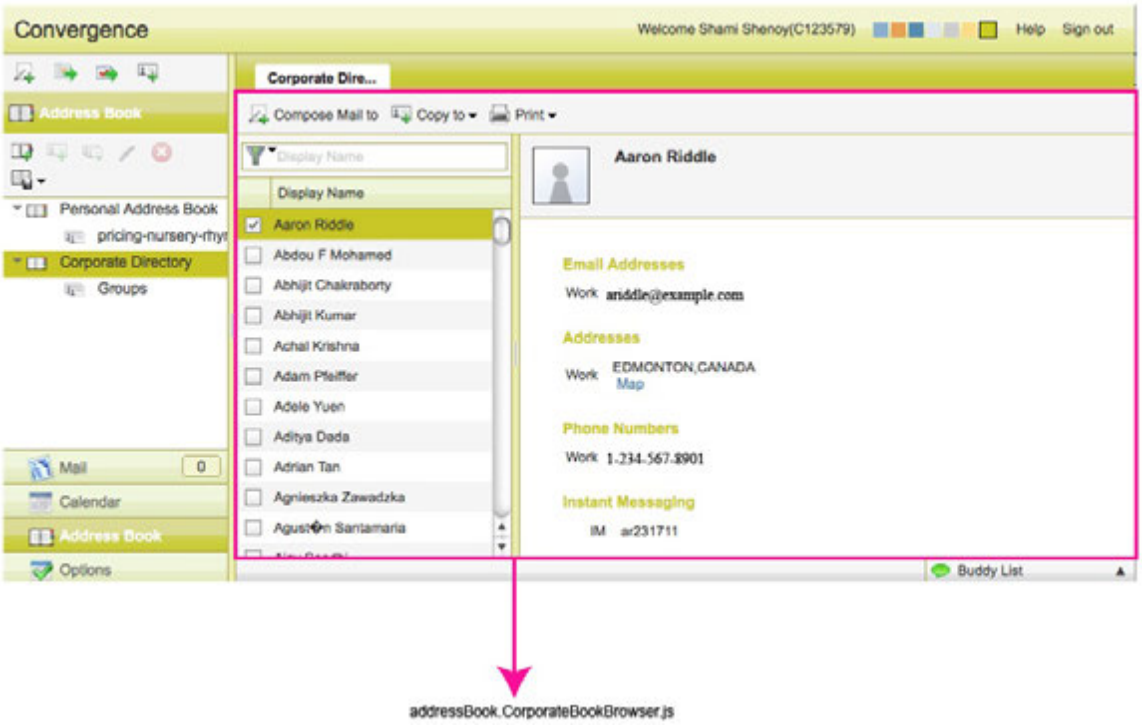
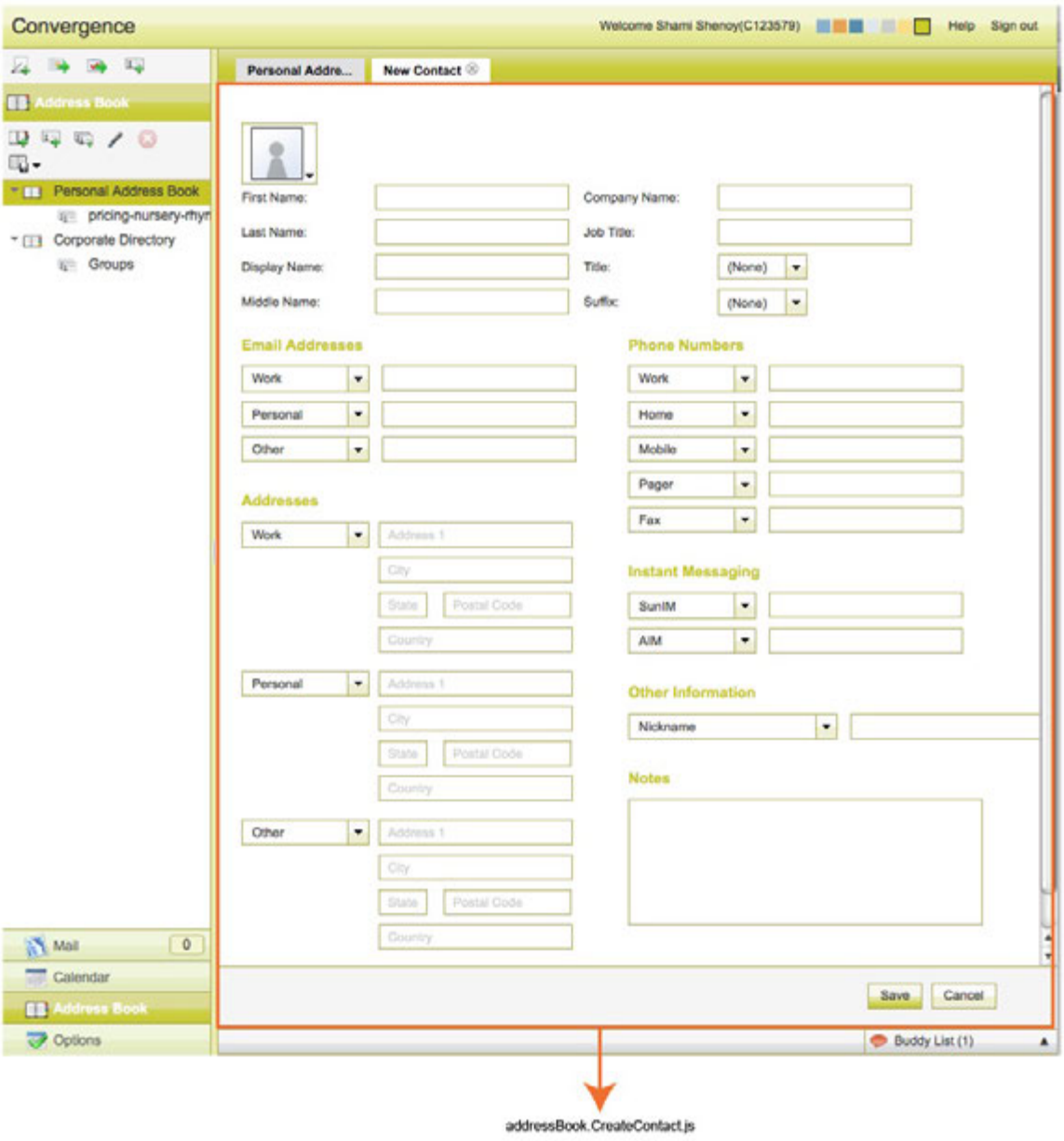


Figure 11: Address Book - Create Contact

The `addressBook.CreateContact.js` widget is highlighted in the following illustration:



;

Figure 12: Address Book - Create Group

The `addressBook.CreateGroup.js` widget is highlighted in the following illustration:

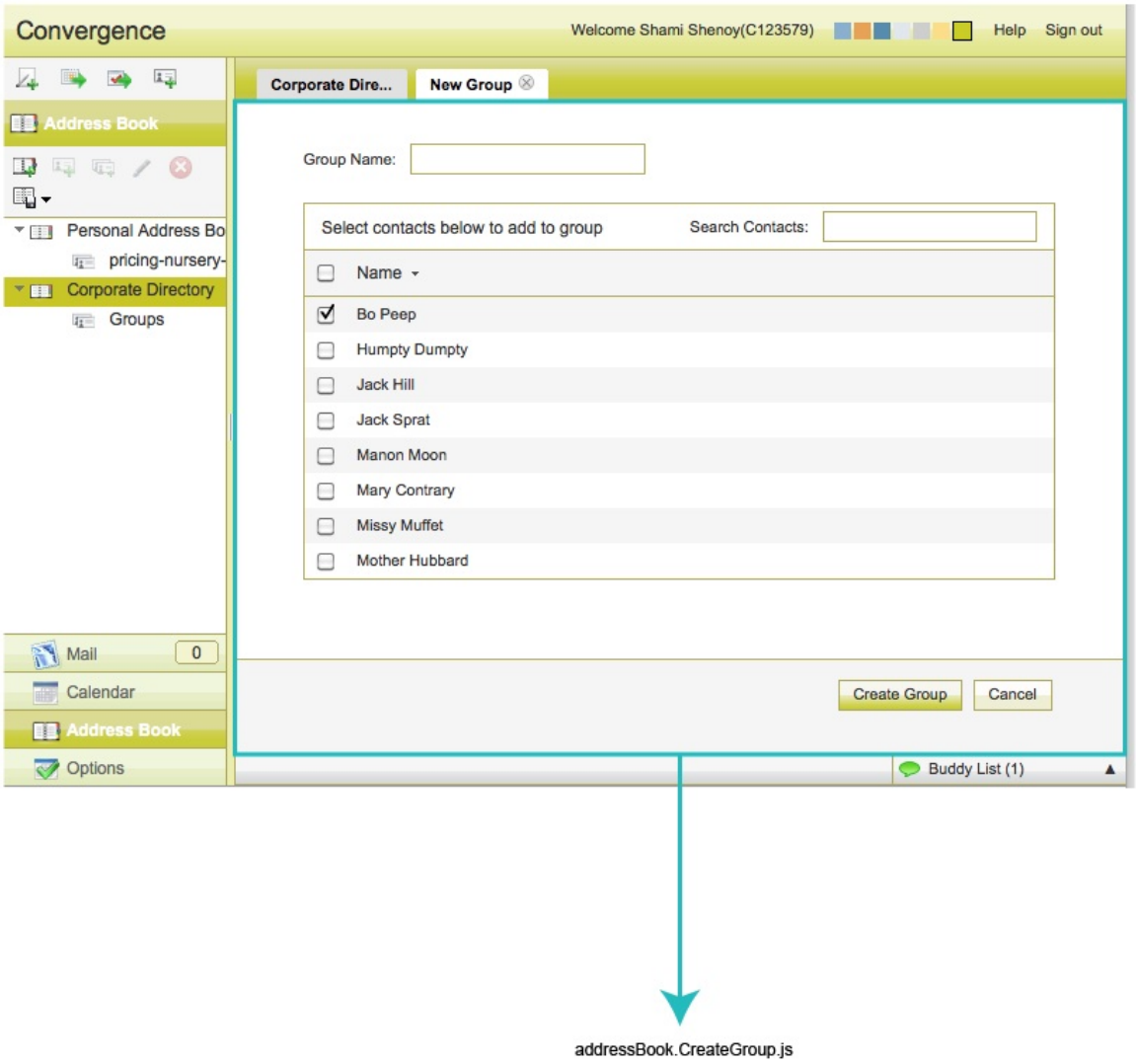


Figure 13: Address Book - Export Contact Dialog

The `addressBook.ExportContactsDialog.js` widget is highlighted in the following illustration:

Figure 14: Address Book - Import Contact Dialog

The `addressBook.ImportContactsDialog.js` widget is highlighted in the following illustration:

Figure 15: Address Book - Quick New Contact Dialog

The `addressBook.CreateContactDialog.js` widget is highlighted in the following illustration:

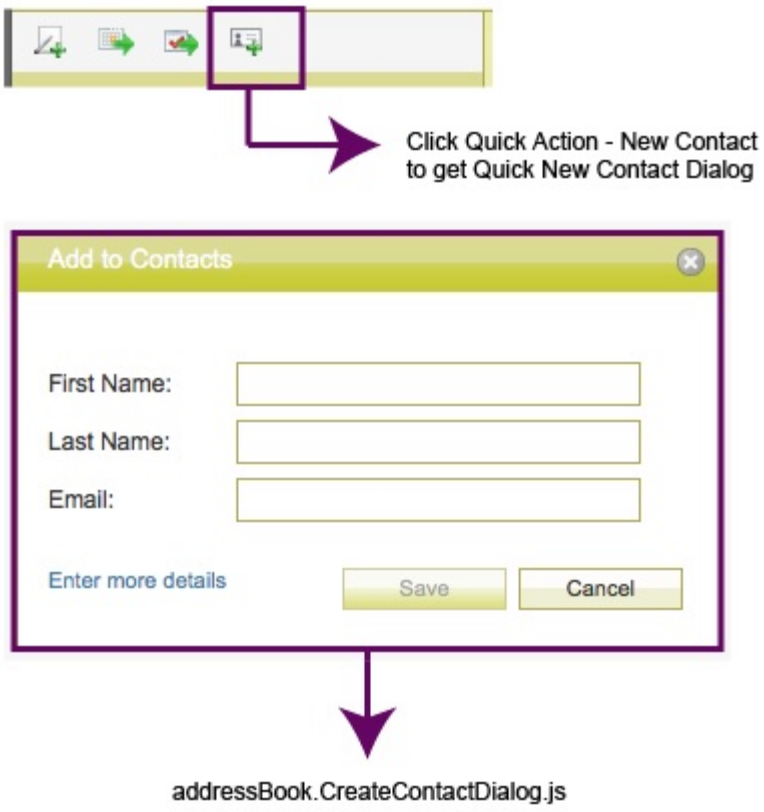


Figure 16: Address Book - Rename Group Dialog

The `addressBook.RenameGroupDialog.js` widget is highlighted in the following illustration:

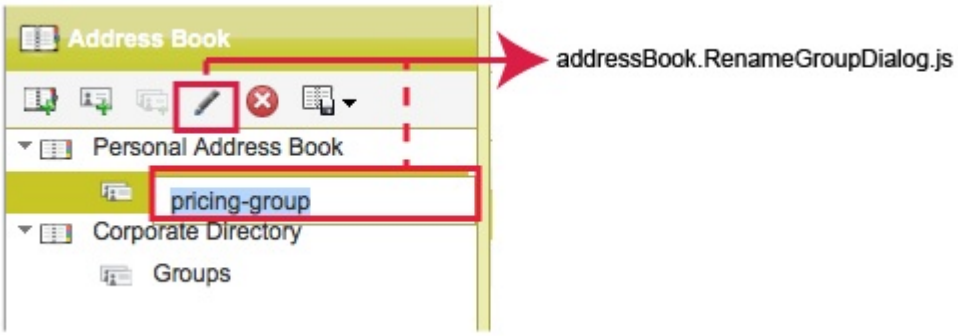


Figure 17: Address Book - Bookstore Item Selector

The `addressBook.BookStoreItemSelector.js` widget is highlighted in the following illustration:

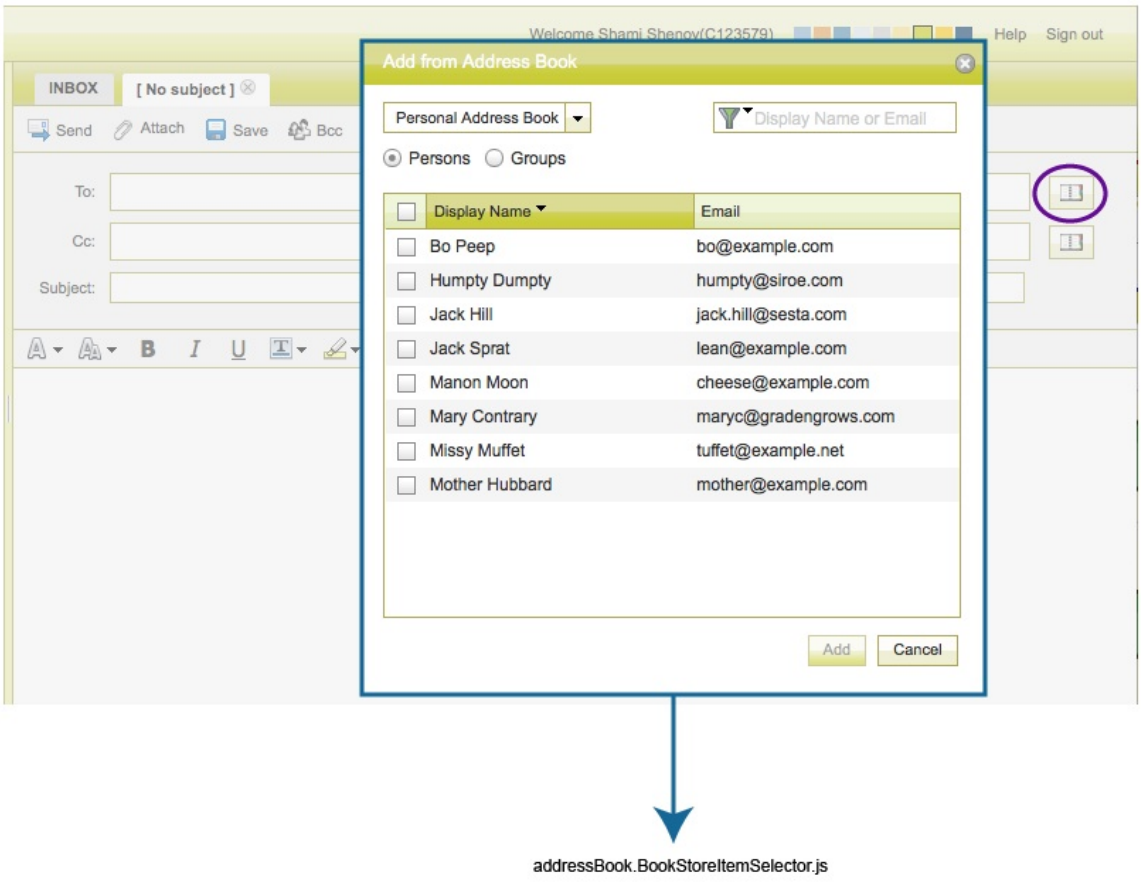
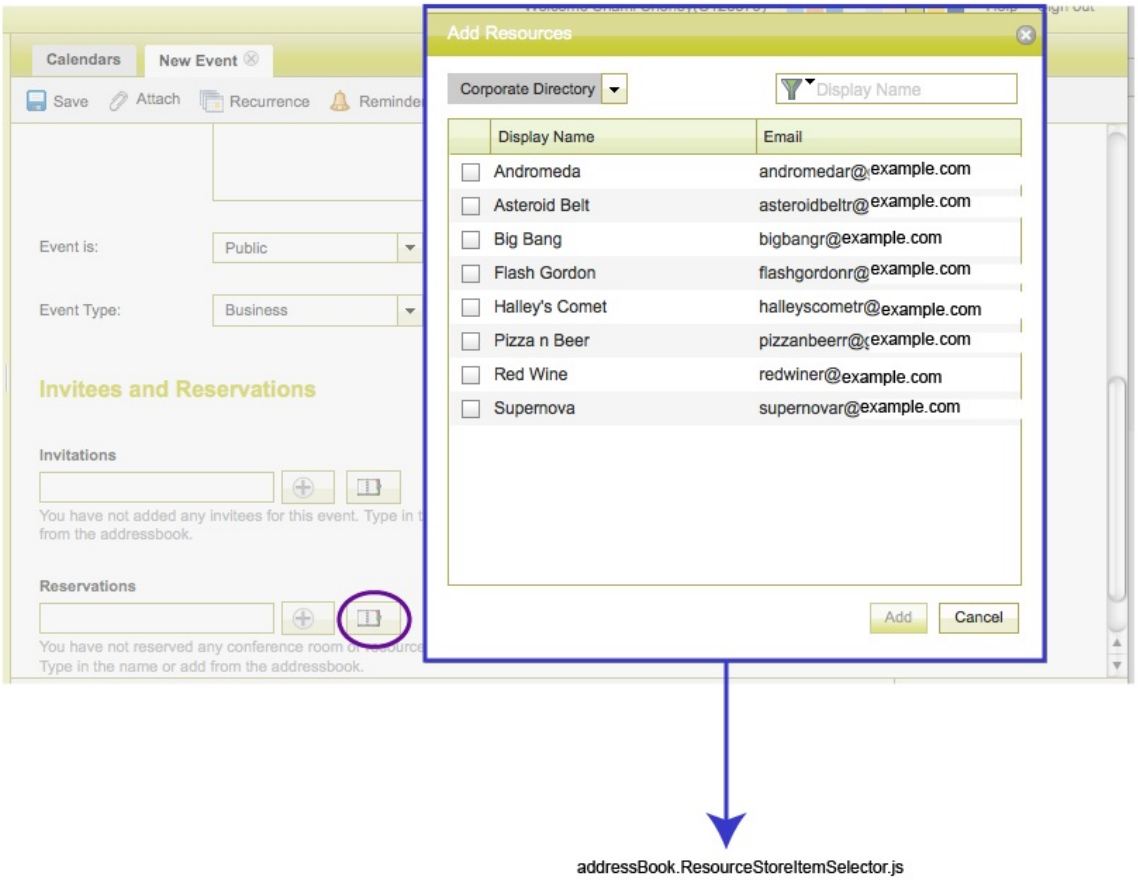


Figure 18: Address Book - Resource Store Item Selector

The `addressBook.ResourceStoreItemSelector.js` widget is highlighted in the following illustration:



Options Widgets

- calendar.option.Event
- calendar.option.General
- calendar.option.Notification
- im.option.General
- mail.option.Filter
- mail.option.FilterList
- mail.option.Forwarding
- mail.option.General
- mail.option.Layout
- mail.option.Identity
- mail.option.Vacation.Message
- option.General
- option.Navigator
- option.Password
- option.ViewerContainer

Figure 19: Options - Global General View and Options Navigator

The following Options widgets are highlighted in this illustration: `option.Navigator.js`, `option.ViewerContainer.js`, and `option.General.js`.

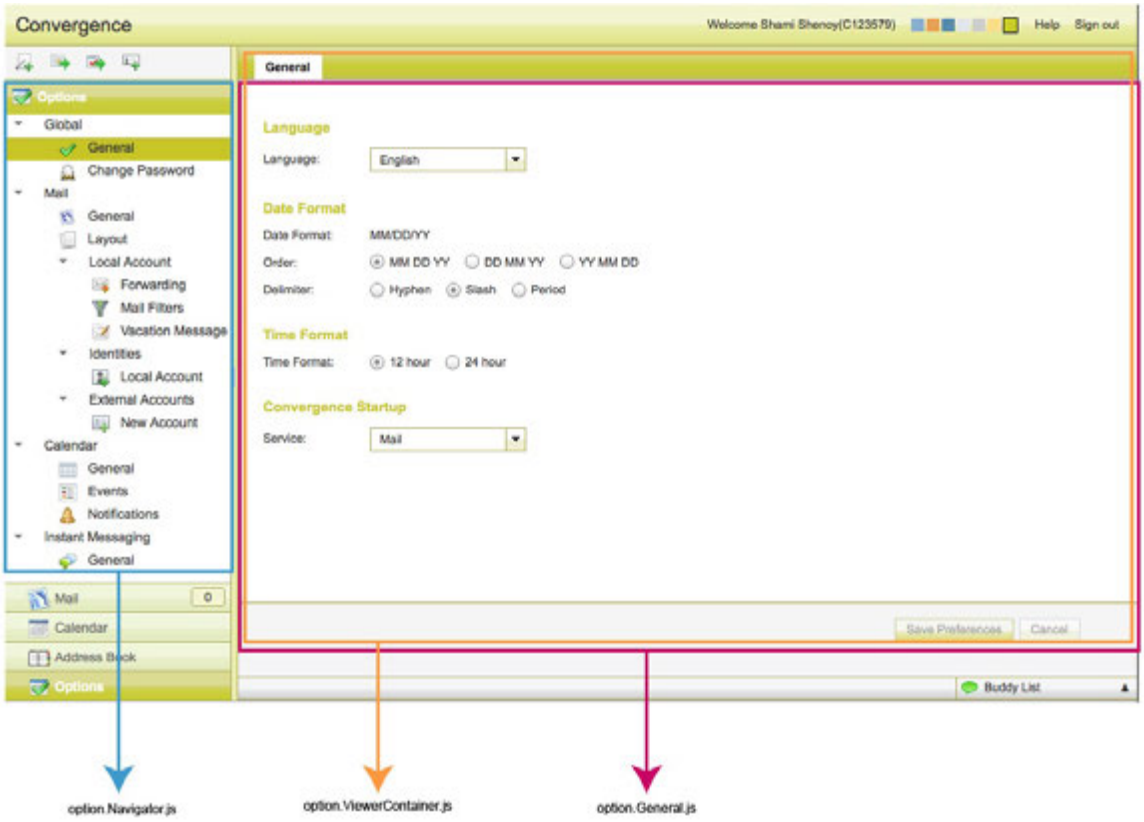


Figure 20: Options - Global Change Password

The `option.Password.js` widget is highlighted in the following illustration:

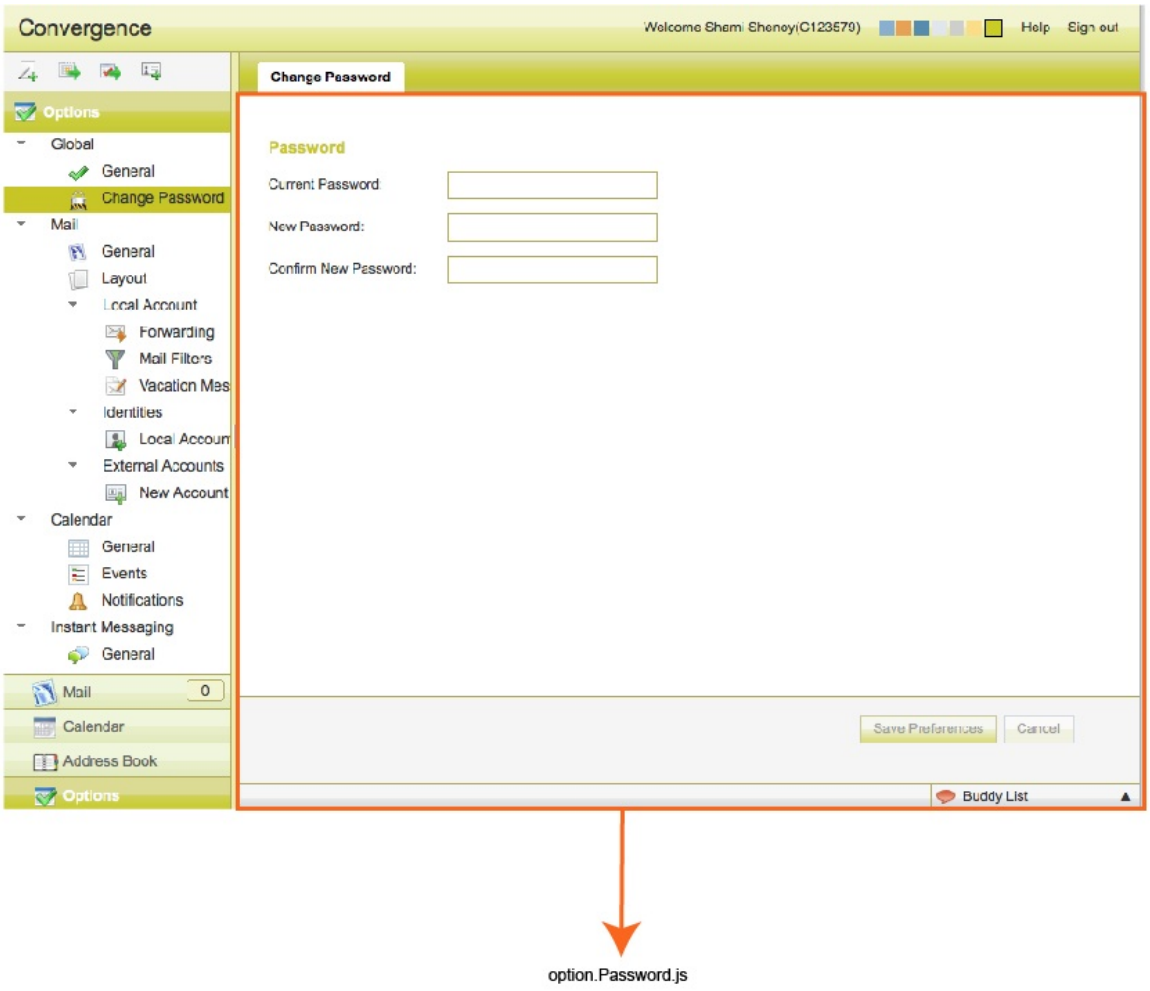


Figure 21: Options - Mail General View

The `mail.option.General.js` widget is highlighted in the following illustration:

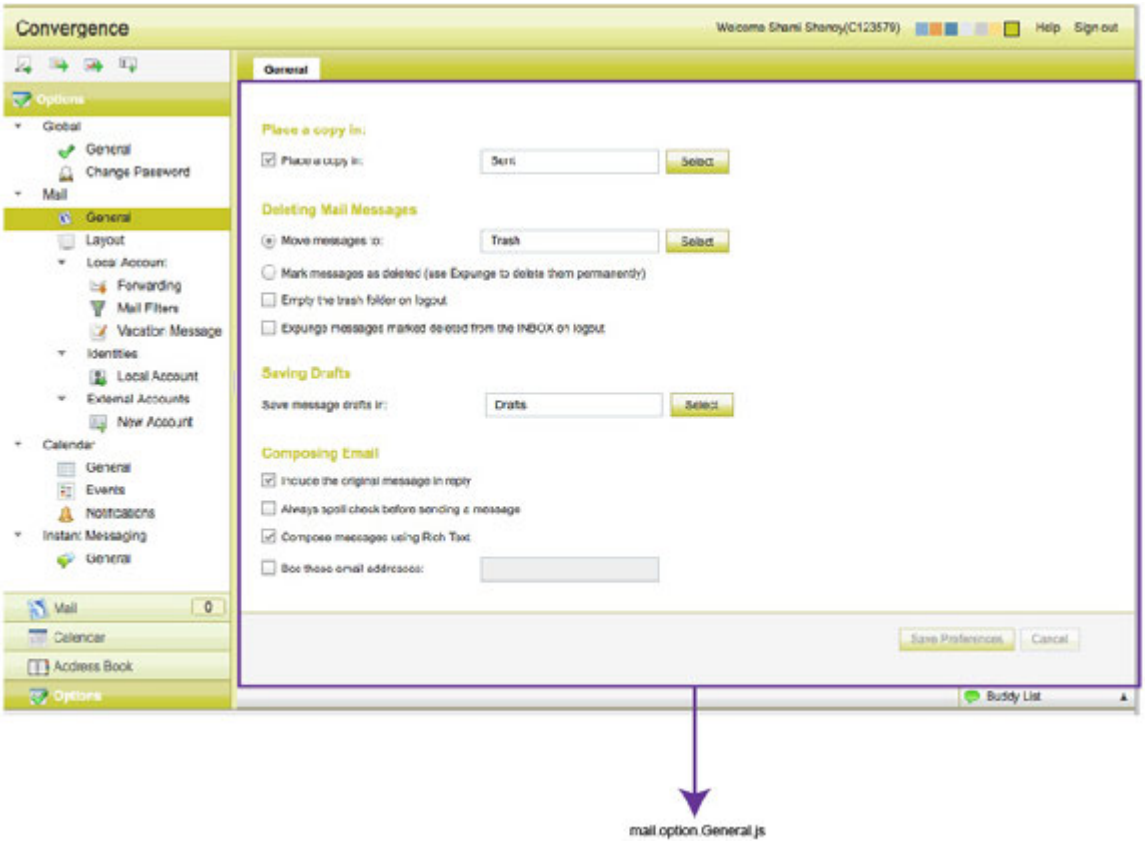


Figure 22: Options - Mail Personal Account Settings

The `mail.option.Identity.js` widget is highlighted in the following illustration:

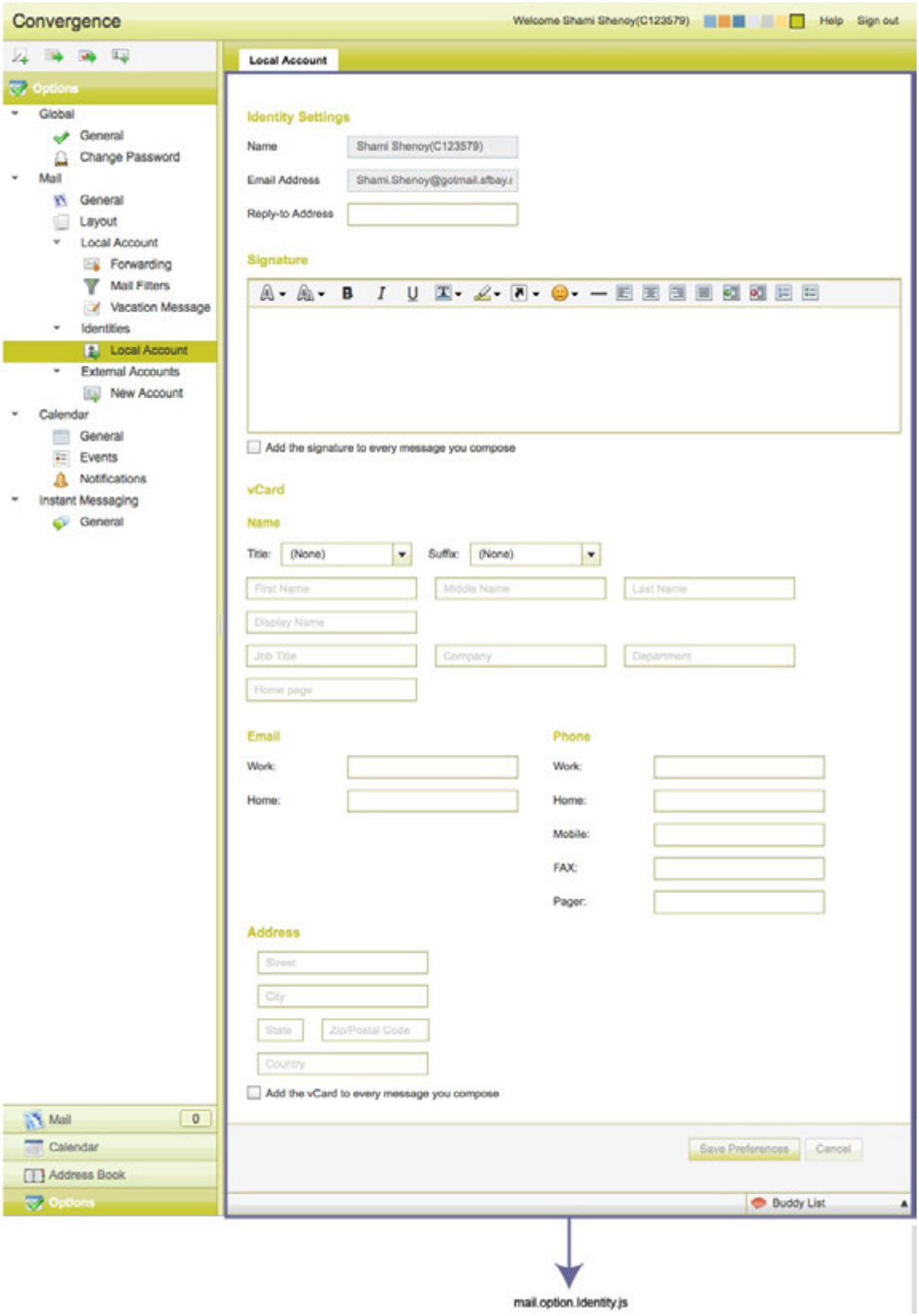


Figure 23: Options - Mail Fowarding

The `mail.option.Forwarding.js` widget is highlighted in the following illustration:

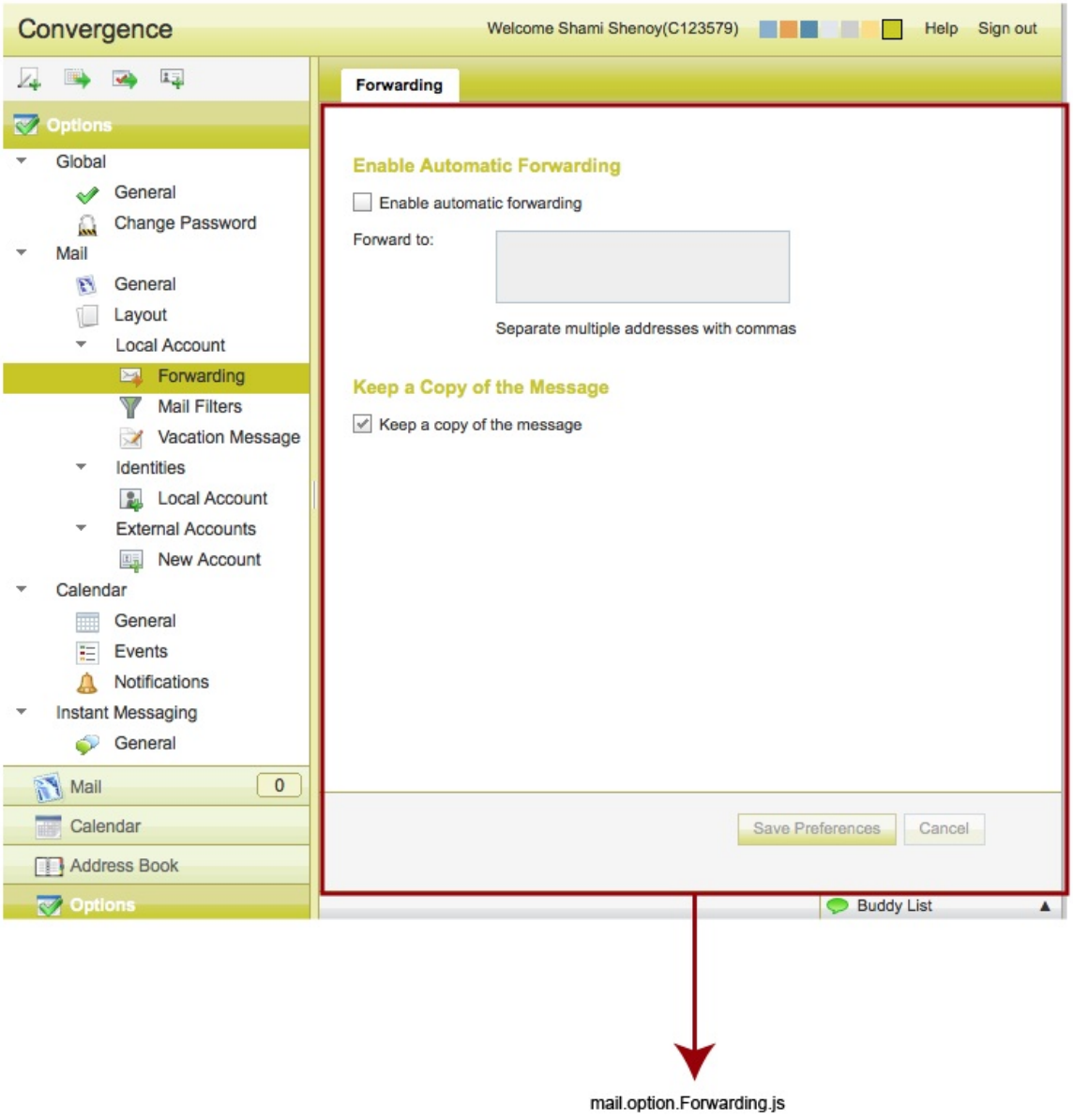


Figure 24: Options - Mail Layout

The `mail.option.Layout.js` widget is highlighted in the following illustration:

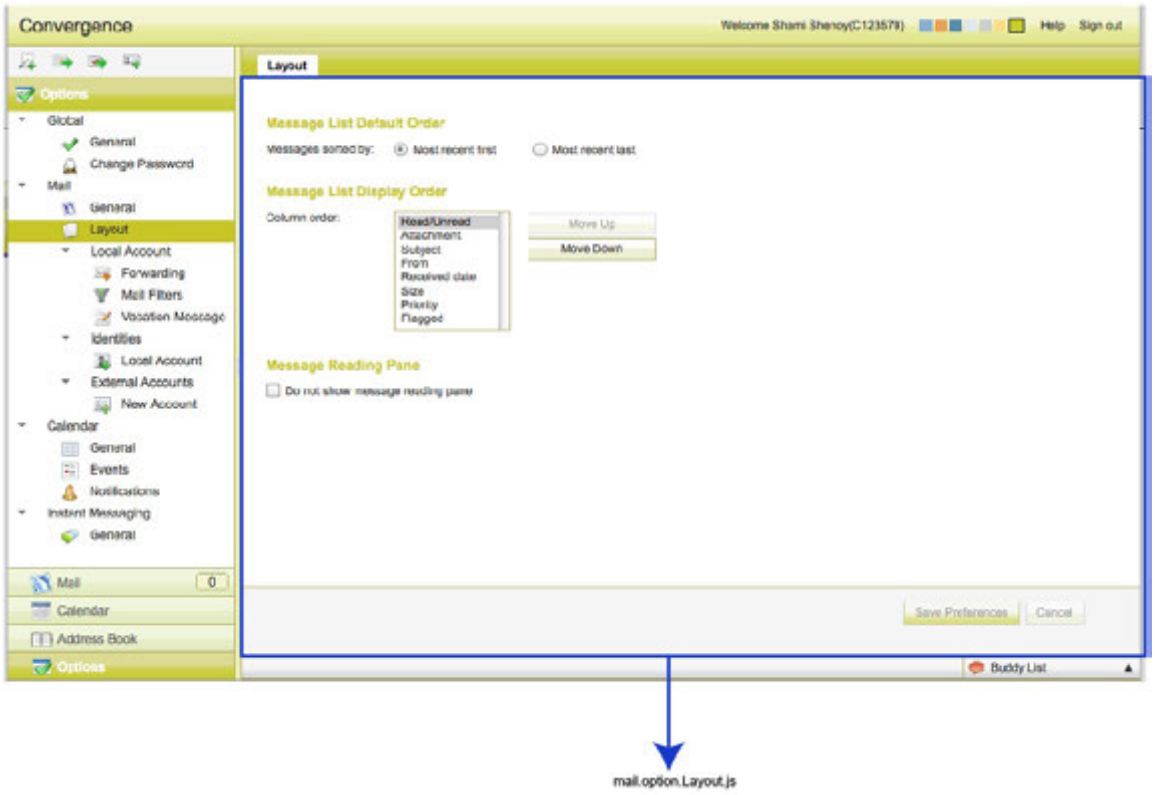


Figure 25: Options - New Mail Filter

The `mail.option.Filter.js` widget is highlighted in the following illustration:

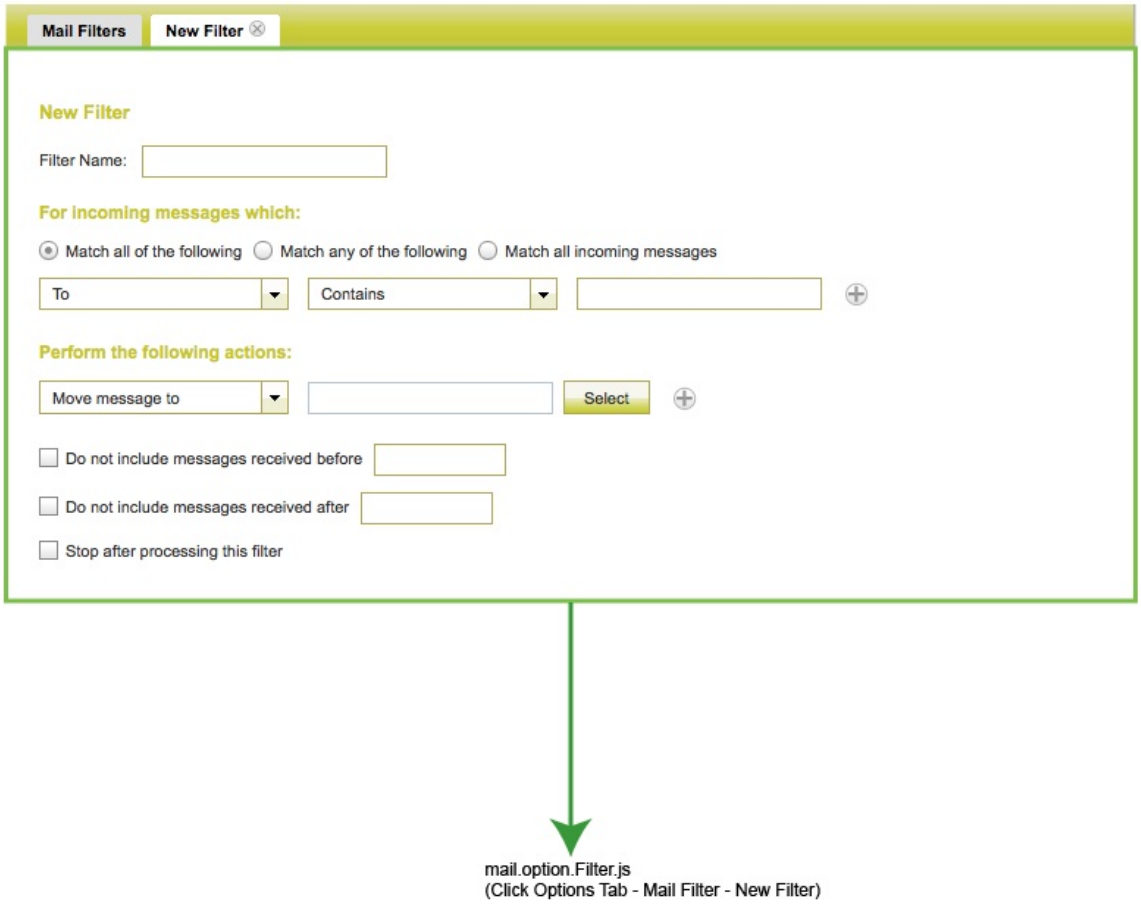


Figure 26: Options - Mail Filters

The `mail.option.FilterList.js` widget is highlighted in the following illustration:

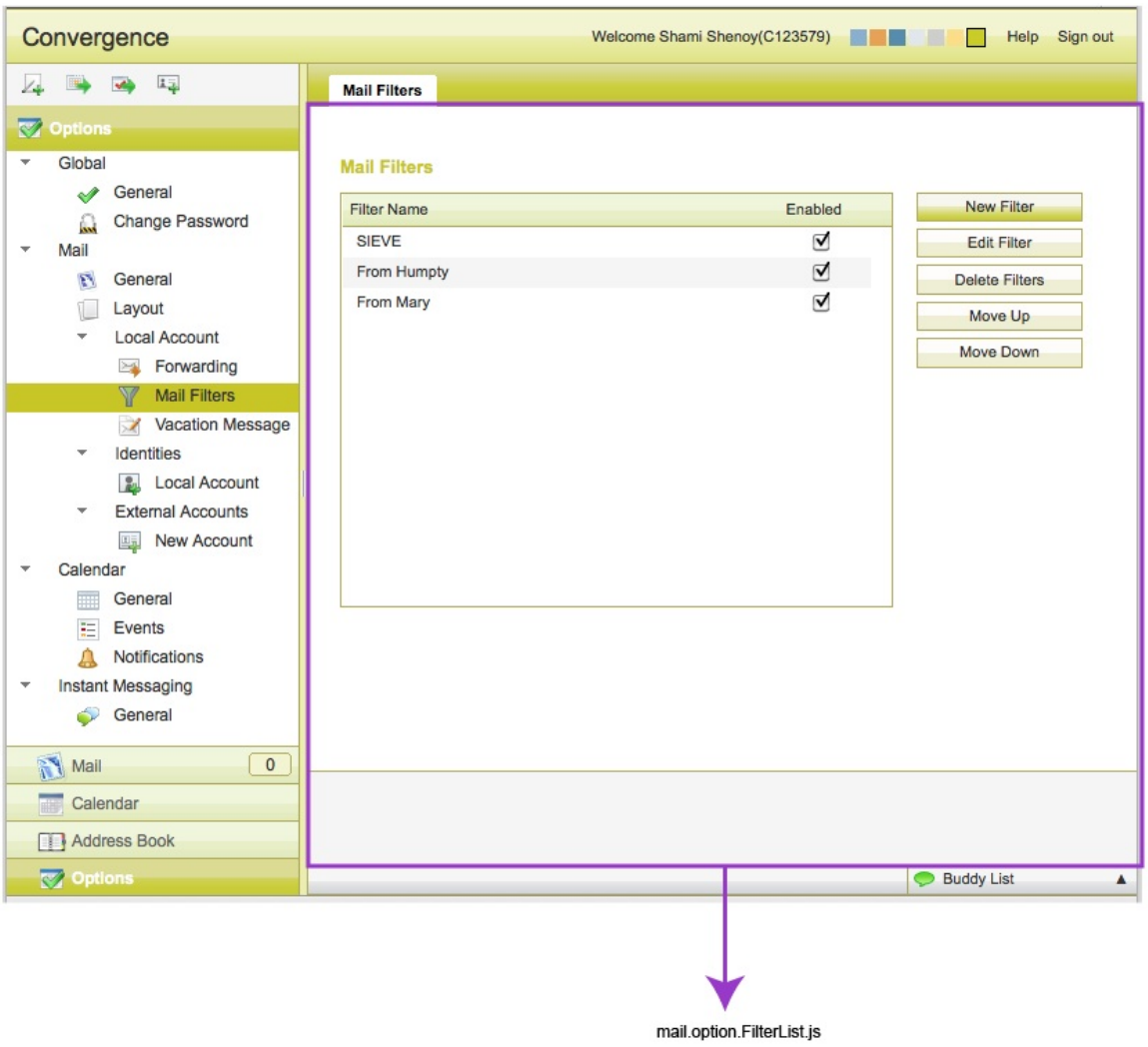


Figure 27: Options - Mail Vacation Message

The `mail.option.VacationMessage.js` widget is highlighted in the following illustration:

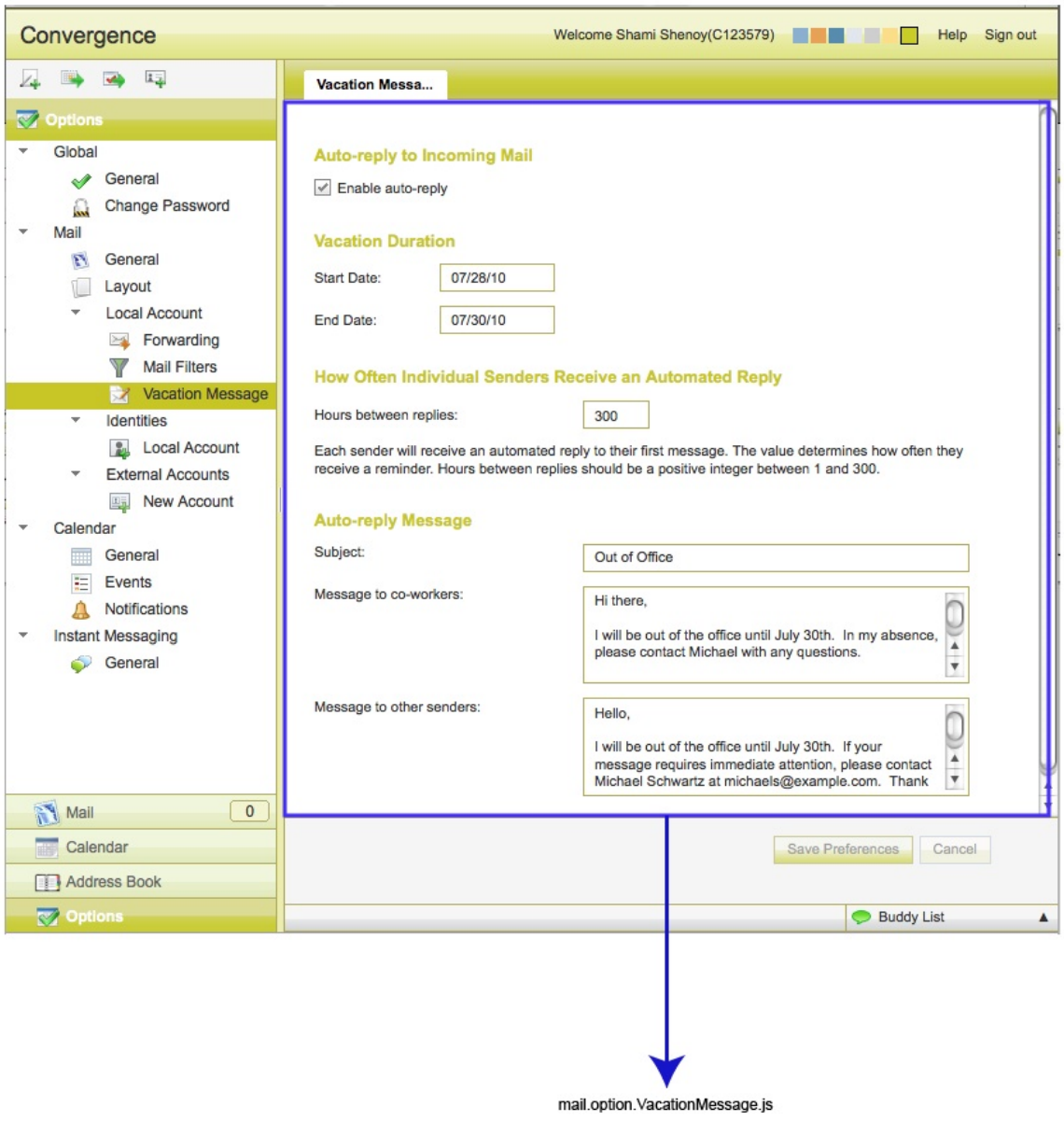


Figure 28: Options - Calendar General

The `calendar.option.General.js` widget is highlighted in the following illustration:

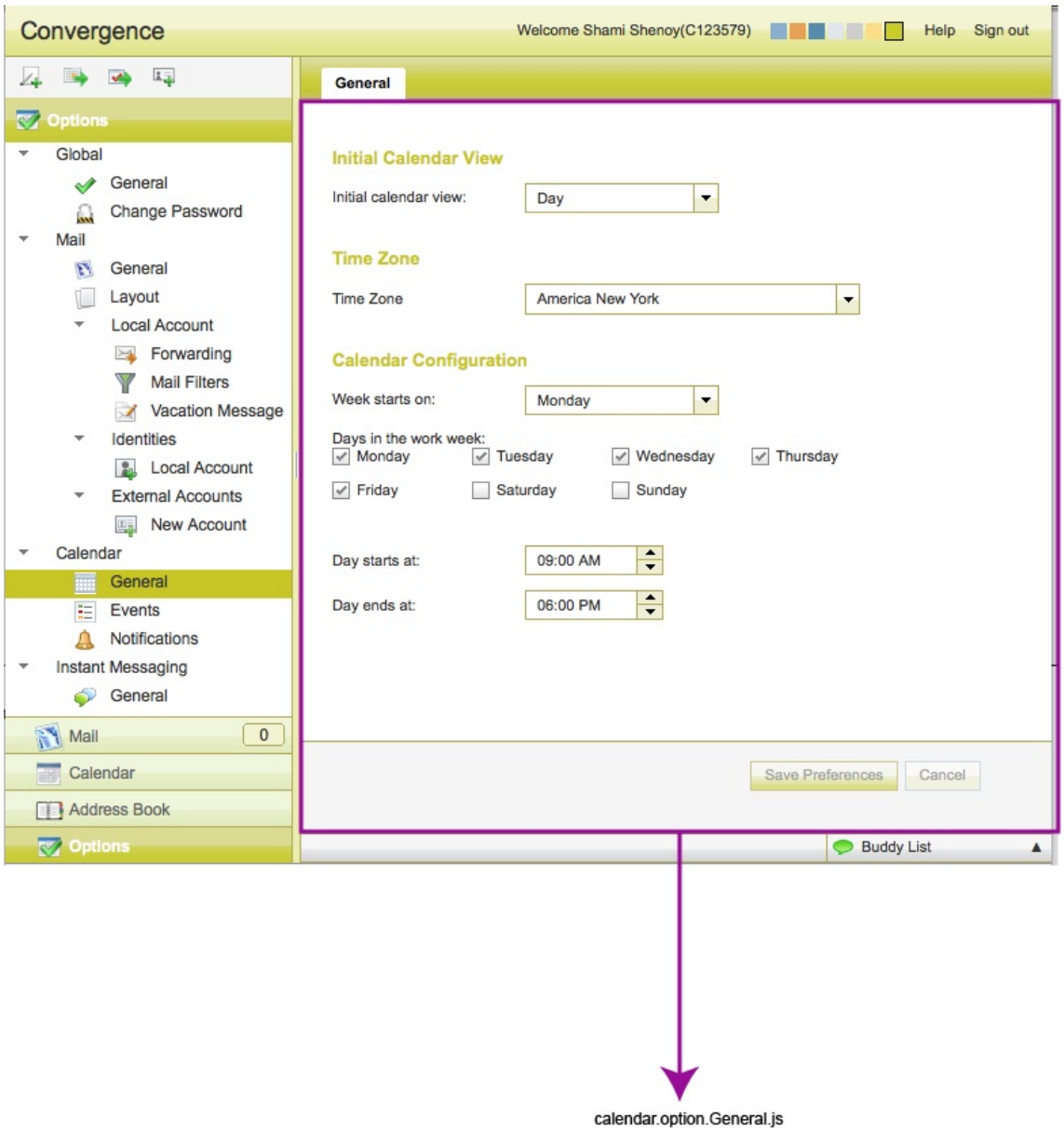


Figure 29: Options - Calendar Events

The `calendar.option.Event.js` widget is highlighted in the following illustration:

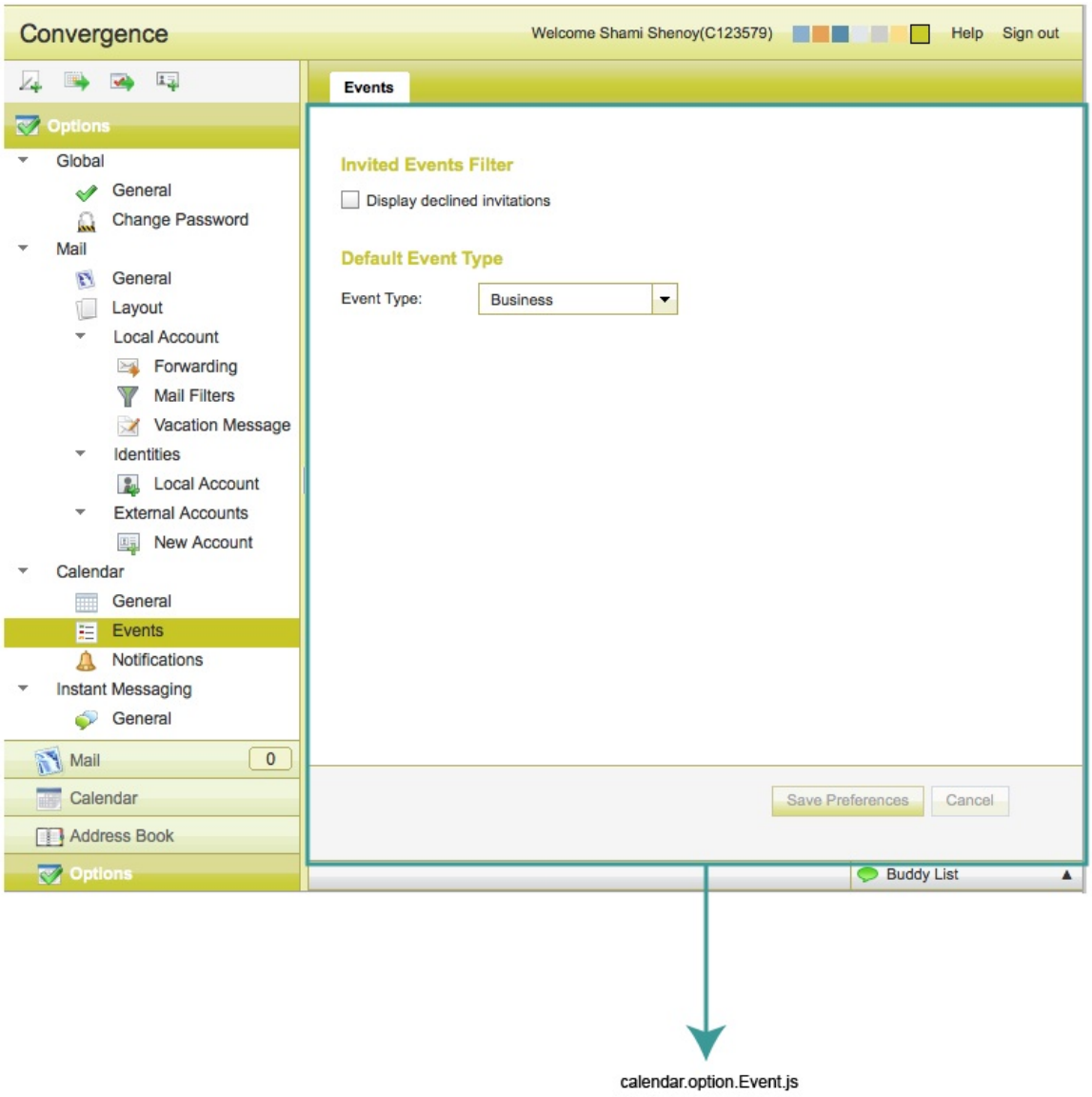


Figure 30: Options - Calendar Notifications

The `calendar.option.Notification.js` widget is highlighted in the following illustration:

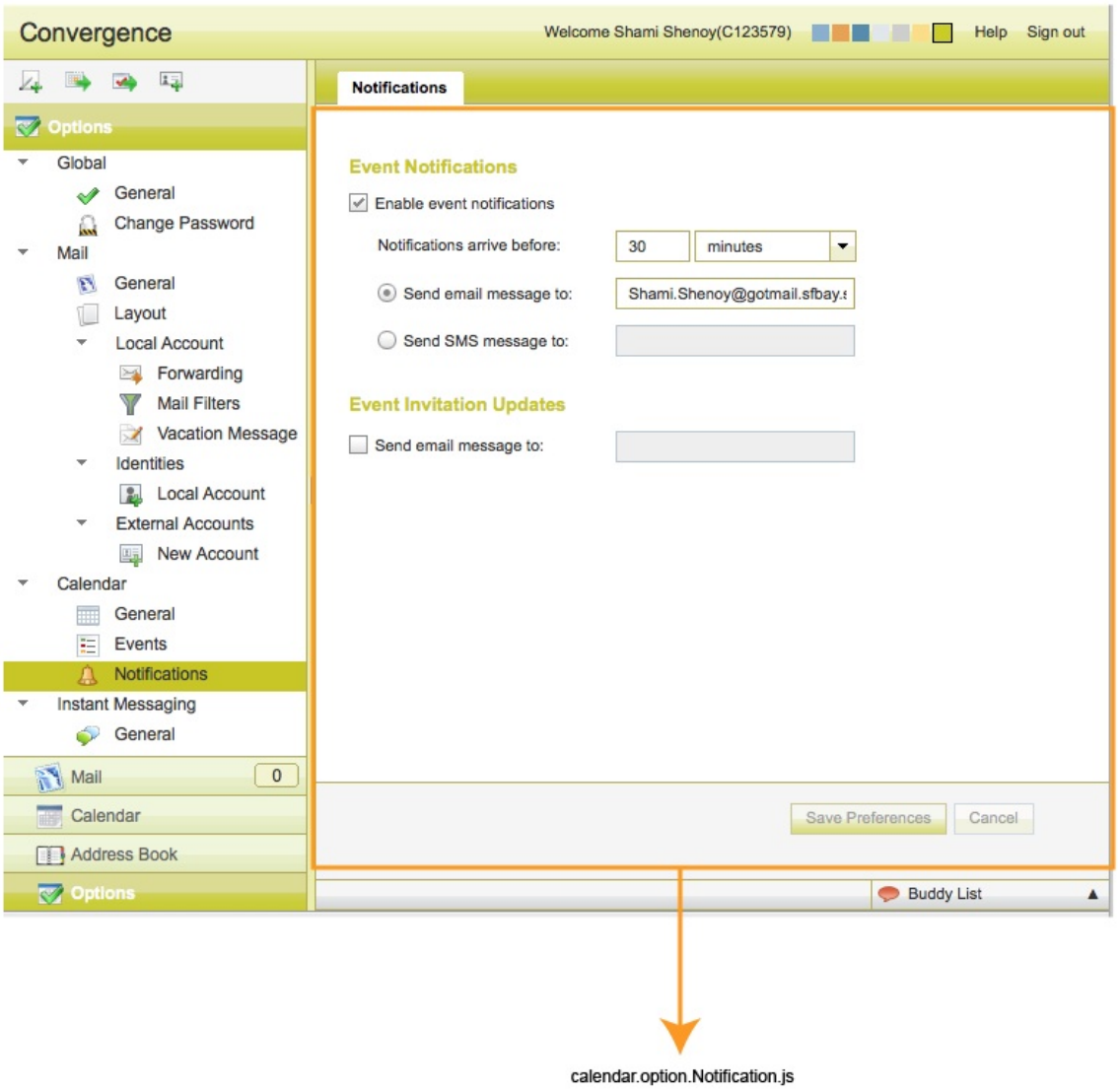
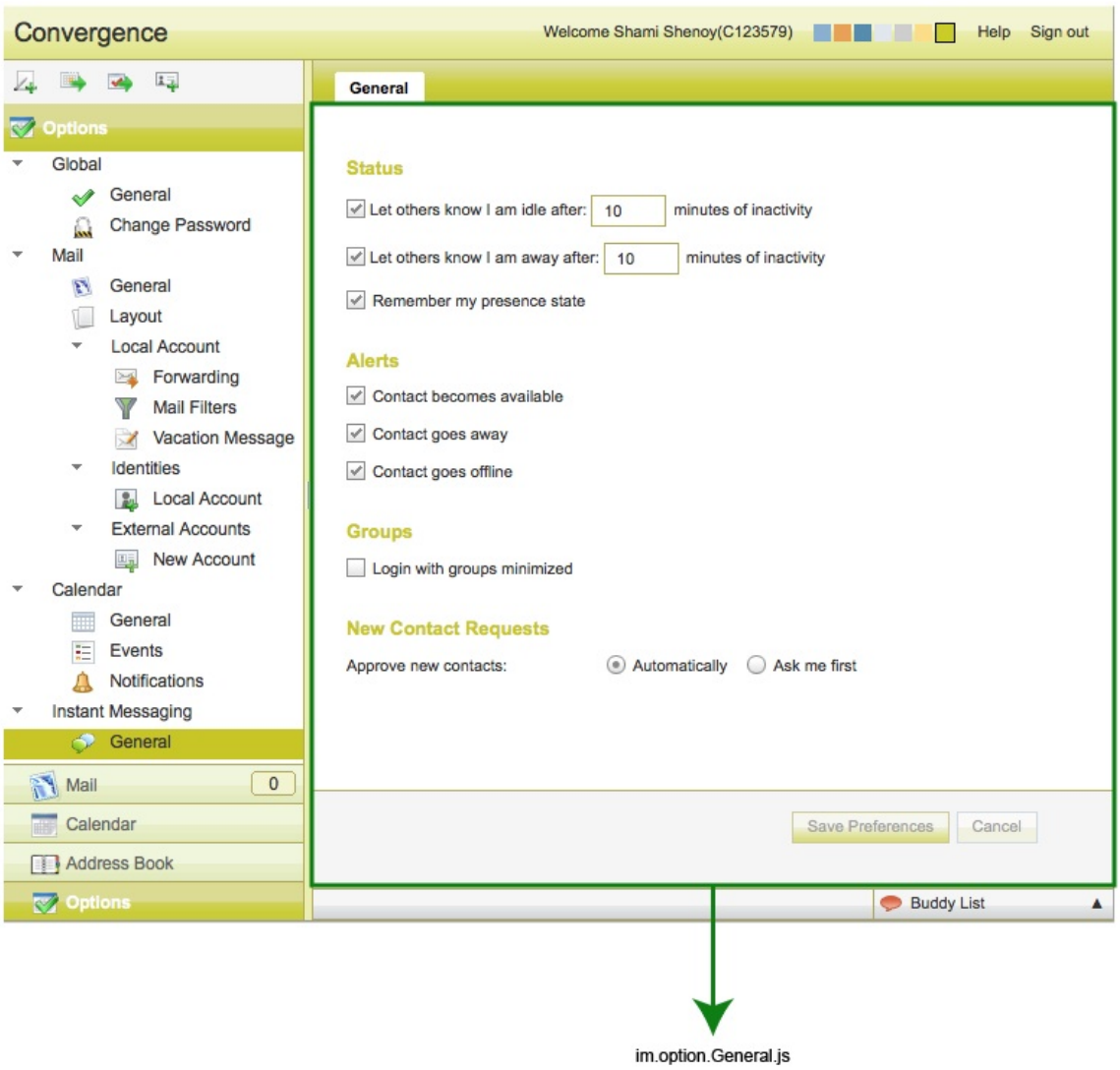


Figure 31: Options - IM General

The `im.option.General.js` widget is highlighted in the following illustration:



Calendar Widgets

Listed Alphabetically

- calendar.Availability
- calendar.CalendarPropertiesDialog
- calendar.CreateEvent
- calendar.CreateTaskDialog
- calendar.DayView
- calendar.Event
- calendar.EventBalloon
- calendar.ExportDialog
- calendar.ImportDialog
- calendar.Invitees
- calendar.ListView
- calendar.monthlyEvent
- calendar.MonthView
- calendar.Navigator
- calendar.Next7View
- calendar.NotificationDialog

- calendar.Print
- calendar.PrintDialog
- calendar.quickEventBalloon
- calendar.RecurrenceDialog
- calendar.Subscribe
- calendar.TaskDetail
- calendar.TimezoneDialog
- calendar.ViewerContainer
- calendar.ViewDispatcher
- calendar.ViewEvent
- calendar.ViewEventItem
- calendar.ViewInviteesItem
- calendar.ViewTaskItem
- calendar.WeekView
- digit._Calendar

Listed By Function

Calendar Views

The following calendar views call out the most common regions and widgets in the UI:

Figure 32: Calendar - Viewer, Daily View & Navigator

The following Calendar widgets are highlighted in this illustration: `calendar.Navigator.js`, `digit._Calendar.js`, `calendar.Event.js`, `calendar.ViewDispatcher.js`, `calendar.DayView.js`, and `calendar.ViewerContainer.js`.

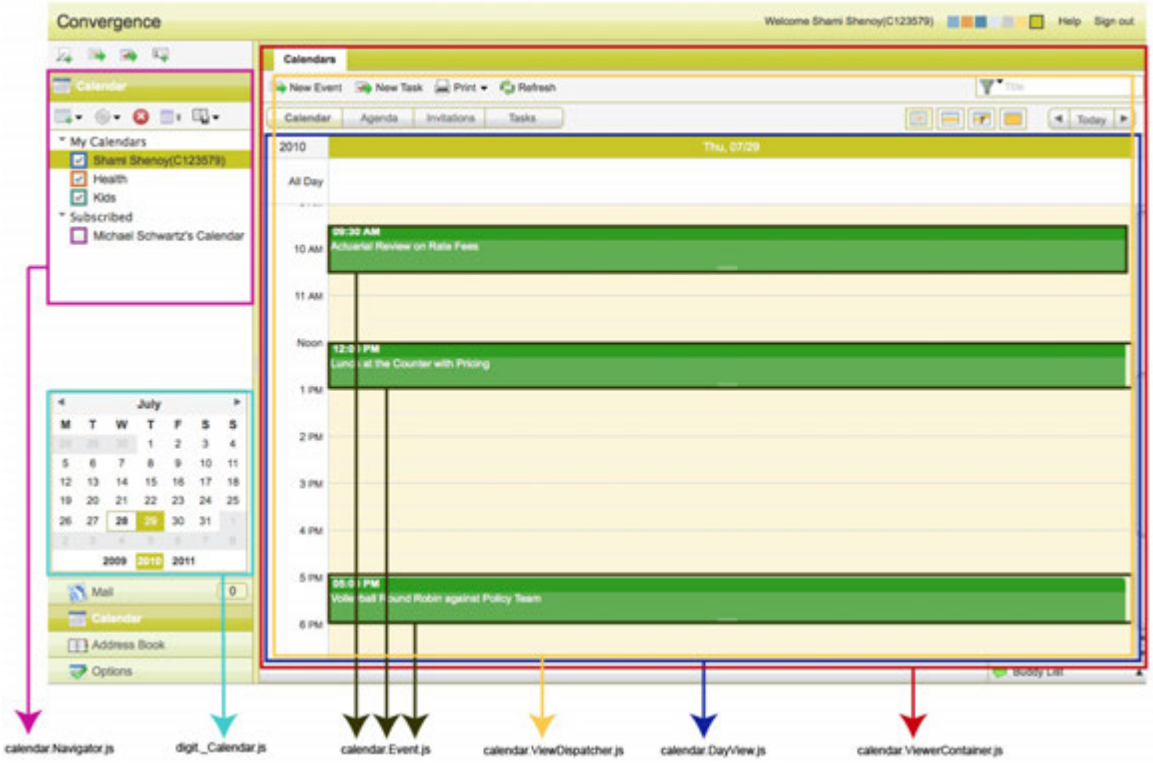


Figure 33: Calendar - Month View

The `calendar.MonthView.js` widget is highlighted in the following illustration:

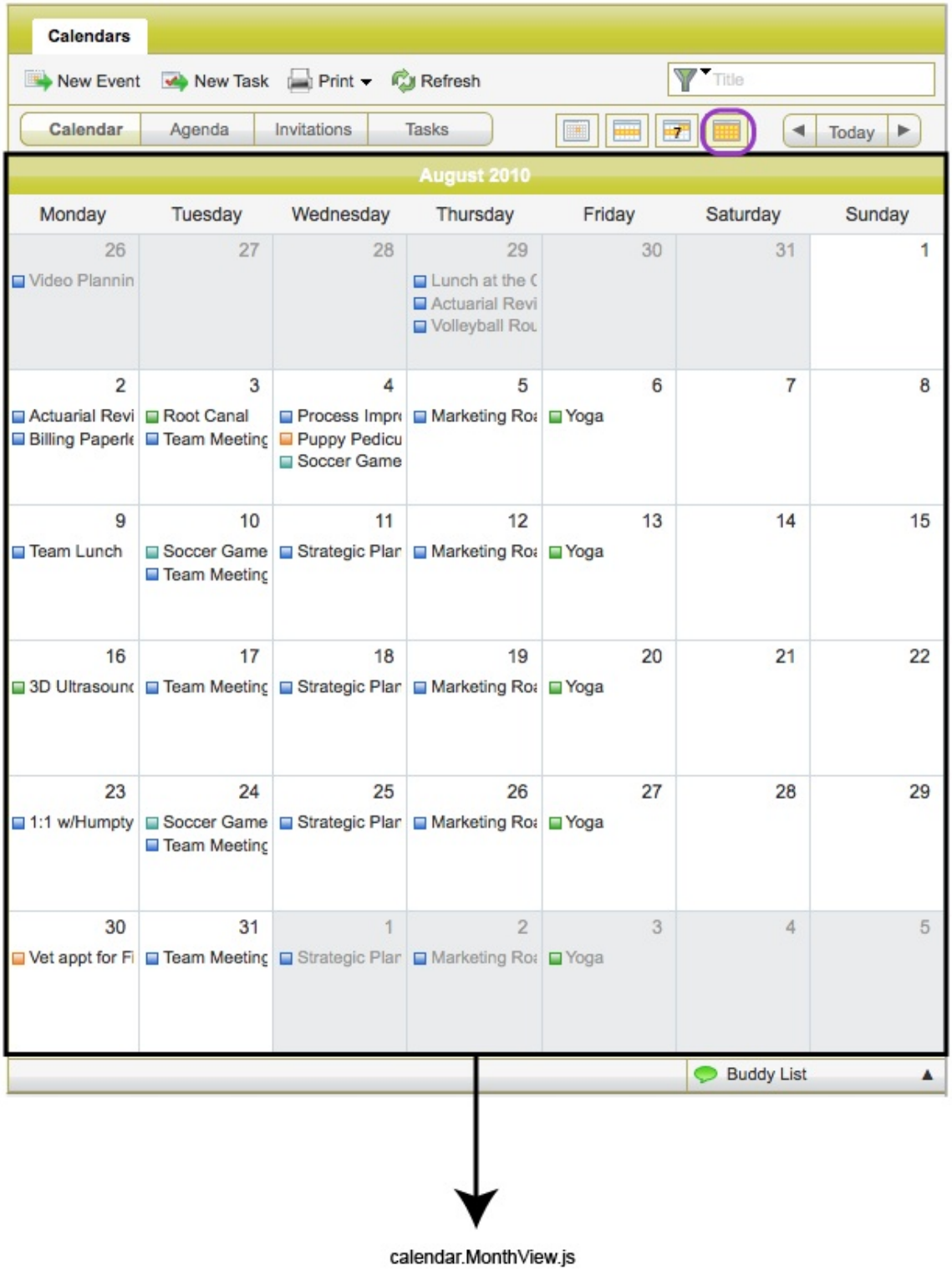


Figure 34: Calendar - Weekly View

The `calendar.WeekView.js` widget is highlighted in the following illustration:

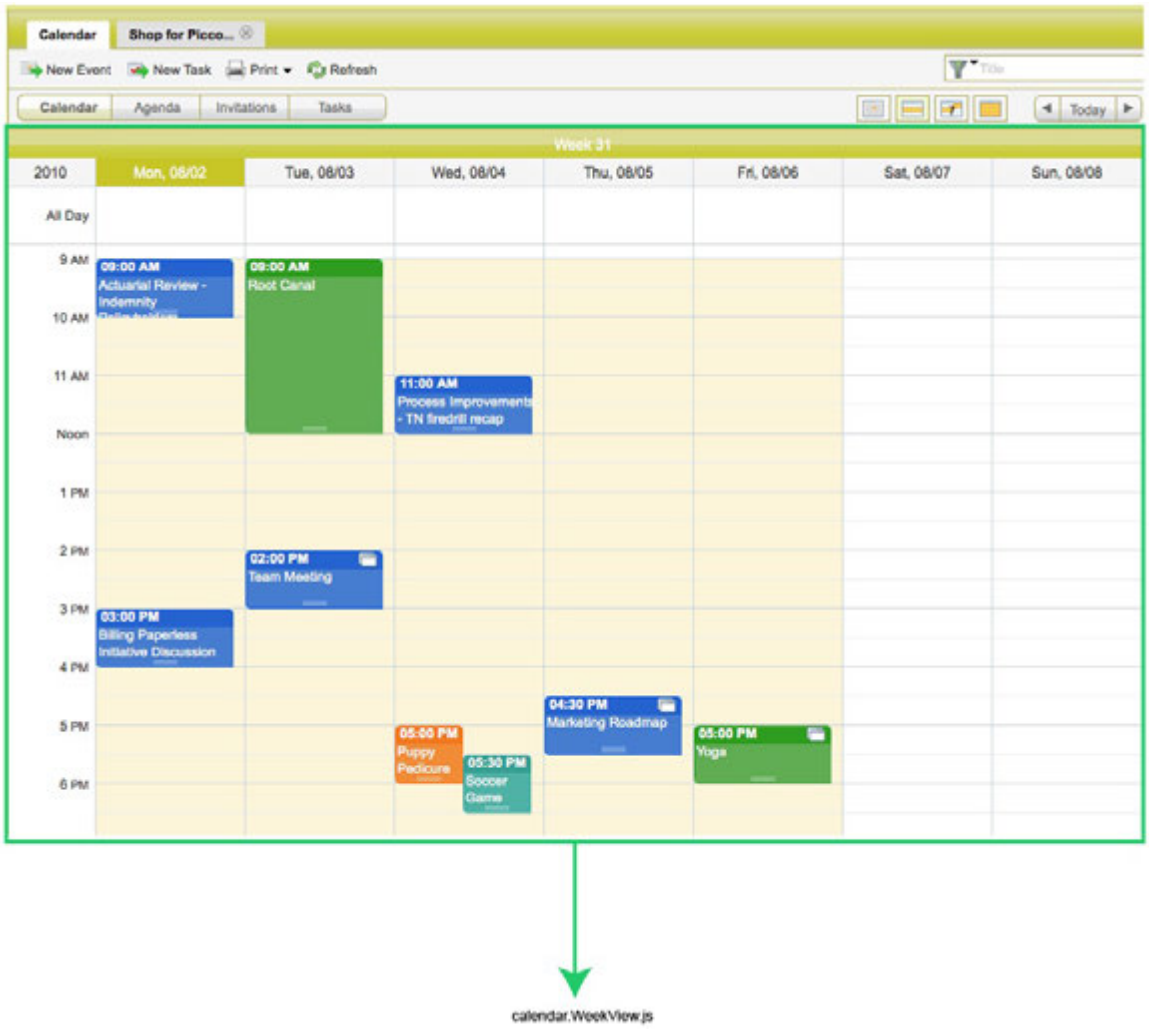


Figure 35: Calendar - Next 7 View

The `calendar.Next7View.js` widget is highlighted in the following illustration:

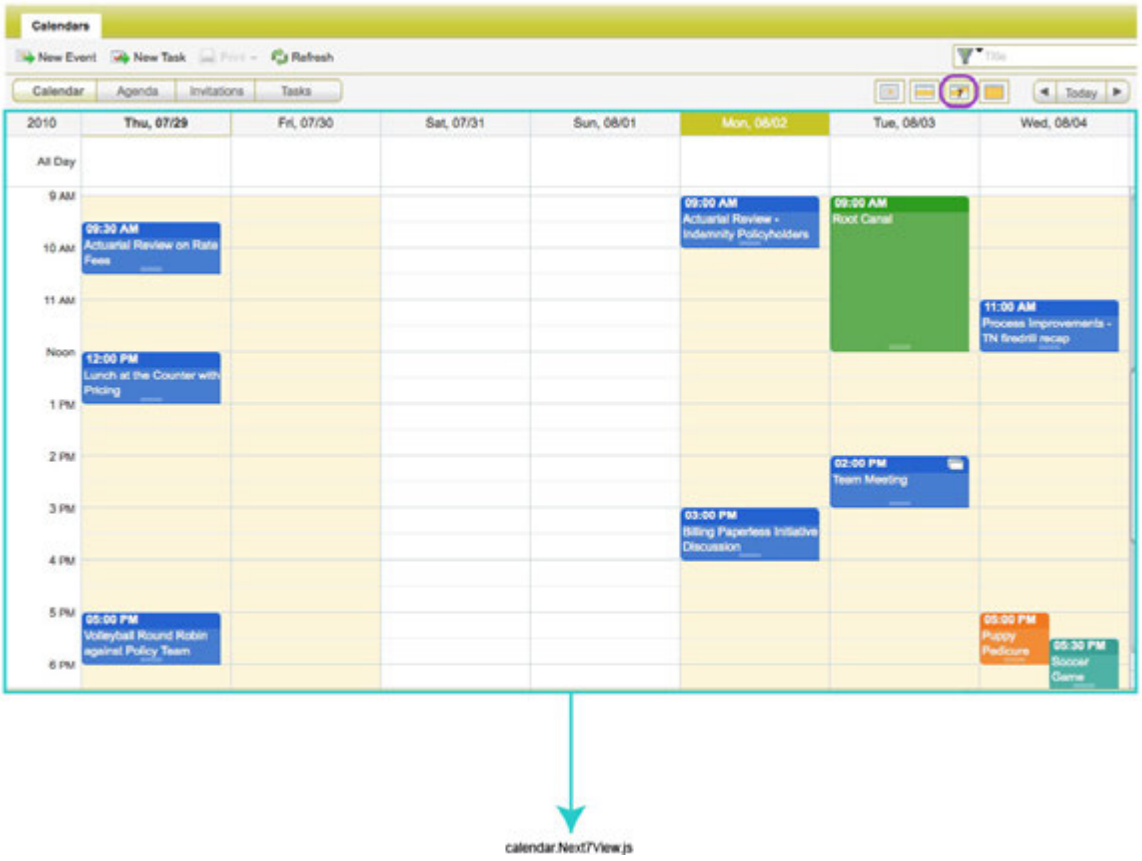


Figure 36: Calendar - List View (for Agenda) and View Event Item

The `calendar.ViewEventItem.js` and `calendar.ListView.js` widgets are highlighted in the following illustration:

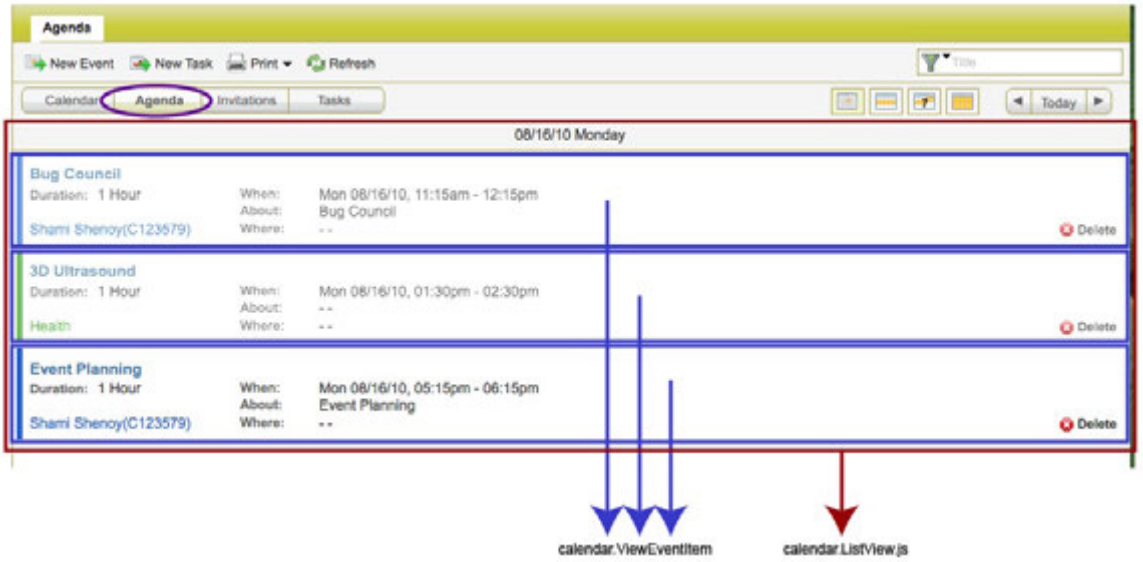


Figure 37: Calendar - List View (for Invitations) and View Invitees Item

The `calendar.ViewInvitesItem.js` and `calendar.ListView.js` widgets are highlighted in the following illustration:

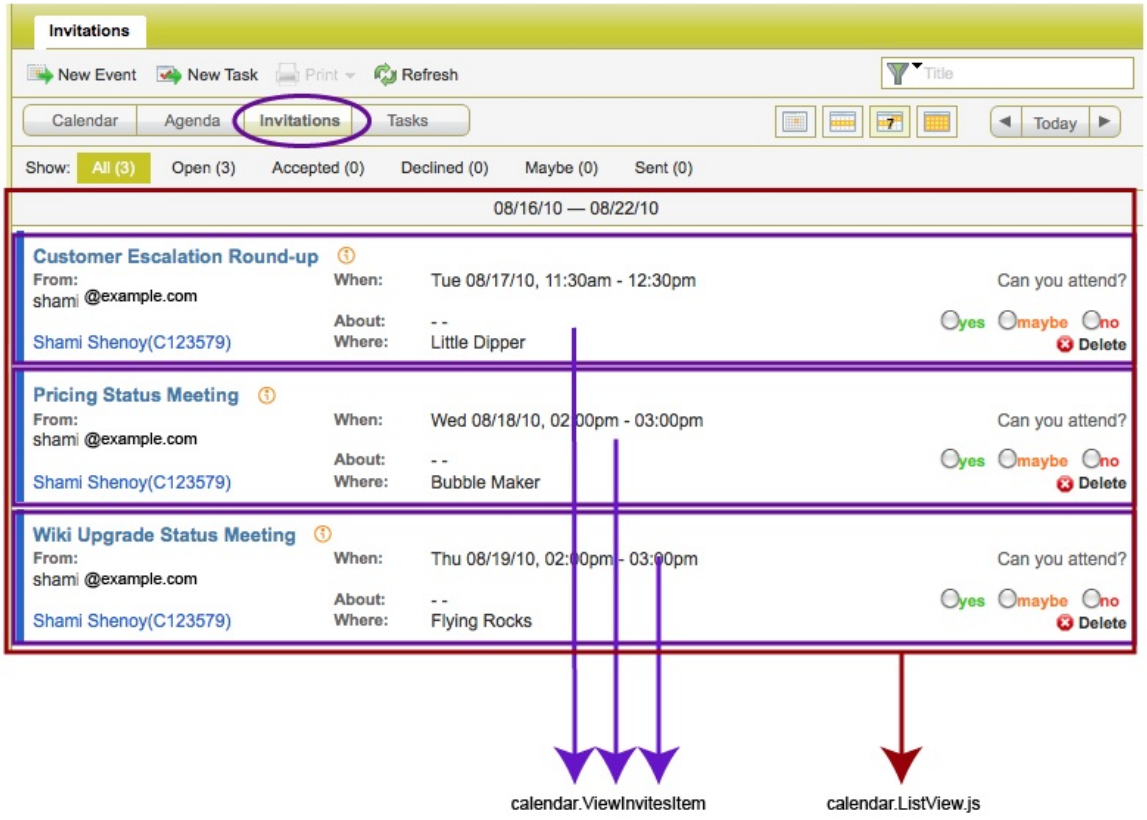


Figure 38: Calendar - List View (for Tasks) and View Task Item

The `calendar.ViewTaskItem.js` and `calendar.ListView.js` widgets are highlighted in the following illustration:

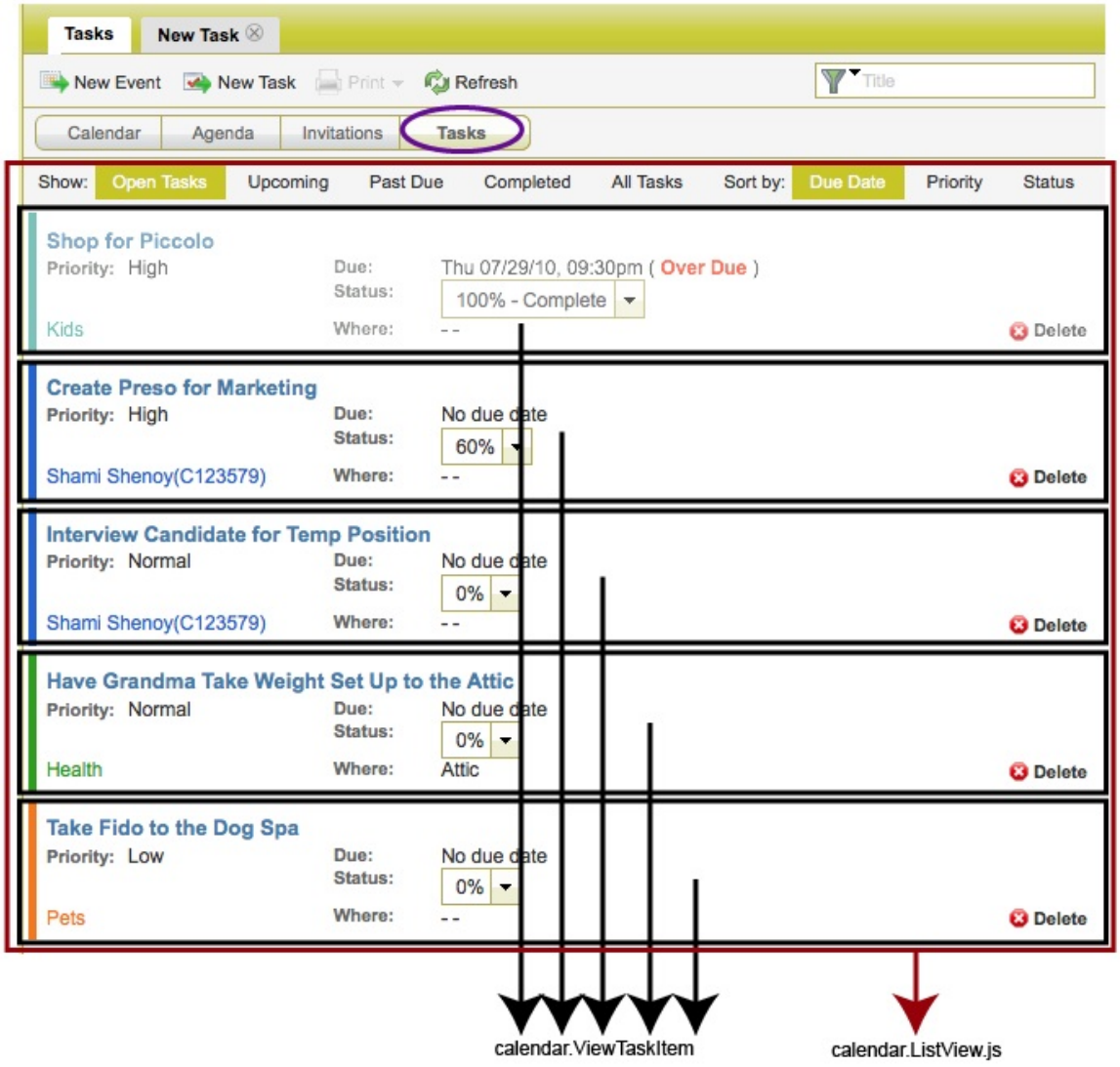


Figure 39: Calendar - Event (Daily Event and Weekly Event)

The `calendar.Event.js` widget is highlighted in the following illustration:

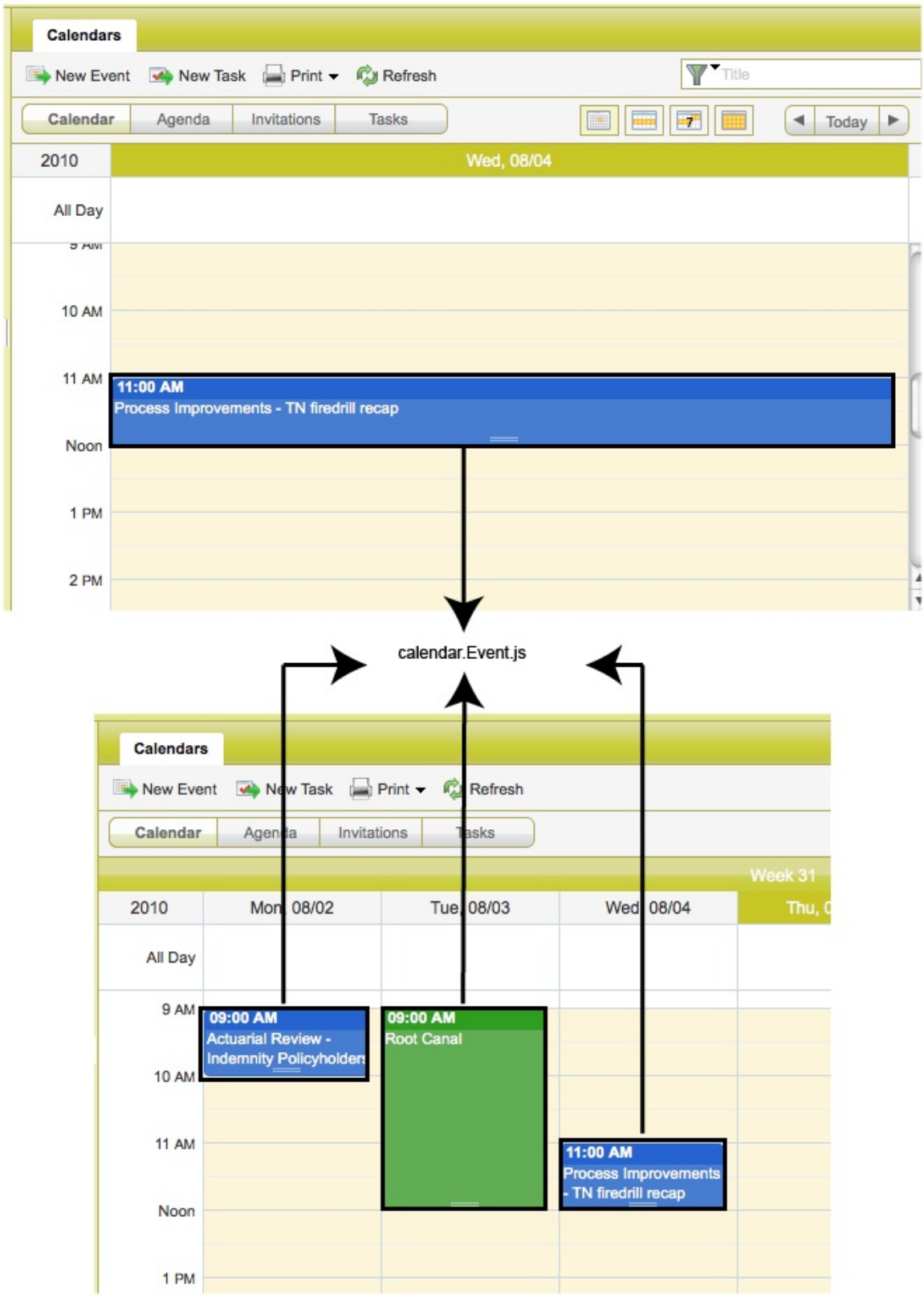
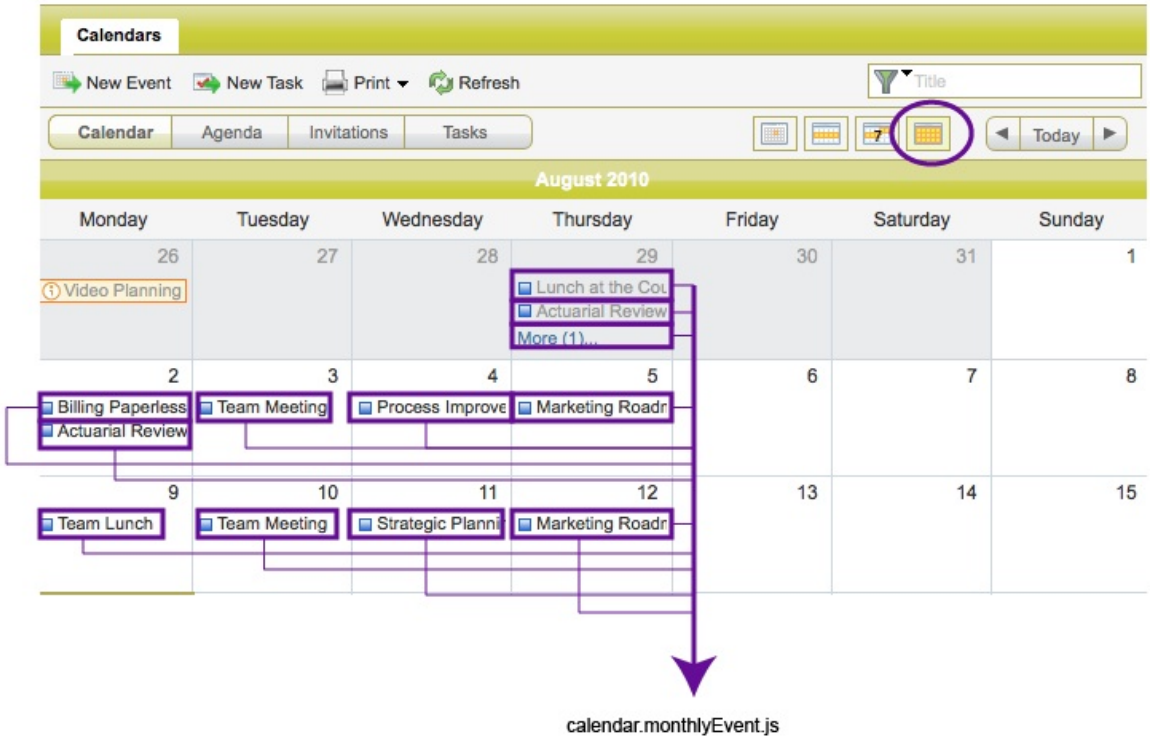


Figure 40: Calendar - Monthly Events

The `calendar.monthlyEvent.js` widget is highlighted in the following illustration:



Calendar Create Events, Create Tasks, Balloons, and Availability

The following calendar widgets are used to customize events, tasks, balloons, and availability:

Figure 41: Calendar - Create Event

The `calendar.CreateEvent.js` and `calendar.Invitees.js` widgets are highlighted in the following illustration:

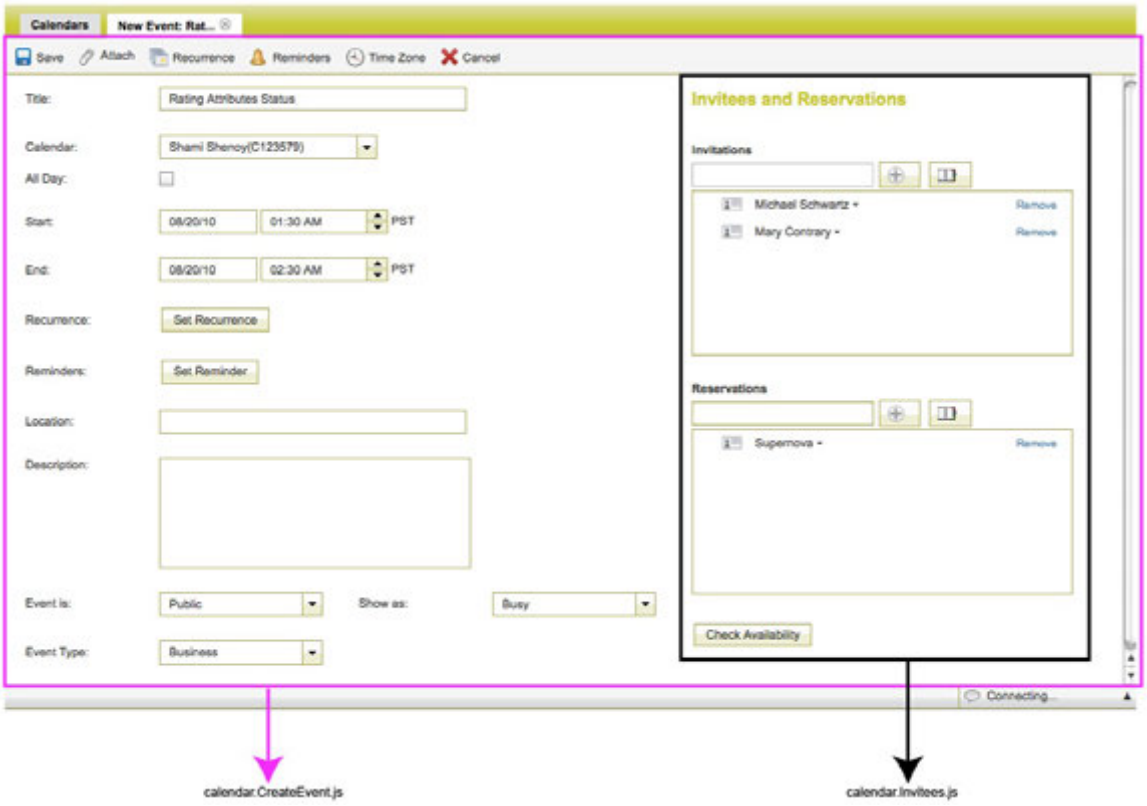


Figure 42: Calendar - Recurring Event Dialog

To get this dialog, you go to the Create Event Dialog and then press the Recurrence Button.

The `calendar.RecurrenceDialog.js` widget is highlighted in the following illustration:

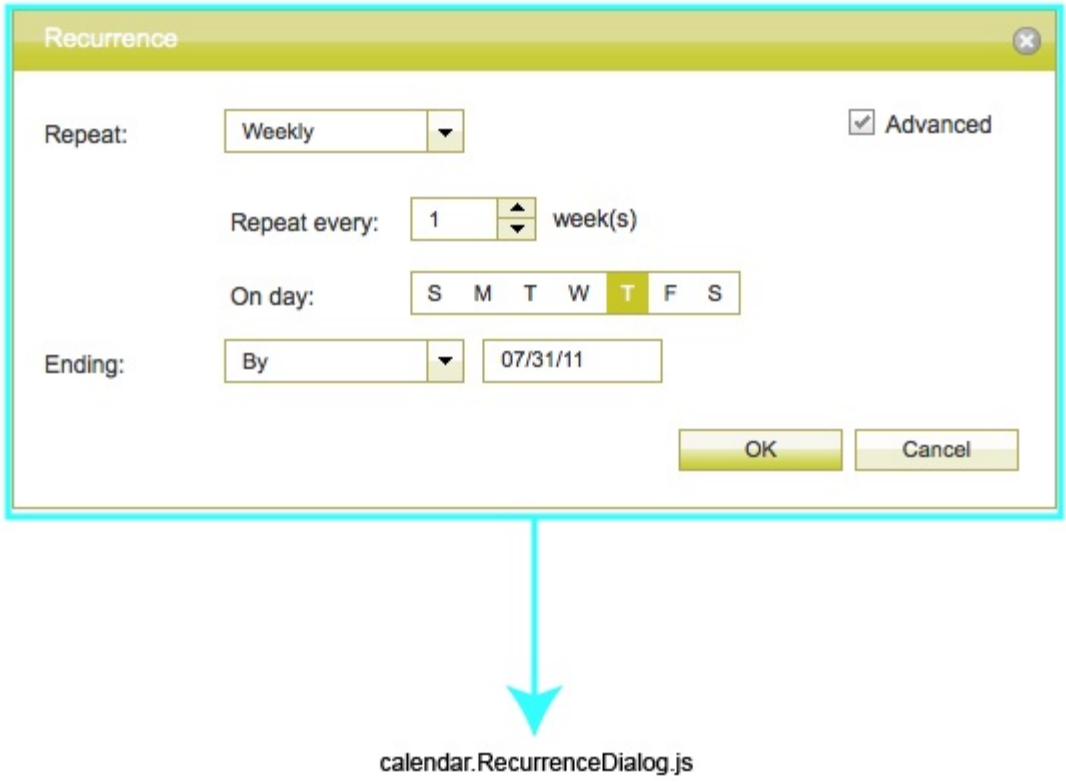


Figure 43: Calendar - Create Task

The `calendar.CreateTaskDialog.js` widget is highlighted in the following illustration:

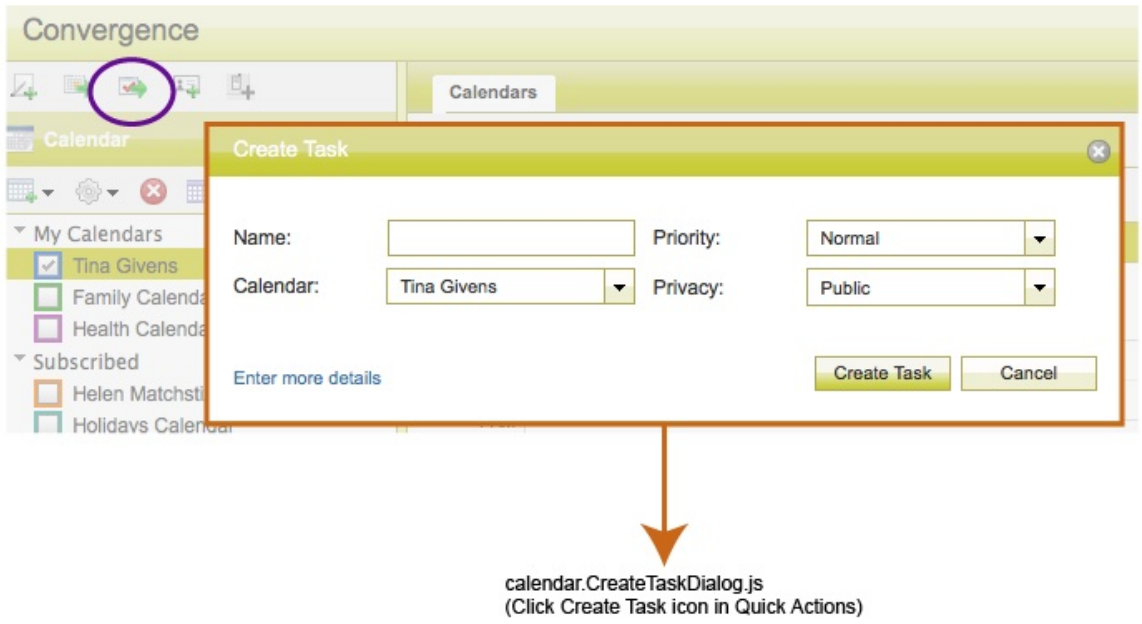


Figure 44: Calendar - Task Detail

The `calendar.TaskDetail.js` widget is highlighted in the following illustration:

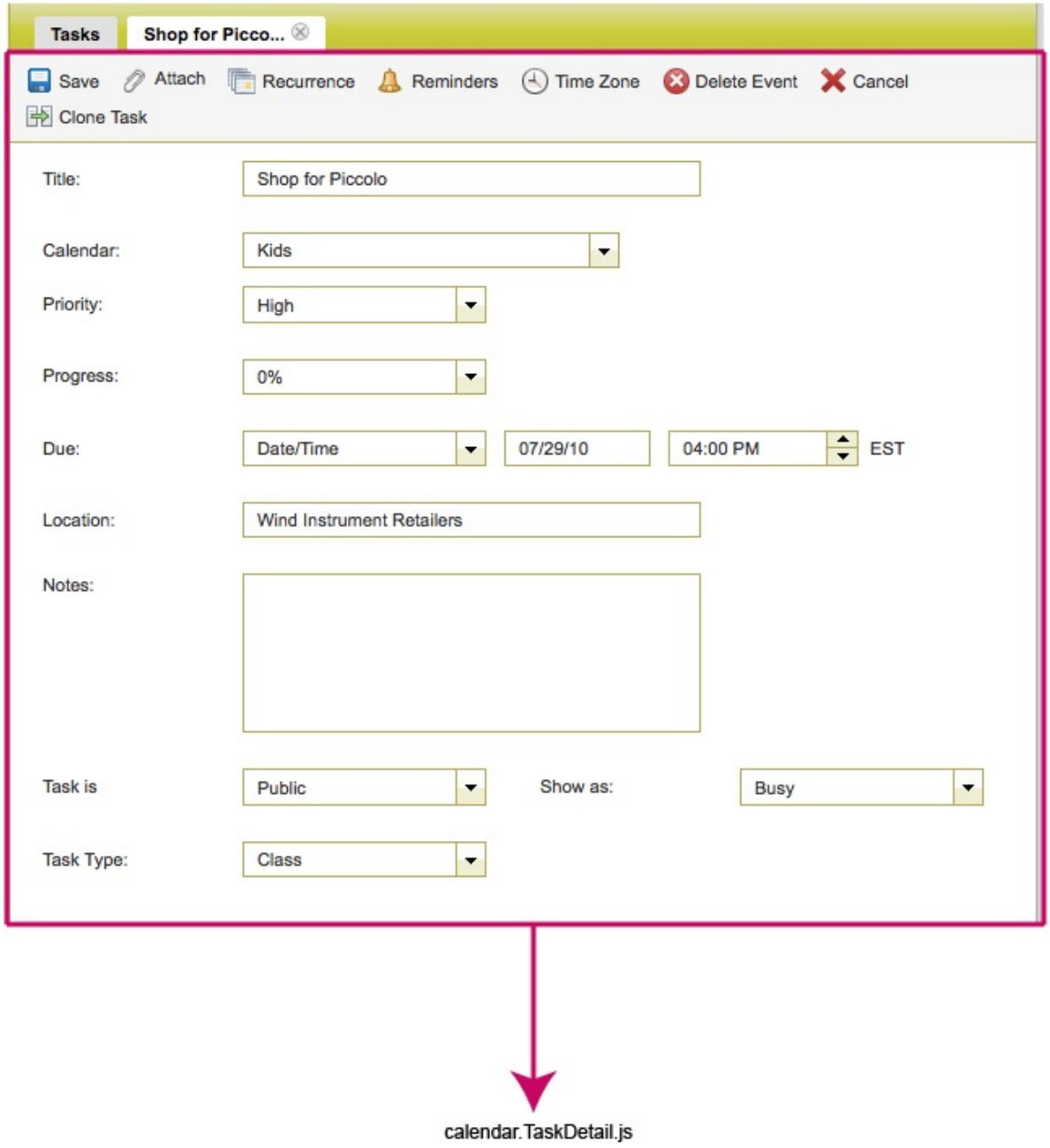


Figure 45: Calendar - Event Balloon

The `calendar.EventBalloon.js` widget is highlighted in the following illustration:

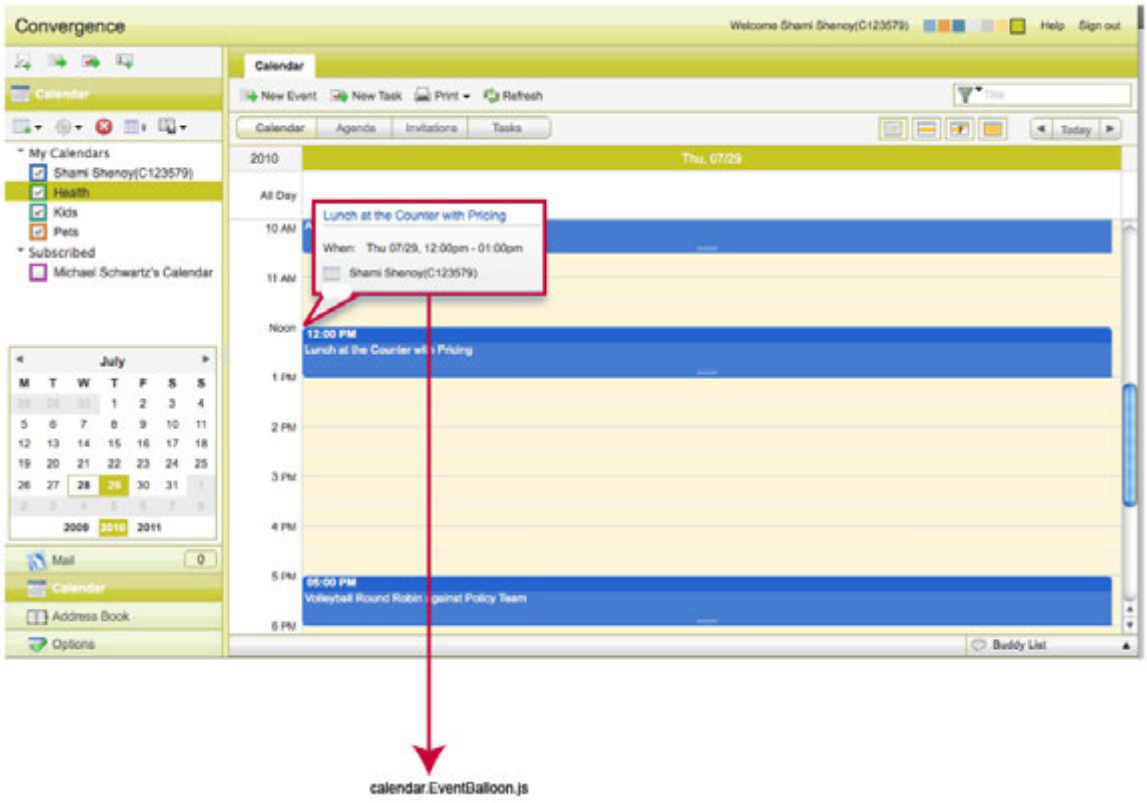


Figure 46: Calendar - View Event

The `calendar.ViewEvent.js` widget is highlighted in the following illustration:

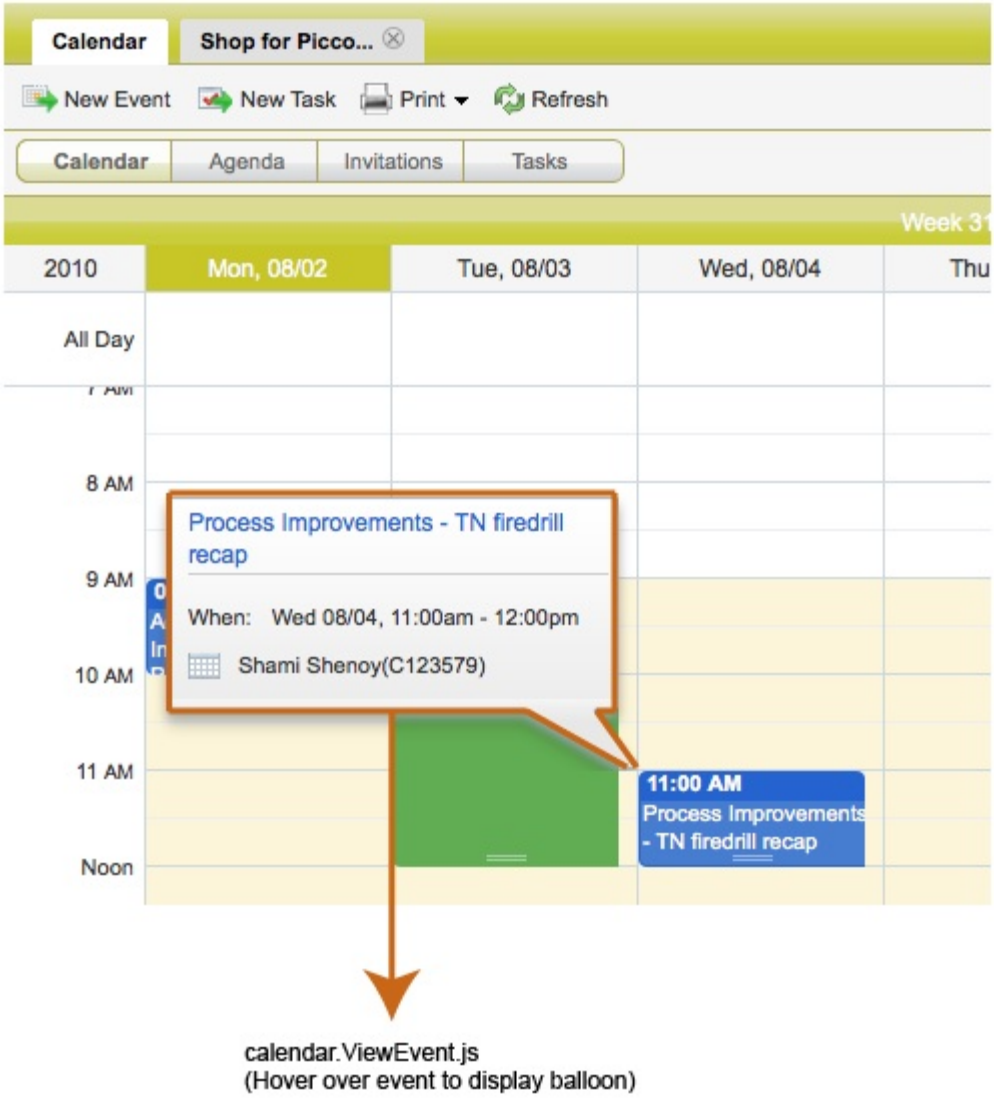


Figure 47: Calendar - Quick Event Balloon

The `calendar.QuickEventBalloon` widget in `calendar.Balloon.js` is highlighted in the following illustration:

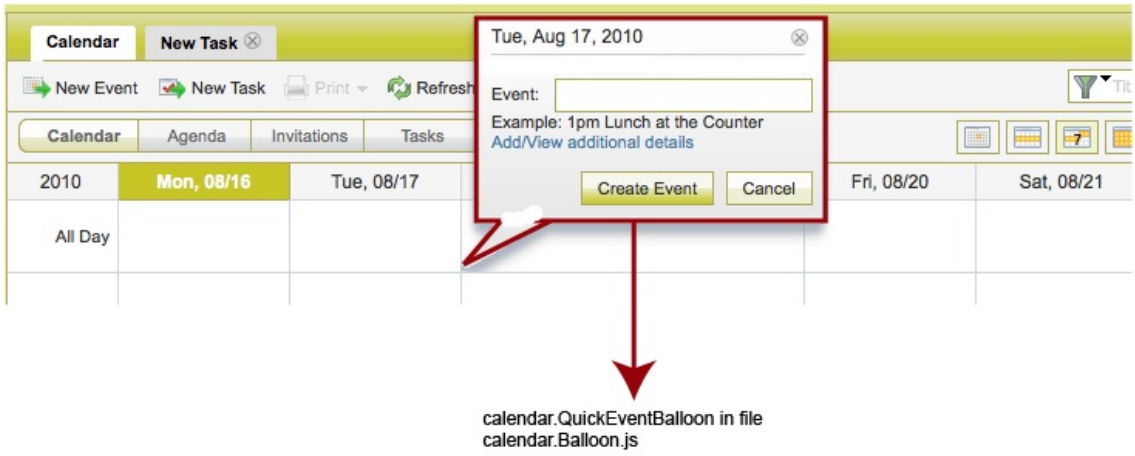
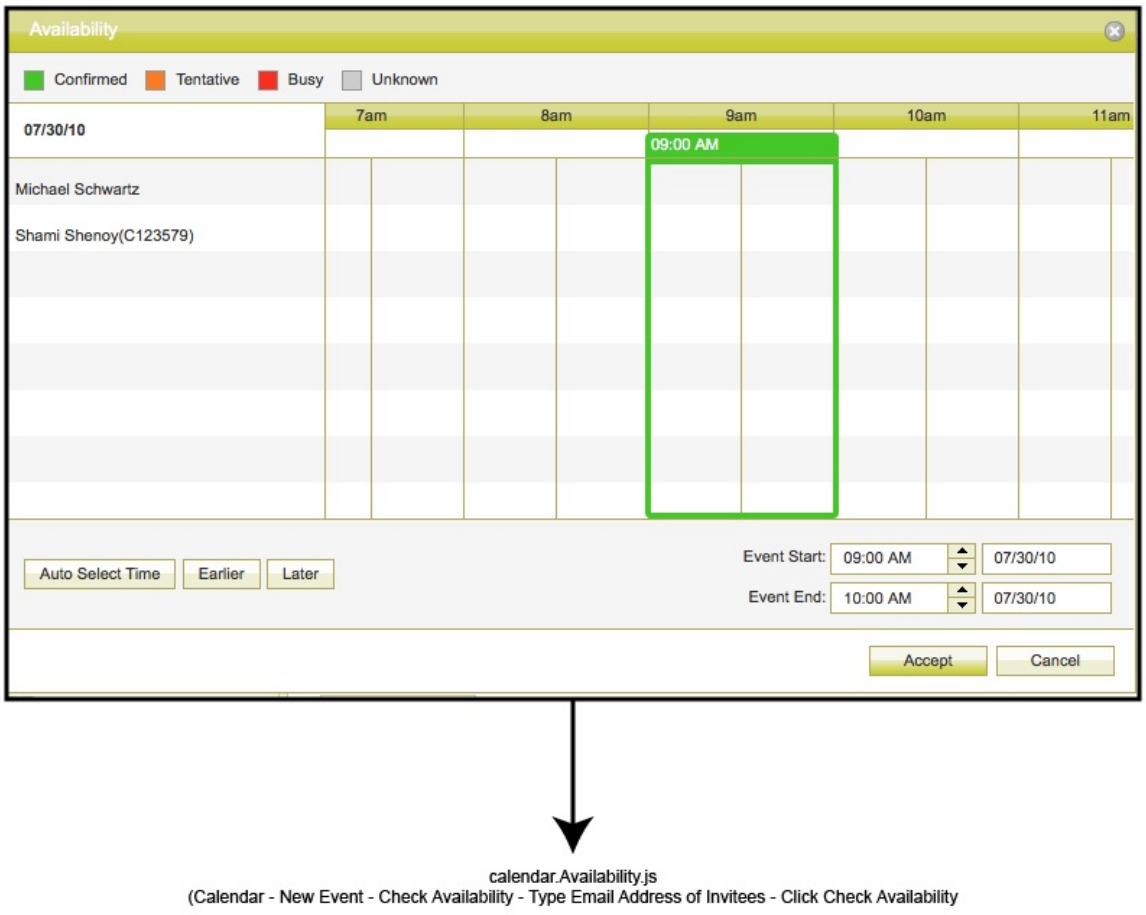


Figure 48: Calendar - Calendar Availability

The `calendar.Availability.js` widget is highlighted in the following illustration:



Calendar Dialogs

The following Calendar widgets display various dialog windows that can be customized:

Figure 49: Calendar - Reminder Dialog

To get this dialog, you go to the Create Event Dialog and then press the Reminders Button.

The `calendar.RecurrenceDialog.js` widget is highlighted in the following illustration:

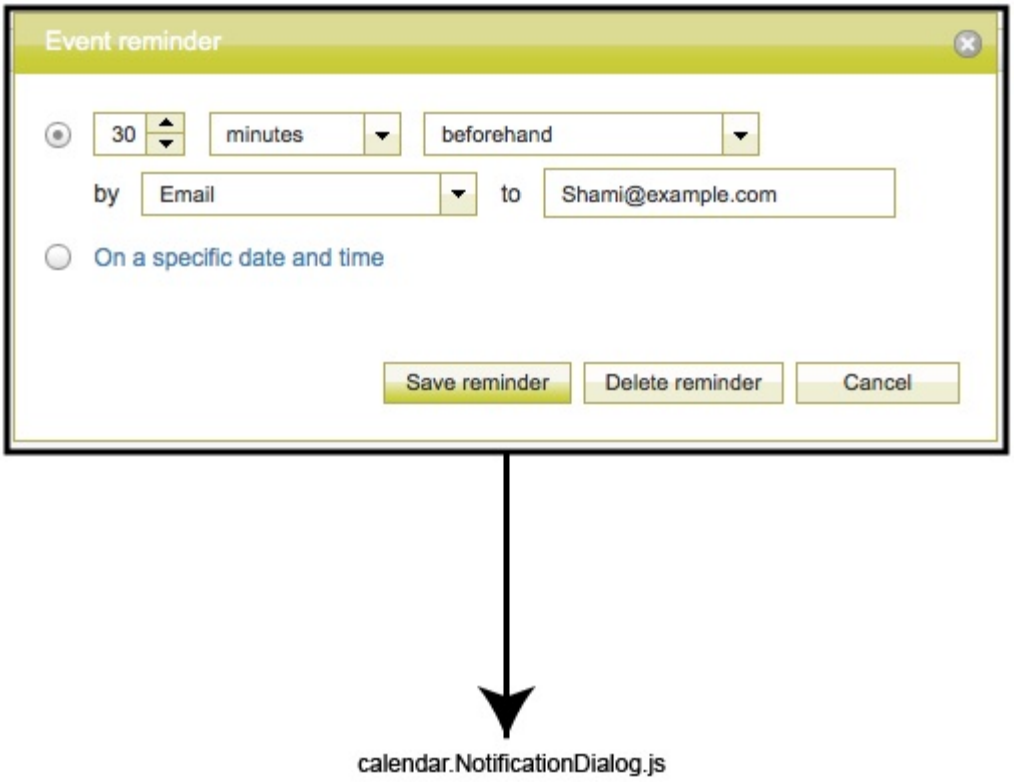


Figure 50: Calendar - Calendar Properties

The `calendar.CalendarPropertiesDialog.js` widget is highlighted in the following illustration:

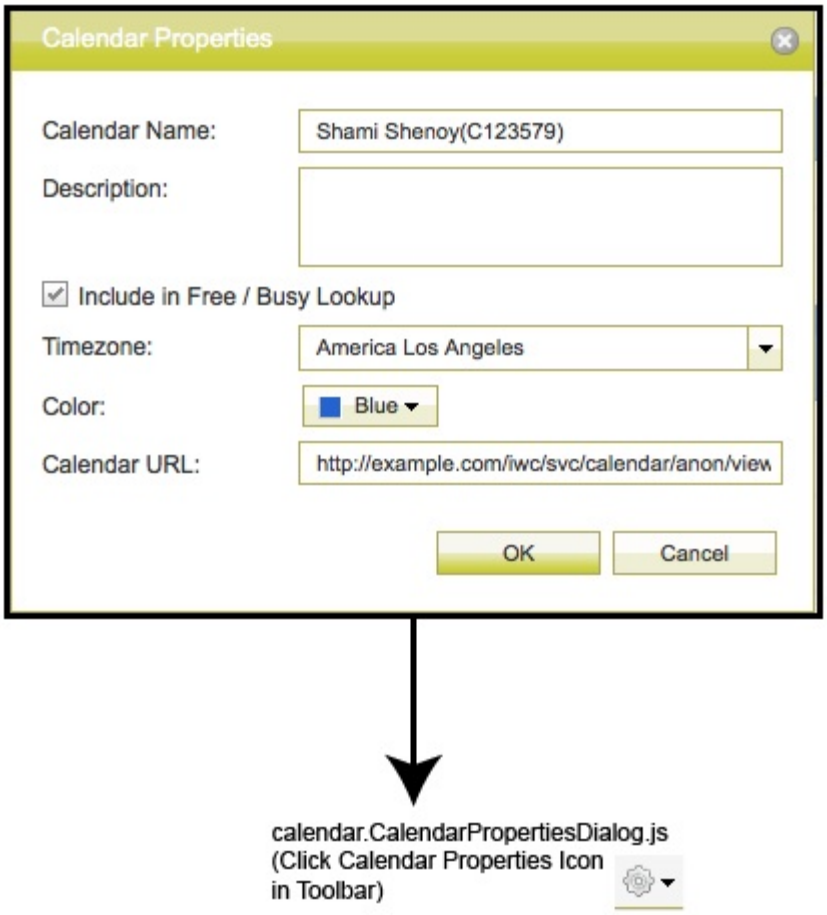


Figure 51: Calendar - Print Calendar View

The `calendar.PrintDialog.js` widget is highlighted in the following illustration:

Figure 52: Calendar - Print

The `calendar.Print.js` widget is highlighted in the following illustration:

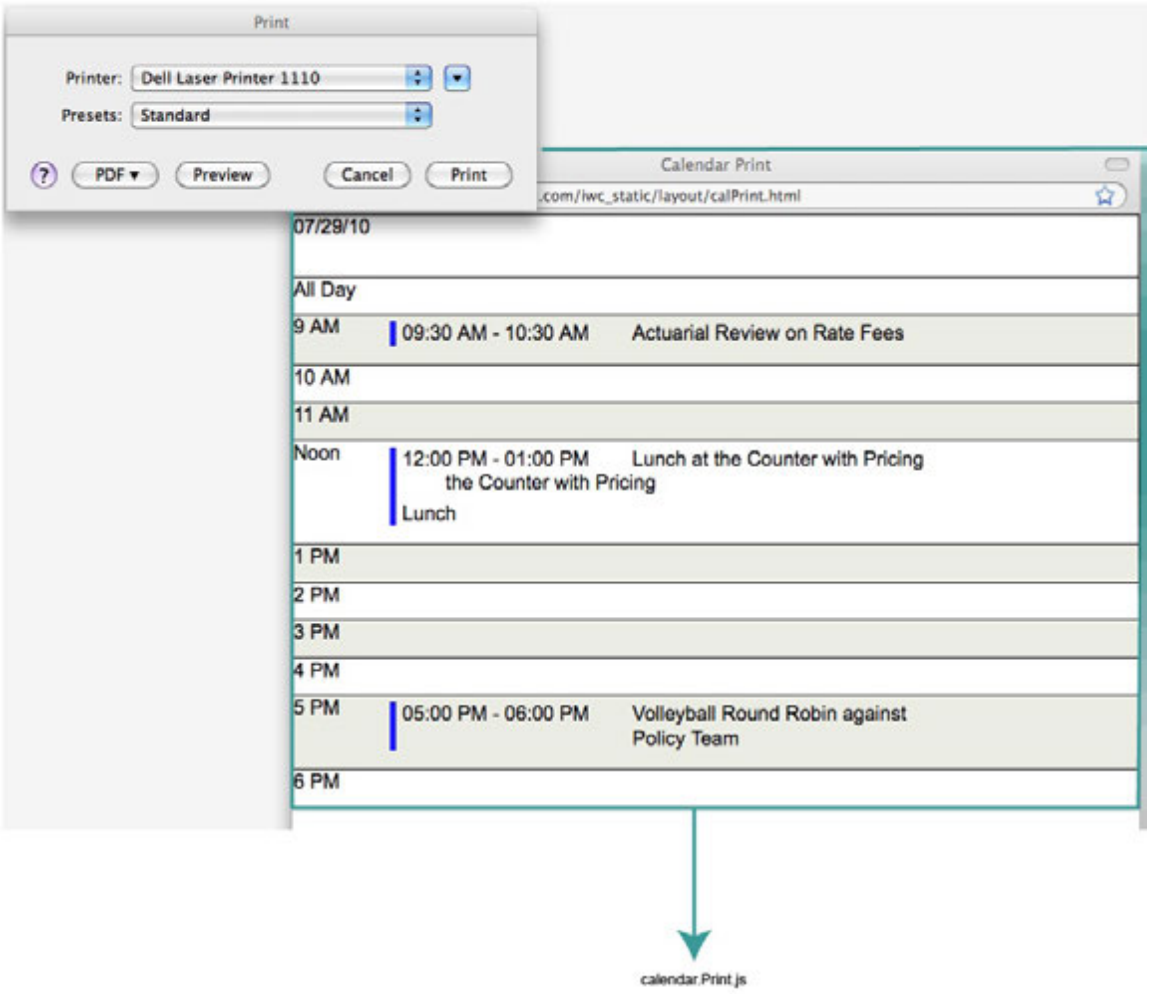


Figure 53: Calendar - Timezone Dialog

The `calendar.TimezoneDialog.js` widget is highlighted in the following illustration:

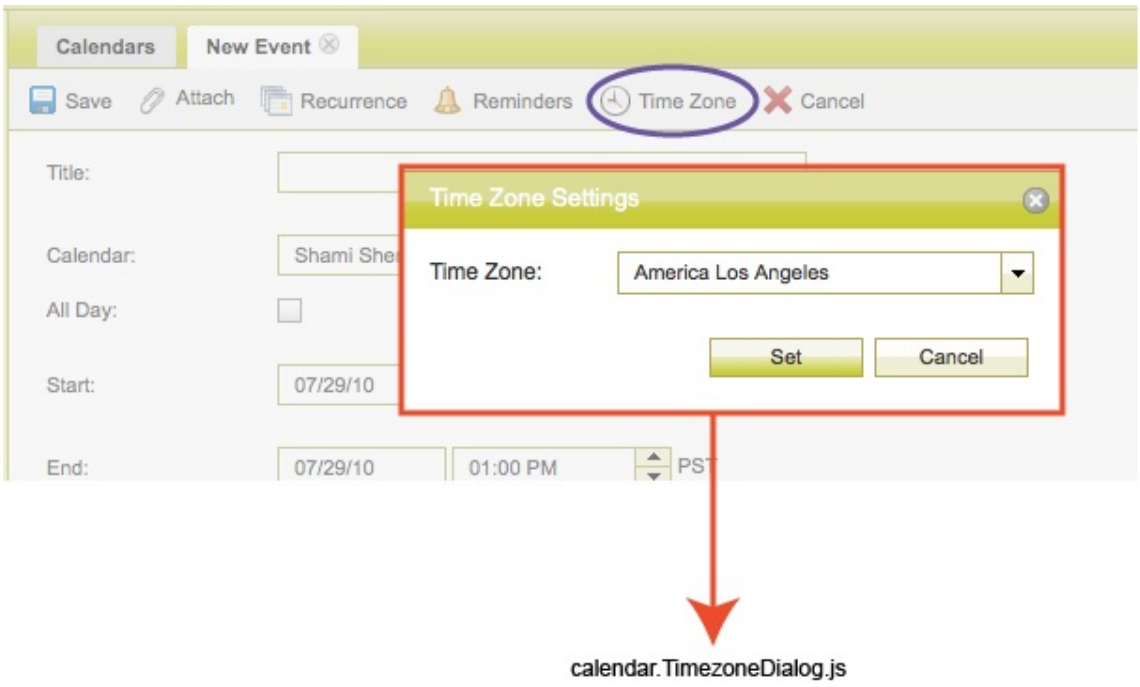


Figure 54: Calendar - Export Dialog

The `calendar.ExportDialog.js` widget is highlighted in the following illustration:

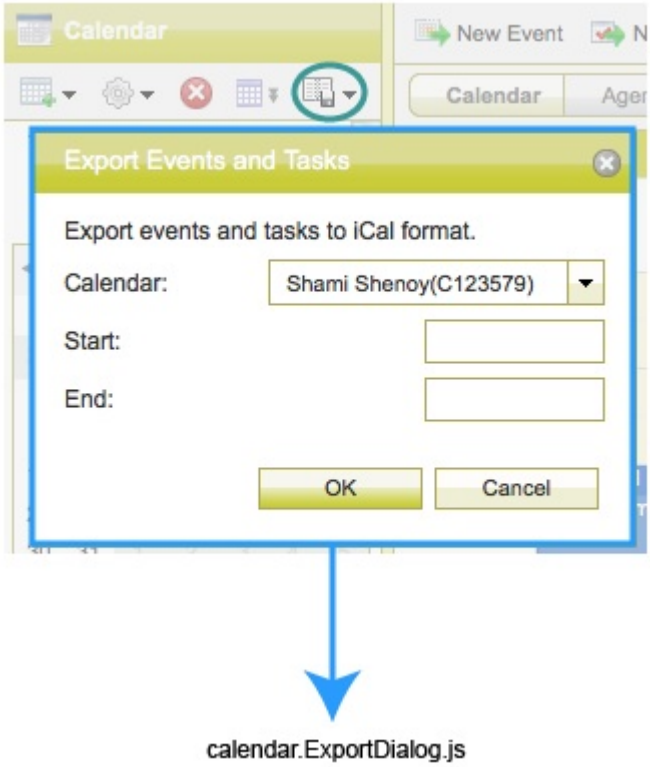


Figure 55: Calendar - Import Dialog

The `calendar.ImportDialog.js` widget is highlighted in the following illustration:

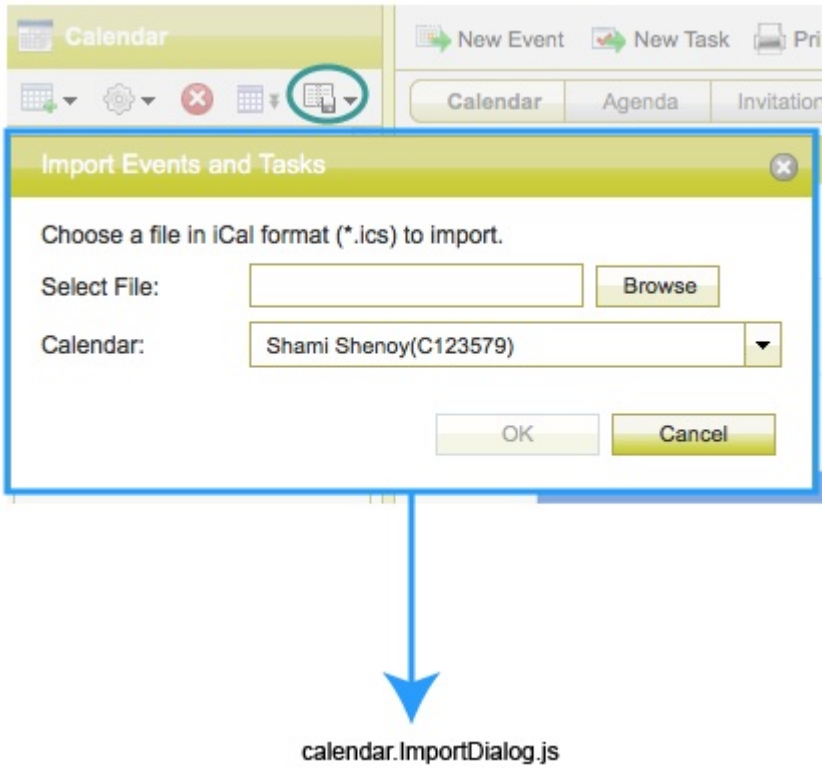
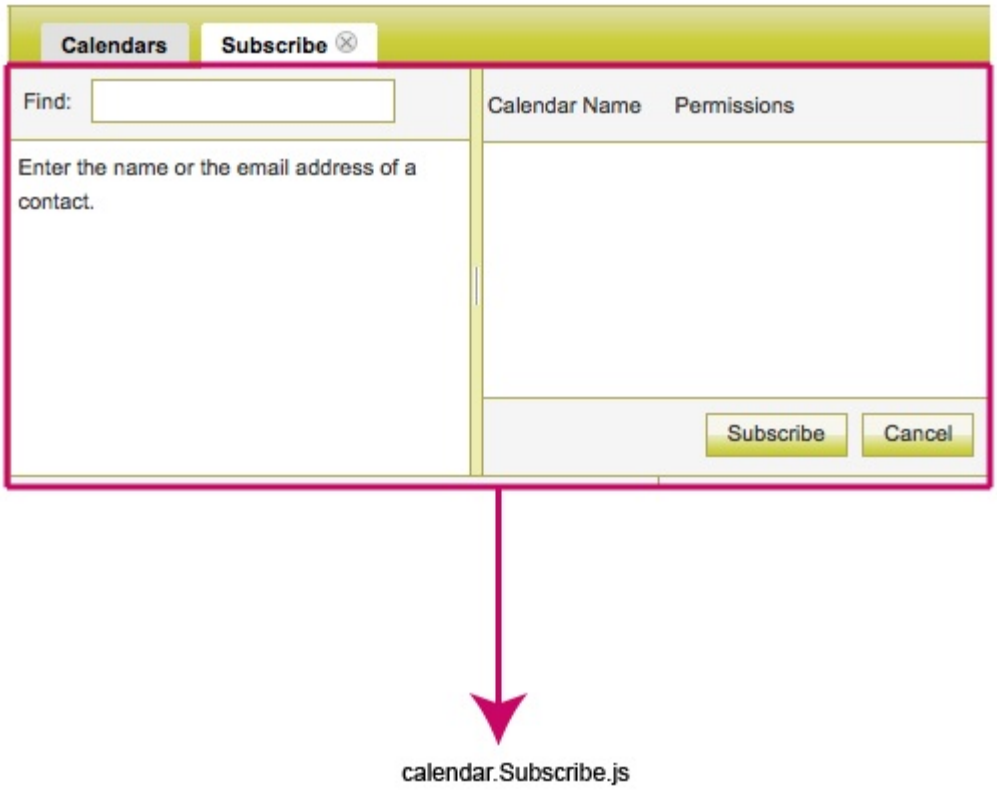


Figure 56: Calendar - Calendar Subscribe Dialog

The `calendar.Subscribe.js` widget is highlighted in the following illustration:



Instant Messaging Widgets

- im.AddBuddyDialog
- im.AvatarDialog
- im.Contacts
- im.ContactPropertiesDialog
- im.GroupPropertiesDialog
- im.CustomPresenceDialog
- im.PresenceButton
- im.Session

Figure 57: IM Contacts and Presence Button

The `im.Contacts.js` widget is highlighted in the following illustration:

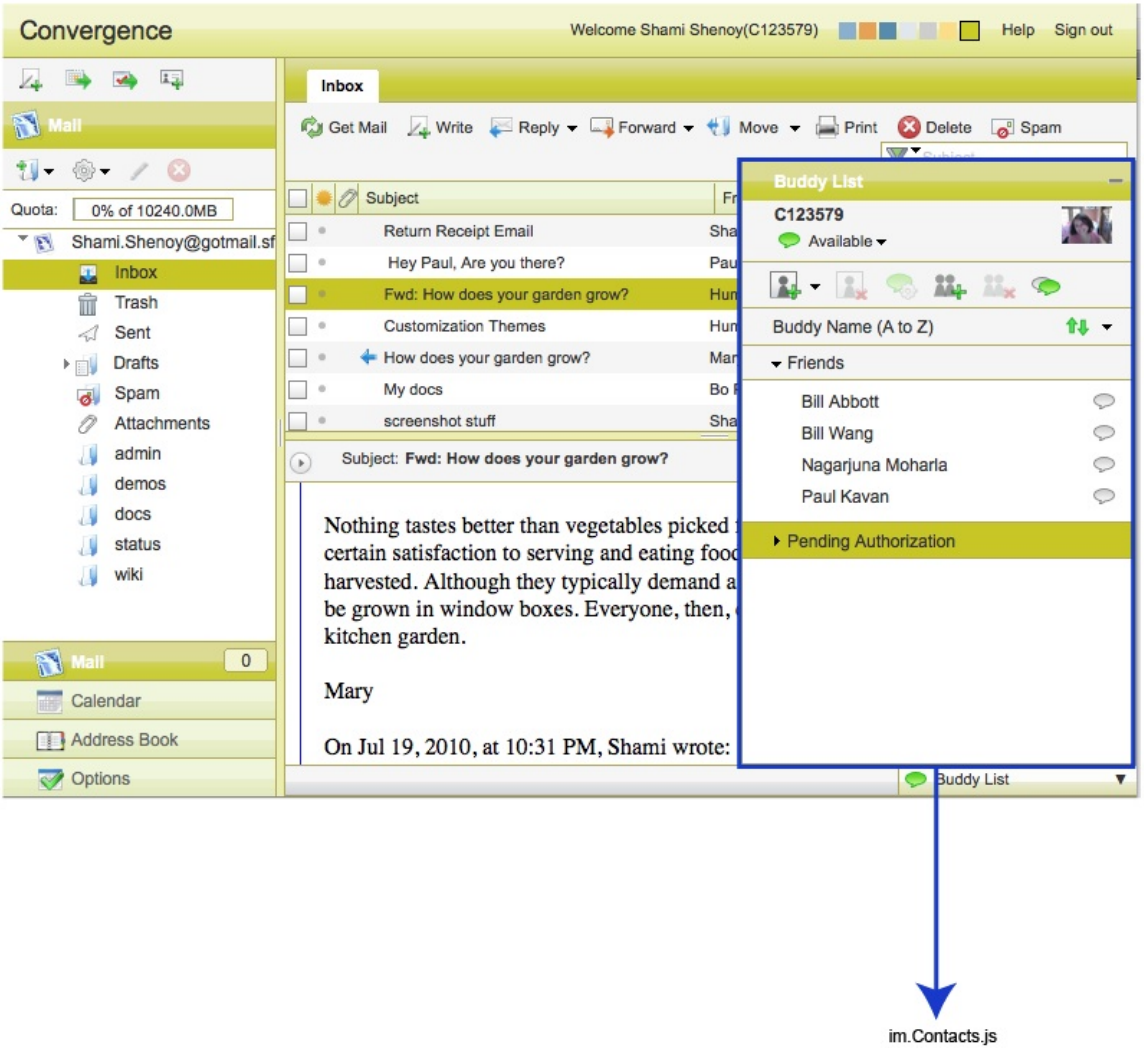


Figure 58: IM - IM Session

The `im.Session.js` widget is highlighted in the following illustration:

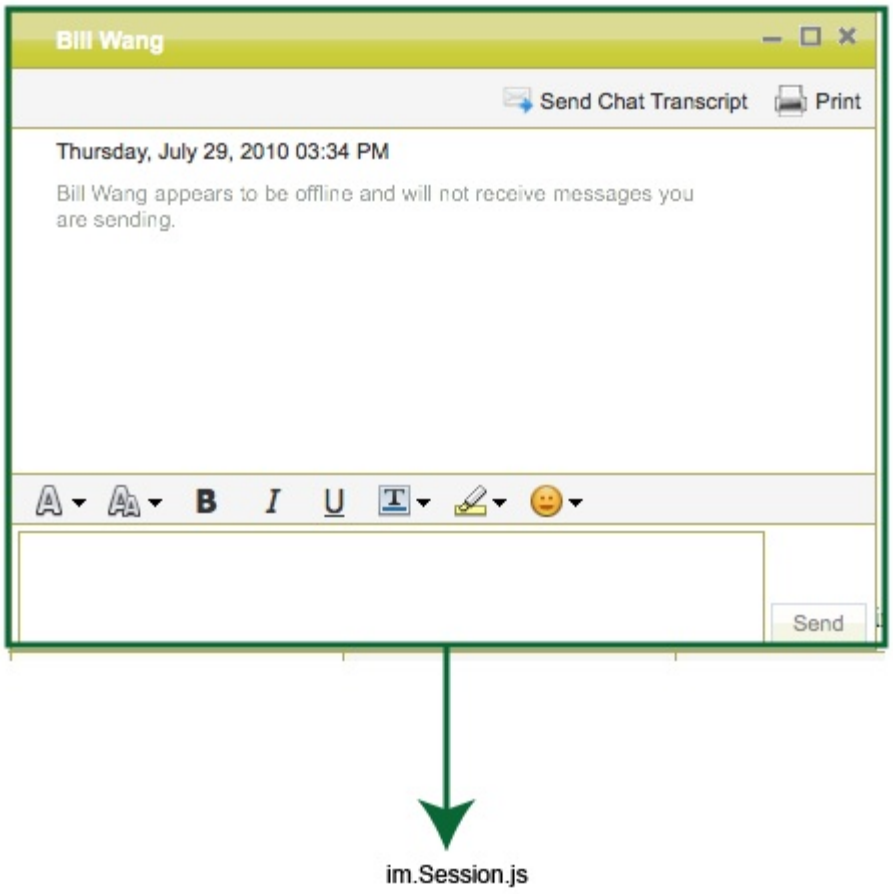


Figure 59: IM - Add Buddy Dialog

The `im.AddBuddyDialog.js` widget is highlighted in the following illustration:

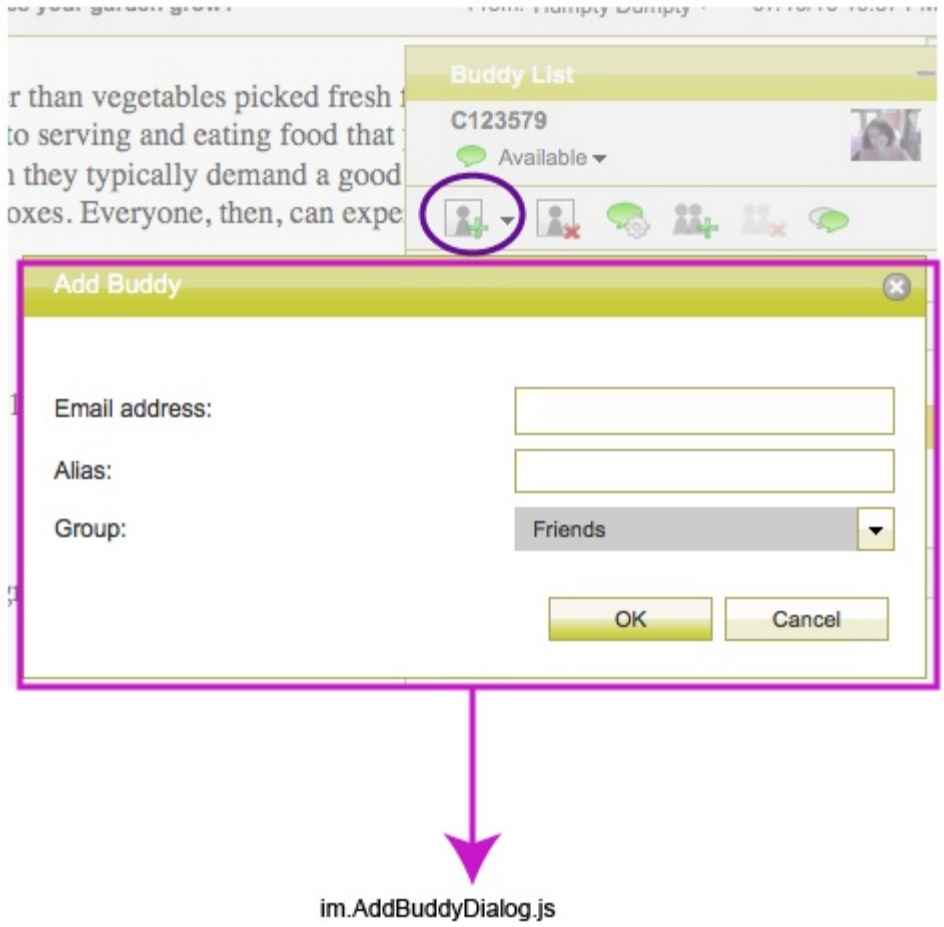


Figure 60: IM - Contact Properties Dialog

The `im.ContactPropertiesDialog.js` widget is highlighted in the following illustration:

Figure 61: IM - Create Group and Group Properties Dialog

The `im.GroupPropertiesDialog.js` widget is highlighted in the following illustration:

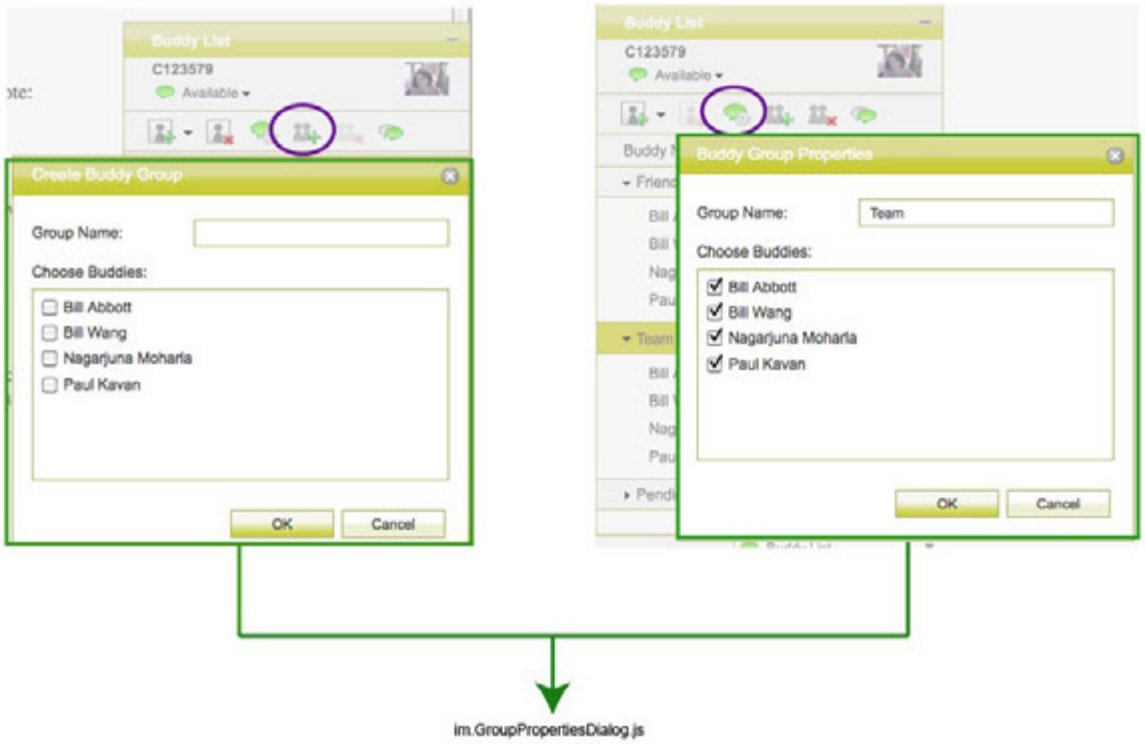


Figure 62: IM - Upload Avatar Dialog

The `widget.PhotoDialog.js` widget is highlighted in the following illustration:

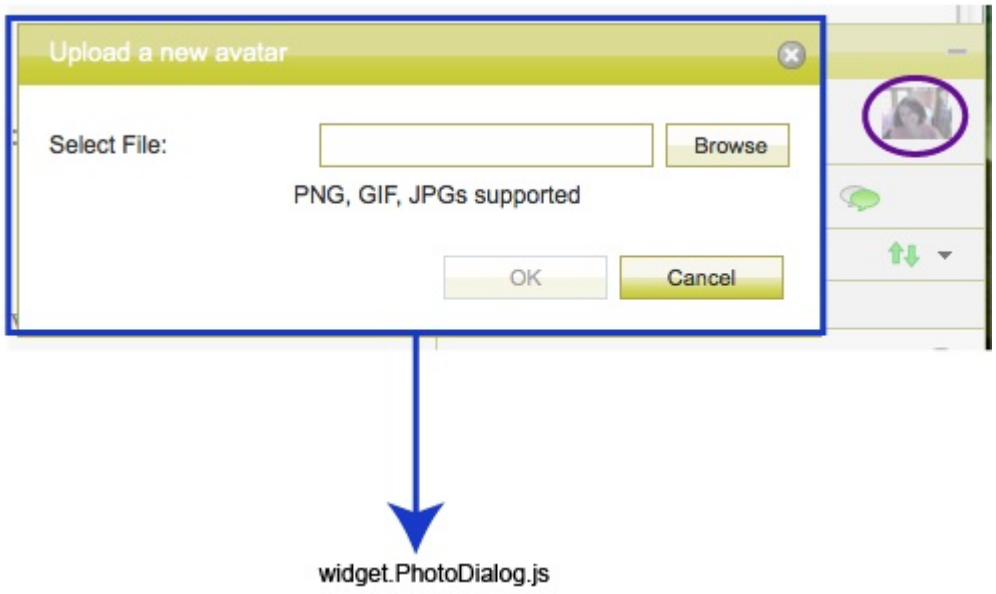
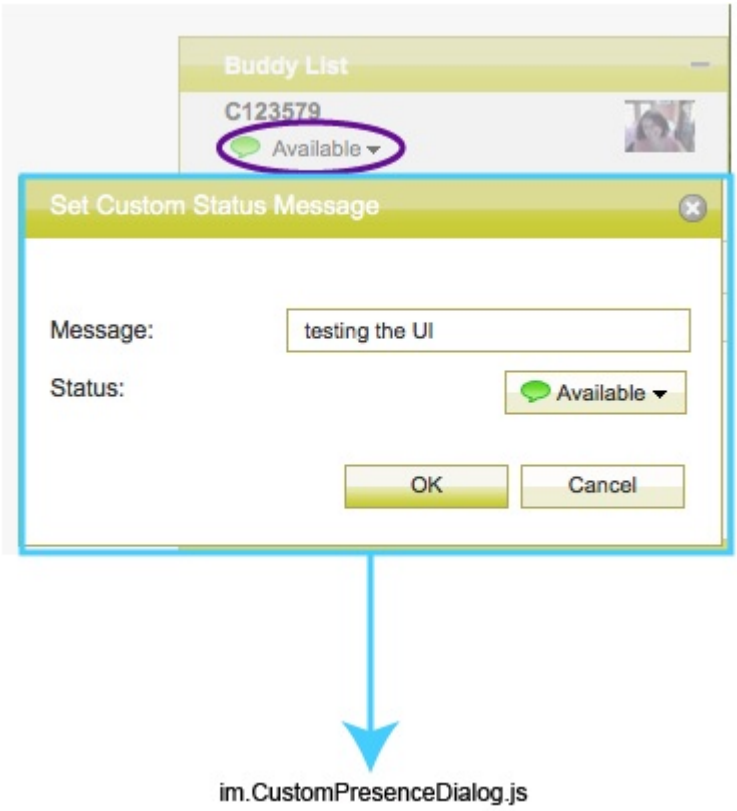


Figure 63: IM - Set Custom Presence Dialog

The `im.CustomPresenceDialog.js` widget is highlighted in the following illustration:



im.CustomPresenceDialog.js

Chapter 28. Customizing Layout HTML Pages

Customizing Layout HTML Pages

> **Note**
> This example applies to Convergence 2.x. For Convergence 3.0.0.0.0, see Customizing Layout HTML Pages in Convergence 3.0.0.0.0.
>
> For additional information on customizing your Convergence UI, review the Convergence Customization Guide.

Topics:

In Convergence, you can customize the `login.html`, `main.html`, and anonymous `calendar.html` HTML layout pages. But unlike other c11n customization, HTML layout pages are customized differently:

- Because the c11n modules are not invoked like with themes or widget customization, the steps for HTML customization do not require enabling JavaScript or i18n files in the `config.js`.
- Modifications to the login page are overwritten when a patch or update is applied in the future. Any changes should be therefore be recorded so they can be re-applied if required.

The following common examples describe how to customize `login.html`, `main.html`, and anonymous `calendar.html` HTML layout pages.

> **Note**
> Do not copy the code from `iwc_static/layout` because URL references are not updated.

To Create and Customize `login.html`

1. To create and customize the `login.html` page for allDomain, copy the sample from `c11n_sample/allDomain/layout` to `iwc_static/c11n/allDomain/layout` directory.
2. Run the `iwcadmin` tool to set the new layout file location.
   For example, to set allDomain to use the new `login.html`,

```
iwcadmin -o client.loginpage -v
"/iwc_static/c11n/allDomain/layout/login.html"
```

To Create and Customize `login.html` in a Hosted Domain

In the following example, a hosted domain's (example.com) `login.html` page is customized:

1. After copying the sample `login.html` page from `c11n_sample/allDomain/layout` to the `iwc_static/c11n/allDomain/layout` directory, change the following two lines that load the allDomain resources from:

   ```
   dojo.requireLocalization("c11n.allDomain", "resources");
   var l10n = dojo.i18n.getLocalization("c11n.allDomain",
   "resources");
   ```

   to the following lines which will load the example_com resources:

   ```
   dojo.requireLocalization("c11n.example_com", "resources");
   var l10n = dojo.i18n.getLocalization("c11n.example_com",
   "resources")
   ```

2. Enable the example_com `login.html` page:

   ```
   iwcadmin -o client.{example.com}.loginpage -v
   "/iwc_static/c11n/example_com/layout/login.html"
   ```

To Modify the Login Page Welcome Message

> **Note**
> To modify the login page welcome message in Convergence 1.x, see: Modifying Login Page Welcome Message in Convergence 1.x.

The following example steps you through the process of modifying **Convergence** welcome message to **Welcome to Convergence** on the Convergence login page. This task lets you replace the default welcome message with a message appropriate for your site.

To modify the login page welcome message, follow these steps:

1. Since the `login.html` does not load any `c11n` modules or resources, you will have to copy the `c11n` resources from the `c11n_sample` into your `c11n` directory. For example, copy the following code in the `/iwc/static/c11n_sample/allDomain/layout/login.html` to your `c11n/allDomain/layout/login.html`:

   > **Note**
   > The function `loadC11nResources()` loads the `c11n/allDomain/nls/resources.js` and uses the new `login_welcome_msg` string.

```
function loadC11nResources() {
   dojo.registerModulePath("c11n", "../../../c11n");
   dojo.requireLocalization("c11n.allDomain",  "resources");
   var l10n = dojo.i18n.getLocalization("c11n.allDomain",
"resources");
   dojo.mixin(iwc.l10n, l10n);
}
```

2. Copy `resources.js` from `c11n_sample/allDomain/nls/` to `c11n/allDomain/nls/`.
   - In the `<lang>/resources.js`, you can modify the localized `login_welcome_msg` string.
   - Additionally, you can provide language translation strings to the `login_welcome_msg` in this file.
   - Because the c11n modules are not invoked, there is no need to modify `c11n/config.js` to set `i18nEnabled`.
3. If you want to hide the `login_welcome_msg` string, add the following style to `c11n/allDomain/layout/login.html`:

```
<style type="text/css">
    #copyright, #welcomeMsg {
        display: none;
    }
</style>
```

4. Restart Application Server and clear the browser cache to see the change.

To Create and Customize `main.html`

1. To create and customize the `main.html` page, for allDomain, copy the sample from `c11n_sample/allDomain/layout` to `iwc_static/c11n/allDomain/layout` directory.
2. Run the `iwcadmin` tool to set the new layout file location.
   For example, to set allDomain to use the new `main.html`:

```
iwcadmin -o client.mainpage -v
"/iwc_static/c11n/allDomain/layout/main.html"
```

Note
In some cases, after customizing `main.html`, the print option may no longer work for Mail and Calendar.
Workaround: Copy `shell.html` and `calPrint.html` from `iwc_static/layout` to `iwc_static/c11n/allDomain/layout` and restart Application Server.

To Configure the Per-Domain Main Page

If you want to configure a per-domain main page for each of your domains, follow these steps:

1. If it has not been done already, add the `sunUCPreferences` object class to the domain, as in the following example:

```
ldapmodify -D "cn=directory manager" -w <password>
dn: o=domain.com,o=isp
changetype:modify
add:objectclass
objectclass:sunUCPreferences
```

2. Add the per-domain client-preference LDAP attribute for the main page in the domain's LDAP entry. To do this, set the `sunUCExtendedClientPrefs` attribute of `sunUCPreferences` as in the following example:

```
ldapmodify -D "cn=directory manager" -w <password>
dn: o=domain.com,o=isp
changetype:modify
add: sunUCExtendedClientPrefs
sunUCExtendedClientPrefs:
mainpage=/iwc_static/layout/my-mainpage.html
```

3. Restart GlassFish Server and clear the browser cache to see the change.
4. When a user logs into the Convergence UI, the user is redirected to the main page specified by the MainPage LDAP attribute.

If the MainPage attribute is not configured, then the default main page, which is configured by the `client.mainpage` Convergence configuration option, is loaded. For more information on the MainPage attribute, see: sunUCExtendedClientPrefs.

To Create and Customize `calendar.html`

1. To create and customize the anonymous `calendar.html` page, for allDomain, copy the sample from `c11n_sample/allDomain/layout` to `iwc_static/c11n/allDomain/layout` directory.
2. Run the `iwcadmin` tool to set the new layout file location.
   For example, to set allDomain to use the new `calendar.html`,

```
iwcadmin -o client.anoncalviewpage -v
"/iwc_static/c11n/allDomain/layout/calendar.html"
```

Chapter 29. Customizing Themes and Banner in Convergence 2.x

Customizing Themes and Banner in Convergence 2.x

Note
This information describes enhancements to theme and banner customization beginning with Convergence 2 with Patch 1.

For Convergence 3.0.0.0.0, see Customizing Themes and Banner in Convergence 3.0.0.0.0.
For Convergence 1.x, see Customizing the Convergence Banner and Customizing the Convergence Theme.

For additional information on customizing your Convergence UI, review the Convergence Customization Guide.
Related Theme and Banner Customization Examples:

- Adding a Logo to the Right Side of the Banner
- Making the Customized Logo a Clickable Link in the Banner
- Handling Large Logos in Gradient Themes

Topics:

Overview of Convergence Theme and Banner Customization

This section provides a conceptual overview of how theme customization works in Convergence. In addition to describing updates from the last release, the themes included in this release are explained, along with how the css templates work together.

Changes in Theme Customization from Previous Releases

Beginning with Convergence 2 Patch 1, there are a number of improvements to Convergence customization:

- reducing the repetitiveness of the CSS templates and allows for a single template to be used by multiple themes
- maintaining a consistent way to define CSS declarations that does not exist in previous versions of Convergence
- reducing the dependency on CSS selector changes by isolating those changes in the template file

However, you must specify all values for each theme. In addition, you must have expert knowledge in dojo. And, while easier than in Convergence 1.x, maintenance of these Convergence 2.x and Convergence 3 templates requires knowledge in UI customization. The release beginning with Convergence 2 Patch 1 aims to address these issues.

Beginning with Convergence 2 Patch 1, customization contains a number of enhancements:

- using APIs instead of writing directly into the `themes.attr`("store"), creating streamlined updates and maintenance.
- adding ability to modify themes
- maintaining customization between patches
- reducing dependence on requiring all values to be specified in the `theme.json` for each theme
- reducing repetitive CSS values. By adding parent information when defining the theme in `customize.js` file, you no longer need to specify each value in each theme.
    - For example, `parentTheme: "theme_blue"` specifies all values in the blue theme.
    - When additional Convergence updates are introduced, you do not need to specify any additional values to this theme.

Themes Included

Themes included are a theme template called CommSuite:`theme_basic` and the themes shown in the following figure: `theme_blue`, `theme_orange`, theme_dark_blue, theme_light_blue, `theme_grey`, `theme_yellow`, and `theme_green`:

The process for the theme customization is the following:

1. CSS declarations are housed in a template file (`/themes/basic/template.css`):

/themes/basic/template.css

```
.Banner {
background: ${mastheadBackground};
color: ${mastheadColor};
}
```

2. CSS property values are in the `theme.json` file. For example in the blue theme, `themes/blue/theme.json`:

/themes/blue/theme.json

```
mastheadBackground: "#89AFDO",
mastheadColor: "#FFFFFF",
```

3. JavaScript substitutes template variables with JSON values resulting in the following:

```
.Banner {
  background: "#89AFDO";
  color: "#FFFFFF";
}
```

Basic Theme (`theme_basic`)

The `theme_basic` is the `parentTheme` of all included themes in Convergence. It is made up of the `theme.json` file and the `template.css` template file. The `theme.json` provides default values and template location. The `template.css` is the template file containing the css declarations with over 90 variables to be substituted from the values provided by `theme.json`. Specifically, it contains a masthead and logo, buttons, and tabs.

See: Basic Theme Reference in Convergence 2.x for for all default values.

Basic Theme (`theme_basic`) with Blue Theme (`theme_blue`)

The following example shows how the `theme_blue` extends the `theme_basic` template:

```
name: 'theme_blue',
parentTheme: "theme_basic",
configPath: "themes/blue/theme.json",
thumbnailColor: #89AFD0",
```

The `theme_basic` template serves as a parent theme to `theme_blue`. Parent theme values are pre-filled with default values so that you don't need to fill every value for `theme_blue`. For example, if the parent theme values for the banner (also referred to as masthead) are:

```
//Masthead
mastheadBackground: "#CCCCCC"
mastheadColor: "#333333",
mastheadHeight: "40px",
```

and the configuration path (`configPath`) which provides the location of the JSON files contains the following blue css property values:

```
//Masthead
mastheadBackground: "#89AFD0"
mastheadColor: "#FFFFFF",
```

the configuration path (`configPath`) values overwrite the parent theme values when specified with the result being the following:

```
mastheadBackground: "#89AFD0"
mastheadColor: "#FFFFFF",
mastheadHeight: "40px",
```

Blue Theme (`theme_blue`) CSS Flow

The process by which theme customization works with `theme_blue` is the following:

1. Load blue_theme css property `configPath` values, which are the values already in `theme_blue` .
2. Load blue_theme modified css values, which are values you modify to suit your customization.
3. Combine modified css values with `configPath` css values.
4. Load parent theme css. In this example, it is `theme_basic`.
5. Add parent values to supply any missing values not filled by modified or `configPath` css values.
6. Load template theme from JSON values.
7. Substitute JSON values into template.
8. Insert new theme into window head stylesheet.
9. Refresh layout.

Before You Begin

Before you begin to customize themes in Convergence, you need to complete the following procedures:

- Create a Convergence customization directory
- Enable customization in Convergence

To Create the Convergence Customization Directory

The customization namespace is `c11n`. The custom files in the `c11n` directory extend the code base, overwriting the standard elements with the customized elements. Note the following information about `c11n`:

- The `c11n` namespace does not exist by default. You must create it.
- If the `/iwc_static/c11n/config.js` file exists, customizations are booted based on how they are defined in this file.
- Convergence is deployed on the GlassFish Server web container. The files to work on are located under GlassFish Server's installation directory, typically:

```
/opt/SUNWappserver/domains/domain1/docroot/iwc_static
```

The easiest way to create the customization directory is to copy the `c11n_sample` directory provided with the Convergence installation. When you copy the `c11n_sample` directory rather than renaming it, the directory is preserved in case you need it again.

- Change to the `iwc_static` directory and copy the `c11n_sample` directory to the `c11n` directory.

For example:

```
cd /opt/SUNWappserver/domains/domain1/docroot/iwc_static
cp -r c11n_sample c11n
# Run the ls command to confirm that the directory was created.
```

The `c11n_sample` directory includes the `allDomain` directory and the domain-specific `example_com` directory. The file structure for the `allDomain` and `example_com` directories on `c11n` looks like the following:

```
c11n_sample/
    config.js                              (configuration file)
    allDomain/
         js/                               (contains widgets and custom
applications)
              service/                     (contains additional services)
              widget/                      (contains UI widgets)
              customize.js                 (general customization
JavaScript file for adding new services, widgets, and so on)
         layout/
         nls/                              (contains language-specific
files)
         themes/                           (contains themes)
              customize.js                 (customization JavaScript file
specific to adding and removing themes.  This file is different from
js/customize.js.)

    example_com/
         js/                               (contains example_com widgets
and custom applications)
              widget/                      (contains example_com UI
widgets)
              customize.js                 (example_com customization
JavaScript file for adding new services, widgets, and so on)
         nls/                              (example_com language-specific
files)
         themes/                           (contains example_com themes)
              customize.js                 (customization JavaScript file
specific to adding and removing themes.  This file is different from
js/customize.js.)
```

Note
The variable *domain* is one of the following:

- `allDomain`: The customizations (including themes) in this directory apply to all domains in your deployment.
- A specific domain name: The customizations in this directory apply to a specific individual domain.

To Enable Customization in Convergence

1. In the Convergence installation directory `<iwc-install-dir>/sbin`, set the `client.enablecustomization` parameter to true:

```
./iwcadmin -o client.enablecustomization -v true
```

2. Restart GlassFish Server and clear the browser cache to see the change.

Common Customization Scenarios

This section contains the following procedures:

To Add a Logo to All Themes

The most common theme customization scenario is adding a logo to all existing themes. In the following example, you can add your own `logo.png` to the set of themes found in the `c11n_sample/` directory.

To add a logo to all themes in your customization:

1. Enable Customization Server. See To Enable Customization in Convergence for these instructions.
2. Make sure you are in the `iwc_static/c11n` directory.
3. If you don't already have a `config.js` file in your `/c11n` directory, copy and configure the `config.js` file from `/iwc_static/c11n_sample/config.js` to create `/c11n/config.js`.
    a. Edit the `c11n/config.js` file.
    b. Make sure that `themeEnabled: true, i18nEnabled: true`, and `jsEnabled: true` are configured for the `allDomain` configuration.

       After you edit the `c11n/config.js` file, it should look like this:

/c11n/config.js

```
dojo.provide("c11n.config");

c11n.config = {

 // allDomain configuration
 allDomain: {
  module: "allDomain",  // module name
  themeEnabled: true,  // true if theme is customized
                i18nEnabled: true,  // true if i18n is
customized
                jsEnabled: true          // true if js is
customized

  // the last entry must not end with comma
 },

 // replace example.com for each domain configuration, change
 // domain name example.com to example_com for javascript
syntax and url syntax
 example_com: {
  module: "example_com",            // module name
  themeEnabled: false,              // true if theme is
customized
  i18nEnabled: false,               // true if i18n is
customized
  jsEnabled: false                  // true if js is
customized

  // the last entry must not end with comma

 }
}
```

4. Modify the `/iwc_static/c11n/allDomain/themes/customize.js` file where the user must enable modifications for a specific theme using `iwc.api.modifyThemeValues`. In this scenario, it is the parent theme for the provided themes. For example:

/iwc_static/c11n/allDomain/themes/basic/customize.js

```
iwc.api.modifyThemeValues("theme_basic",
"../c11n/allDomain/themes/basic/theme.json");
```

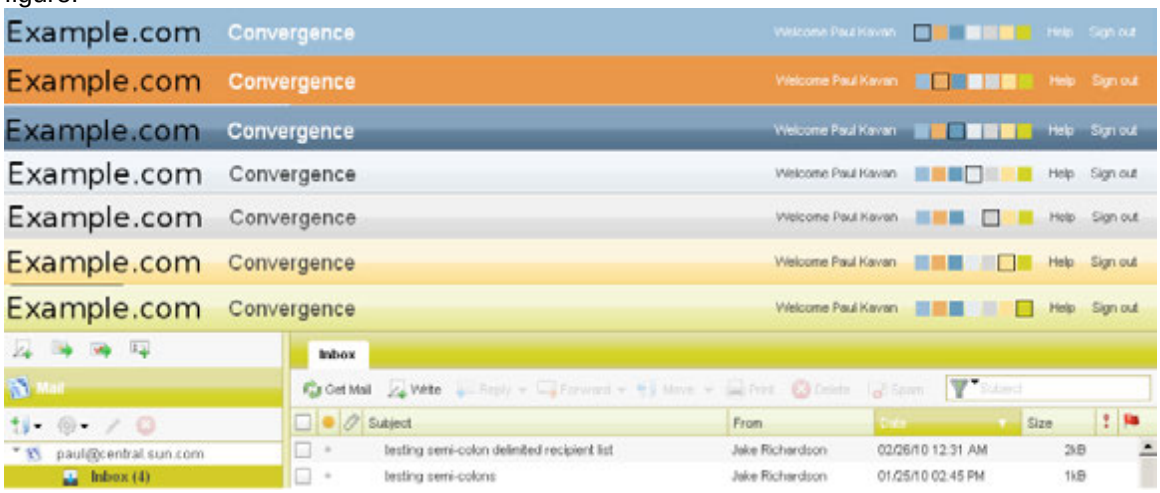
5. Edit the `/iwc_static/c11n/allDomain/themes/basic/theme.json` with the location of the logo, the logo's width, and height:

/iwc_static/c11n/allDomain/themes/basic/theme.json

```
{
// NOTE use should be limited to site wide standards.
// Example Setting the logo
logoBackground: 'transparent
url("../c11n/allDomain/themes/basic/images/logo.png") no-repeat
center center',
logoWidth: "180px",
mastheadHeight: "40px"
}
```

6. Replace the `logo.png` file in the `/themes/basic/images/` directory with the following `example.com` logo:

7. When the `example.com` logo is added in `allDomain`, the logo is added to all included themes: blue, orange, dark blue, light blue, grey, yellow, and green themes as shown in the following figure:



To Modify a Specific Theme

To modify a specific theme in your customization, you change a specific theme file as opposed to modifying `theme_basic`, which changes all themes:

In this example, a different logo is added to `theme_blue`.

1. Enable Customization Server. See To Enable Customization in Convergence for these instructions.
2. In this example, customizations are added under the specific domain `example_com`:

```
/c11n/config.js
/c11n/example_com/themes/customize.js
/c11n/example_com/themes/blue/theme.json
```

To view a sample file directory structure, see To Create the Convergence Customization Directory .

3. Make sure you are in the `iwc_static/c11n` directory.
4. If you don't already have a `config.js` file in your `/c11n` directory, copy and configure the `config.js` file from `/iwc_static/c11n_sample/config.js` to create `/c11n/config.js`.
   a. Edit the `c11n/config.js` file.
   b. Make sure that `themeEnabled: true, i18nEnabled: true`, and `jsEnabled: true` are configured for the `allDomain` configuration.

   After you edit the `c11n/config.js` file, it should look like this:

/c11n/config.js

```
dojo.provide("c11n.config");

c11n.config = {

 // allDomain configuration
 allDomain: {
  module: "allDomain",  // module name
  themeEnabled: true,  // true if theme is customized
                i18nEnabled: true,  // true if i18n is
customized
                jsEnabled: true          // true if js is
customized

  // the last entry must not end with comma
 },

 // replace example.com for each domain configuration, change
 // domain name example.com to example_com for javascript
syntax and url syntax
 example_com: {
  module: "example_com",            // module name
  themeEnabled: false,              // true if theme is
customized
  i18nEnabled: false,               // true if i18n is
customized
  jsEnabled: false                  // true if js is
customized

  // the last entry must not end with comma

 }
}
```

5. Modify the `/iwc_static/c11n/allDomain/themes/customize.js` file. For example:

/iwc_static/c11n/allDomain/themes/customize.js

```
iwc.api.modifyThemeValues("theme_blue",
"../c11n/allDomain/themes/blue/theme.json");
```

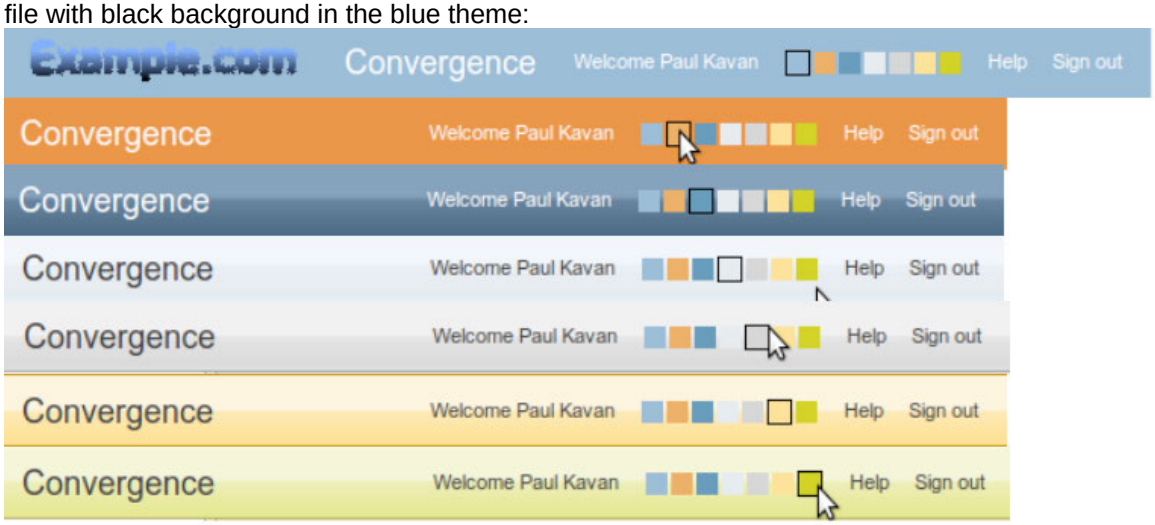
6. Edit the `/iwc_static/c11n/allDomain/themes/blue/theme.json` with the location of the logo, the logo's width, and height:

/iwc_static/c11n/allDomain/themes/blue/theme.json

```
{
logoBackground: 'transparent
url("../c11n/allDomain/themes/blue/images/logo.png") no-repeat
center center',
logoWidth: "180px"
mastheadHeight: "40px"
}
```

7. Add the following blue `logo.png` file with black background in the `/themes/blue/images/` directory:

8. Refresh your web browser. The new logo added to the blue theme results in the blue `logo.png` file with black background in the blue theme:



To Hide a Single Theme

When all themes are displayed, the user selection banner shows all the included themes: blue, orange, dark blue, light blue, grey, yellow, and green:

To hide the yellow theme, edit the `/themes/customize.js` file:

/themes/customize.js

```
iwc.api.hideTheme("theme_yellow");
```

When you remove the yellow theme, the user selection banner displays all the included themes except for the yellow theme:

To Create a New Theme

An end user might choose from a selection of themes for their UI. The Convergence UI banner (also known as the masthead) uses the `themes.attr`("store") to add themes to the banner.

The following purple theme example describes how to add new themes to the UI banner:

1. Enable Customization Server. See To Enable Customization in Convergence for these instructions.
2. Create a new theme directory named `purple` under the `/c11n/`*`domain`*`/themes` directory.

   ```
   /c11n/config.js
   /c11n/allDomain/themes/customize.js
   /c11n/allDomain/themes/purple/theme.json
   ```

   To view a sample file directory structure, see To Create the Convergence Customization Directory .

3. Make sure you are in the `iwc_static/c11n` directory.
4. If you don't already have a `config.js` file in your `/c11n` directory, copy and configure the `config.js` file from `/iwc_static/c11n_sample/config.js` to create `/c11n/config.js`.
   a. Edit the `c11n/config.js` file.
   b. Make sure that `themeEnabled: true, i18nEnabled: true`, and `jsEnabled: true` are configured for the `allDomain` configuration.
   In this example, customizations are added under the specific domain `allDomain`.

   After you edit the `c11n/config.js` file, it should look like this:

/c11n/config.js

```
dojo.provide("c11n.config");

c11n.config = {

 // allDomain configuration
 allDomain: {
  module: "allDomain",  // module name
  themeEnabled: true,  // true if theme is customized
                i18nEnabled: true,  // true if i18n is
customized
                jsEnabled: true          // true if js is
customized

  // the last entry must not end with comma
 },

 // replace sample.com for each domain configuration, change
 // domain name example.com to example_com for javascript
syntax and url syntax
 example_com: {
  module: "example_com",            // module name
  themeEnabled: false,              // true if theme is
customized
  i18nEnabled: false,               // true if i18n is
customized
  jsEnabled: false                  // true if js is
customized

  // the last entry must not end with comma

 }
}
```

5. Add the following code to `customize.js` to add the theme name, parent theme, and configuration path:

/c11n/allDomain/themes/customize.js

```
// Adding a new theme
iwc.api.addTheme({
  name: 'theme_purple',
  parentTheme: "theme_basic",
  configPath: "../c11n/allDomain/themes/purple/theme.json",
  thumbnailColor: "#C290D6"
  visible: true
});
```

6. Add the customized theme to the `resources.js` file.

- Edit the `resources.js` file in the `/iwc_static/c11n/allDomain/nls` directory to add the `Purple` theme above `last`. For example:

  /iwc_static/c11n/allDomain/nls

  ```
  {
      theme_purple : "Purple",
      last: ""
  }
  ```

  For more information on language support, see Adding a New Language.

7. Creating and adding the purple theme to the user selection banner displays a purple option after the included themes:

Making a Newly Created Theme the Default

Once you create a new theme, you can make that theme the default by following these steps:

1. Create a new theme as described in To Create a New Theme.
2. Assuming you have created the purple theme (`theme_purple`), set the user preference Convergence configuration property, `user.common.theme` to `theme_purple` with the `iwcadmin` command-line utility. For example:

   ```
   iwcadmin -o user.common.theme -v theme_purple
   ```

   For information on `user.common.theme`, see: User Preferences Configuration Properties for the Convergence Interface. For `iwcadmin` usage, see: Overview of the Convergence Command-Line Utility.
3. Restart GlassFish Server and clear the browser cache to see the change.

> **Note**
> If a user does not have the sunUCTheme value in their LDAP entry, the default theme (in this example, the purple theme) is loaded by default. See: sunUCTheme.

Chapter 30. Basic Theme Reference in Convergence 2.x

Basic Theme Reference

> **Note**
> This version of the Basic Theme Reference is for Convergence 2.x.

The basic theme provides parent template and default values for any other theme used in Convergence. All themes included in Convergence use the basic theme as their parent theme to provide a consistent look and feel amongst the different color schemes. Additionally, if you create your own theme or add your own logo, you eliminate the need to fill every value in the `theme.json` file when you incorporate the basic theme into your theme template scheme.

> **Note**
> The basic theme is hidden from the user selection as it is not intended to be used as a theme selection choice.

The following table describes the template files that make up the Basic Theme:

Table 1: Components in Basic Theme

Basic Theme Template Files	Description	Directory Location
`theme.json`	A single `.json` file that defines all the styles and colors in a theme.	`/iwc_static/c11n/themes/basic/theme.json`
`theme_basic`	`parentTheme` of all included visible themes. The `theme_basic` is made up of the `theme.json` file and the `template.css` file.	`/iwc_static/layout/themes/basic/`
`template.css`	Template file containing the css declarations with over 90 variables to be substituted from the values provided by `theme.json`. Specifically, it contains a masthead, logo, buttons, and tabs.	`/iwc_static/layout/themes/basic/`

Definition

The basic theme definition:

```
{
  name: 'theme_basic',
  configPath: "themes/basic/theme.json",
  thumbnailColor: "#FFFFFF",
  visible: false
}
```

Basic Theme Values in `theme.json` File

The following `theme.json` table describes the variables and default values derived from `theme_basic`.

Table 2: `theme.json` in Tabularized Format

Variable	Basic (Default)
templatePath	`themes/basic/template.css`
fontColor	#333333
linkColor	#3F79AA
foregroundBackground	#FFFFFF
disabledColor	#999999
disabledBackground	#CCCCCC
disabledBorderColor	#CCCCCC
buddySearchBoxBackground	transparent url('themes/basic/images/SearchBox_buddy.png') repeat-x top left
iconsClose	transparent url('themes/basic/images/Button_close.png') no-repeat top left
contentHeaderColor	#666666
warningBorderColor	#FF9C00
errorBorderColor	#FF0000
growlColor	#D5DFE8
growlContentBackground	#5A8DB9
mastheadBackground	#CCCCCC
mastheadColor	#333333
mastheadHeight	`40px`
logoWidth	`0px`
logoBackground	`transparent`
titlebarBackground	#CCCCCC
titlebarColor	#FFFFFF

tabContainerBackground	#CCCCCC
tabContainerBorderColor	#9D9D9D
betweenForegroundBackground	#F5F5F5
buttonBorderColor	#9D9D9D
buttonBackground	#CCCCCC
defaultActionButtonBorderColor	#9D9D9D
defaultActionButtonBackground	#CCCCCC
toolbarBordercolor	#9D9D9D
toolbarBackground	#F5F5F5
toolbarButtonHoverBorderColor	#9D9D9D
toolbarButtonHoverBackground	#EDEDED
selectedItemColor	#FFFFFF
selectedItemBackground	#A6A6A6
hoverItemBackground	#E8E8E8
alternatingItemBackground	#F5F5F5
contentOverlayBorderColor	#9D9D9D
borderColor	#9D9D9D
todayBorderColor	#A6A6A6
dayBorderColor	#D4D4D4
hourBorderColor	#D4D4D4
halfhourBorderColor	#EEEEEE #D4D4D4 #EEEEEE #D4D4D4
groupToggleButtonSeparatorBackground	#CCCCCC
groupToggleButtonColor	#666666
groupToggleButtonLeftBackground	#CCCCCC
groupToggleButtonRightBackground	#CCCCCC
groupToggleButtonCenterBackground	#CCCCCC
groupToggleButtonHoverColor	#666666
groupToggleButtonHoverLeftBackground	#CCCCCC
groupToggleButtonHoverRightBackground	#CCCCCC
groupToggleButtonHoverCenterBackground	#CCCCCC
groupToggleButtonSelectedColor	#666666
groupToggleButtonSelectedLeftBackground	#CCCCCC
groupToggleButtonSelectedRightBackground	#CCCCCC
groupToggleButtonSelectedCenterBackground	#CCCCCC
groupToggleButtonTodayColor	#666666

groupToggleButtonTodayLeftBackground	#CCCCCC
groupToggleButtonTodayRightBackground	#CCCCCC
groupToggleButtonTodayCenterBackground	#CCCCCC
groupToggleButtonTodayHoverColor	#666666
groupToggleButtonTodayHoverLeftBackground	#CCCCCC
groupToggleButtonTodayHoverRightBackground	#CCCCCC
groupToggleButtonTodayHoverCenterBackground	#CCCCCC
serviceNavigatorUnreadCountLeftBackground	#CCCCCC
serviceNavigatorUnreadCountRightBackground	#CCCCCC
wizardStepColor	#000
wizardStepBackground	#F5F5F5
wizardStepBorderColor	#A6A6A6
wizardStepSelectedColor	#FFF
wizardStepSelectedBackground	#A6A6A6
wizardStepBeforeSelectedBackground	#F5F5F5
tabUnselectedLeftBackground	#CCCCCC
tabUnselectedCenterBackground	#CCCCCC
tabUnselectedRightBackground	#CCCCCC
tabHoverLeftBackground	#CCCCCC
tabHoverCenterBackground	#CCCCCC
tabHoverRightBackground	#CCCCCC
tabSelectedLeftBackground	#CCCCCC
tabSelectedCenterBackground	#CCCCCC
tabSelectedRightBackground	#CCCCCC
taskbarBackground	#FFFFFF
taskbarButtonBackground	#CCCCCC
taskbarButtonHoverBackground	#99999
taskbarButtonSelectedBackground	#666666
serviceMenuItemBackground	#CCCCCC
serivceMenuItemColor	#666666
serviceMenuItemHoverBackground	#99999
serviceMenuItemHoverColor	#666666
serviceMenuItemSelectedBackground	#666666
serviceMenuItemSelectedColor	#FFFFFF
tableHeaderBackground	#CCCCCC

splitterBorderColor	#9D9D9D
splitterBackground	#E1E1E1

Chapter 31. ThemeJsonReference

Theme.json Reference

What values are supported?

- Variables ending with "Color" support css values for W3.org property 'color'
- Variables ending with "BorderColor" support css values for W3.org property 'border-color'
- Variables ending with "Background" support css values for W3.org property 'background'

Example

```
{
  // required
  // A path to a css file these values will be inserted to and applied.
  templatePath: "themes/templates/basic.css",

  fontColor: "#333333",
  linkColor: "#3F79AA",
  foregroundBackground: "#FFFFFF",

  disabledColor: "#999999",
  disabledBackground: "#CCCCCC",
  disabledBorderColor: "#CCCCCC",

  buddySearchBoxBackground: "transparent
url('themes/green/images/SearchBox_buddy.png') repeat-x top left",

  iconsClose: "transparent url('themes/green/images/Button_close.png')
no-repeat top left",

  contentHeaderColor: "#C9C600",

  warningBorderColor: "#FF9C00",
  errorBorderColor: "#FF0000",

  // usage: Notifications that appear and then disappear.
  // example: IM buddy signs on line
  growlColor: "#D5DFE8",
  growlContentBackground: "#5A8DB9",

  // Masthead
  mastheadBackground: "url('themes/green/images/Masthead.png')",
  mastheadColor: "#333333",
  mastheadHeight: "40px",

  // Masthead logo
  logoWidth: "0px",
```

```
  logoBackground: "transparent",

  // Titlebar
  titlebarBackground: "transparent
url('themes/green/images/titlebar_serviceMenu_selected.png') repeat-x
left 50%",
  titlebarColor: "#FFFFFF",

  // Tab container
  tabContainerBackground: "url('themes/green/images/TabContainer.png')",
  tabContainerBorderColor: "#B0A954",

  betweenForegroundBackground: "#F5F5F5",

  // button
  buttonBorderColor: "#B0A954",
  buttonBackground: "transparent url('themes/green/images/Button.png')
repeat-x bottom left",

  // Default Action Button
  // usage: dialog submit button, form submit buttons
  // This is the style for the action that will occur if the user submit
the form or dialog by pressing enter.
  // Or the default action the user will likely take.
  defaultActionButtonBorderColor: "#B0A954",
  defaultActionButtonBackground: "transparent
url('themes/green/images/Button_default_action.png') repeat-x bottom
left",

  // Toolbar
  toolbarBorderColor: "#B0A954",
  toolbarBackground: "#F5F5F5",

  toolbarButtonHoverBorderColor: "#B0A954",
  toolbarButtonHoverBackground: "#F3F5CF",

  // Selected Item
  selectedItemColor: "#FFFFFF",
  selectedItemBackground: "#C9C600",

  // Hover Item
  hoverItemBackground: "#E8E8E8",

  alternatingItemBackground: "#F5F5F5",

  // Content Overlay
  // summary: Used for content which displays on top of the main content
area but does not take the entire screen space.
  // usage: IM Dialogs, Dialogs, drop down menus, context menus: outer
border color
  contentOverlayBorderColor: "#B0A954",

  borderColor: "#B0A954",

  // Calendars
  todayBorderColor: "#B0A954",
```

```
  dayBorderColor: "#CCD7DF",
  hourBorderColor: "#CCD7DF",
  halfhourBorderColor: "#EBEFF1 #CCD7DF #EBEFF1 #CCD7DF",

  // Group Toggle Buttons
  // a group of buttons where only one can be selected at a time.
  groupToggleButtonSeparatorBackground: "transparent
url('themes/green/images/GroupToggle_separator.png') repeat-y top left",

  groupToggleButtonColor: "#666666",
  groupToggleButtonLeftBackground: "transparent
url('themes/green/images/GroupToggle_left_unselected.png') no-repeat top
left",
  groupToggleButtonRightBackground: "transparent
url('themes/green/images/GroupToggle_right_unselected.png') no-repeat
top right",
  groupToggleButtonCenterBackground: "transparent
url('themes/green/images/GroupToggle_middle_unselected.png') repeat-x
top center",

  groupToggleButtonHoverColor: "#666666",
  groupToggleButtonHoverLeftBackground: "transparent
url('themes/green/images/GroupToggle_left_hover.png') no-repeat top
left",
  groupToggleButtonHoverRightBackground: "transparent
url('themes/green/images/GroupToggle_right_hover.png') no-repeat top
right",
  groupToggleButtonHoverCenterBackground: "transparent
url('themes/green/images/GroupToggle_middle_hover.png') repeat-x top
center",

  groupToggleButtonSelectedColor: "#666666",
  groupToggleButtonSelectedLeftBackground: "transparent
url('themes/green/images/GroupToggle_left_selected.png') no-repeat top
left",
  groupToggleButtonSelectedRightBackground: "transparent
url('themes/green/images/GroupToggle_right_selected.png') no-repeat top
right",
  groupToggleButtonSelectedCenterBackground: "transparent
url('themes/green/images/GroupToggle_middle_selected.png') repeat-x top
center",

  groupToggleButtonTodayColor: "#666666",
  groupToggleButtonTodayLeftBackground: "transparent
url('themes/green/images/GroupToggle_left_previous.png') no-repeat top
left",
  groupToggleButtonTodayCenterBackground:"transparent
url('themes/green/images/GroupToggle_middle_unselected.png') repeat-x
top center",
  groupToggleButtonTodayRightBackground: "transparent
url('themes/green/images/GroupToggle_right_next.png') no-repeat top
right",

  groupToggleButtonTodayHoverColor: "#666666",
  groupToggleButtonTodayHoverLeftBackground: "transparent
url('themes/green/images/GroupToggle_left_previous_hover.png') no-repeat
```

```
top left",
  groupToggleButtonTodayHoverCenterBackground: "transparent
url('themes/green/images/GroupToggle_middle_hover.png') repeat-x top
center",
  groupToggleButtonTodayHoverRightBackground: "transparent
url('themes/green/images/GroupToggle_right_next_hover.png') no-repeat
top right",

  // Mail Unread Count in service menu
  serviceNavigatorUnreadCountLeftBackground: "transparent
url('themes/green/images/Splash_count_left.png') no-repeat left center",
  serviceNavigatorUnreadCountRightBackground: "transparent
url('themes/green/images/Splash_count_right.png') no-repeat right
center",

  // Wizard
  // Usage: Options > Mail > External Accounts > new account
  wizardStepColor: "#000",
  wizardStepBackground: "#ECEFAD
url('themes/green/images/Wizard_step_unselected.png') no-repeat center
right",
  wizardStepBorderColor: "#C9C600",
  wizardStepSelectedColor: "#FFF",
  wizardStepSelectedBackground: "#C9C600
url('themes/green/images/Wizard_step_selected.png') no-repeat center
right",
  wizardStepBeforeSelectedBackground: "#ECEFAD
url('themes/green/images/Wizard_step_beforeSelected.png') no-repeat
center right",

  // Tabs
  tabUnselectedLeftBackground: "transparent
url('themes/green/images/Tab_left_unselected.png') no-repeat left top",
  tabUnselectedCenterBackground: "transparent
url('themes/green/images/Tab_middle_unselected.png') repeat-x left top",
  tabUnselectedRightBackground: "transparent
url('themes/green/images/Tab_right_unselected.png') no-repeat right
top",
  tabHoverLeftBackground: "transparent
url('themes/green/images/Tab_left_hover.png') no-repeat left top",
  tabHoverCenterBackground: "transparent
url('themes/green/images/Tab_middle_hover.png') repeat-x left top",
  tabHoverRightBackground: "transparent
url('themes/green/images/Tab_right_hover.png') no-repeat right top",
  tabSelectedLeftBackground: "transparent
url('themes/green/images/Tab_left_selected.png') no-repeat left top",
  tabSelectedCenterBackground: "transparent
url('themes/green/images/Tab_middle_selected.png') repeat-x left top",
  tabSelectedRightBackground: "transparent
url('themes/green/images/Tab_right_selected.png') no-repeat right top",

  // Taskbar IM
  taskbarBackground:
"url('themes/green/images/GroupToggle_middle_unselected.png')",
  taskbarButtonBackground: "transparent
url('themes/green/images/GroupToggle_middle_unselected.png') repeat-x 0
```

```
50%",
  taskbarButtonHoverBackground: "transparent
url('themes/green/images/chat_tab_hover.png') repeat-x 0 50%",
  taskbarButtonSelectedBackground: "transparent
url('themes/green/images/GroupToggle_middle_selected.png') repeat-x 0
50%",

  // Service Menu
  serviceMenuItemBackground: "transparent
url('themes/green/images/ServiceMenu_unselected.png') repeat-x 0 50%",
  serviceMenuItemColor: "#666666",
  serviceMenuItemHoverBackground: "transparent
url('themes/green/images/ServiceMenu_hover.png') repeat-x 0 50%",
  serviceMenuItemHoverColor: "#666666",
  serviceMenuItemSelectedBackground: "transparent
url('themes/green/images/titlebar_serviceMenu_selected.png') repeat-x 0
50%",
  serviceMenuItemSelectedColor: "#FFFFFF",

  // Table Header
  tableHeaderBackground: "transparent
url('themes/green/images/ServiceMenu_unselected.png') repeat-x 0 50%",

  // Splitter
```

```
  splitterBorderColor: "#B0A954",
  splitterBackground: "#ECEFAD"
}
```

Chapter 32. Customization Example - Setting Themes in Anonymous Calendar

Customization Example - Setting Themes in Anonymous Calendar

The following example describes how to set themes in for an anonymous calendar:

1. Enable customized anonymous calendar by configuring the `calendar.html` layout page. See the calendar section in: Customizing Layout HTML Pages.
2. Add theme customizations to `calendar.html` by adding the following lines before `iwc.themes.loadSelectedItem();`. In this example, all the default themes are hidden and a new purple theme is added:

```
// Replicating C11n/allDomain/themes/customize.js
// How to show a custom theme for calendar.html

// Step 1: hide all themes provided themes
iwc.api.hideTheme("theme_blue");
iwc.api.hideTheme("theme_dark_blue");
iwc.api.hideTheme("theme_green");
iwc.api.hideTheme("theme_grey");
iwc.api.hideTheme("theme_light_blue");
iwc.api.hideTheme("theme_orange");
iwc.api.hideTheme("theme_yellow");

// Step 2: Add new theme
// This will be the only theme visible and default selected theme.
iwc.api.addTheme({
name: "theme_purple", // This must be unique
parentTheme: "theme_basic", // must use one of the available themes
basic
configPath: "../c11n/allDomain/themes/purple/theme.json",
thumbnailColor: "#C290D6",
visible: true
});
```

3. Refresh the web browser.
   - The theme you added with `iwc.api.addTheme` is displayed.
   - If the banner of the new theme is too small, use the next step to resolve this issue
4. **Optional--** Follow this step if the banner in the new theme is too small.
   a. Add css style in the `calendar.html` head section by adding the following lines inside the HTML head element:

```
<style type="text/css">
/* overwriting CalendarApplication.css to use theme banner
styles */
.CalendarApplication .Banner {
height: auto;
}
</style>
```

b. Refresh the web browser.

Chapter 33. Disabling Event Balloon User Input From Being Saved as an Event Description

Customization Example - Disabling Event Balloon User Input From Being Saved as an Event Description

The following example disables the event balloon user input from also being saved as an event description. With this customization, the event title is added per user input but a blank event description displays.

1. Enable client customizations.

```
./iwcadmin -o client.enablecustomization -v true
```

2. Create the following directory structure under the `appserver-domain-root/docroot/iwc_static/` directory to hold the customization files.

```
c11n/
c11n/allDomain
c11n/allDomain/js
c11n/allDomain/js/widget
c11n/allDomain/js/widget/calendar
```

3. Generate the general customization configuration file.
This configuration file enables JavaScript customization (`jsEnabled: true`) across all domains:

c11n/config.js (module: "allDomain")

```
dojo.provide("c11n.config");

c11n.config = {

        // allDomain configuration
        allDomain: {
                module: "allDomain",           // module name
                themeEnabled: false,           // true if theme is
customized
                i18nEnabled: false,            // true if i18n is
customized
                jsEnabled: true                // true if js is
customized

                // the last entry must not end with comma

        }
}
```

4. Create and add to the domain specific customization file.

c11n/allDomain/js/customize.js

```
dojo.provide("c11n.allDomain.js.customize");

// Disable event balloon user input from being saved as event
description
dojo.require("c11n.allDomain.js.widget.calendar.Balloon");
```

5. Create the JavaScript file that disables event balloon user input from being saved as event description:

c11n/allDomain/js/widget/calendar/Balloon.js

```
dojo.provide("c11n.allDomain.js.widget.calendar.Balloon");

dojo["require"]("iwc.widget.calendar.Balloon");

dojo.declare("iwc.widget.calendar.QuickEventBalloon",
iwc.widget.calendar.QuickEventBalloon, {

    doSave: function() {
                this.evtObject.desc = "";
                this.inherited(arguments);
    },

    _edit: function() {
                if(this.evtObject != null) {
                    this.evtObject.desc = "";
                }
                this.inherited(arguments);
    },
    last: ""
});
```

6. Restart GlassFish Server and clear the browser cache to see the change.

Chapter 34. Integrating Third-Party Applications

Integrating Third-Party Applications

> **Note**
> To integrate third-party applications into Convergence 1.x, see Integrating Third-Party Applications in Convergence 1.x.

Convergence provides access to the following back-end services: mail, address book, calendar, and instant messaging (IM). You can also integrate additional, third-party applications into Convergence. To end users, the application appears in the UI as another component, just like mail or calendar.

To add a third-party application (new service), you normally

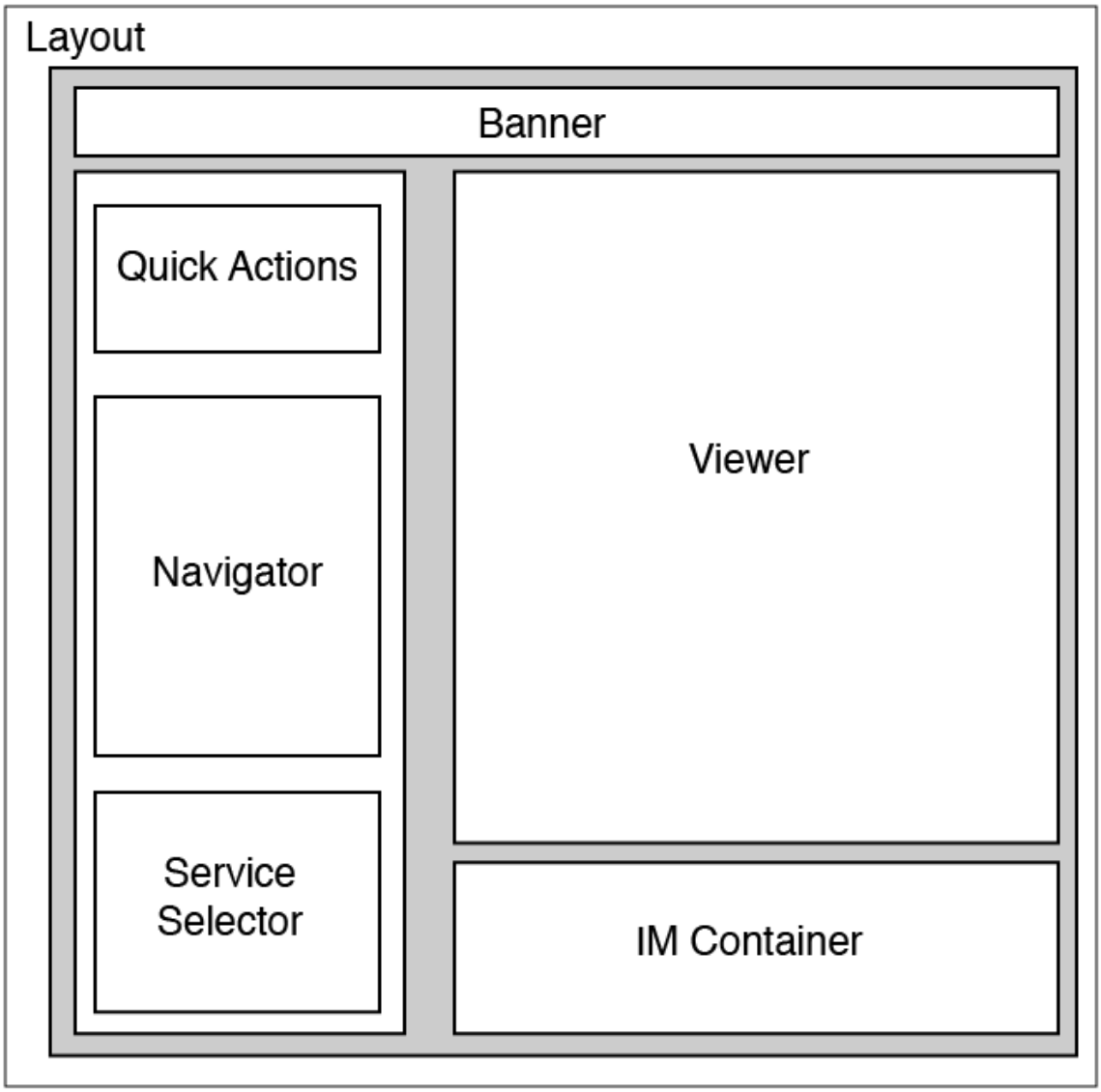
- Create a menu button to select the service
- Create a new service navigator widget
- Create a new service viewer widget

The preceding steps involve creating custom widgets.

You must have experience with dojo to create and customize widgets in Convergence. For general information about customizing widgets, see Customizing Convergence: Technical Overview.

The following figure shows the Convergence UI layout, including the regions of the UI that can be used by an integrated third-party application.

Figure 1 Integrating a 3rd-Party Application into Convergence: Convergence UI Layout



Example: Adding HelloConvergence to Convergence

The following example shows how a third-party application, HelloConvergence, is added to Convergence. Take the following steps to integrate HelloConvergence as a new service in Convergence:

1. Enable customization in the Convergence Server.
   Before you can customize the UI, the Convergence Server must be enabled for customization. Use the command-line utility, `iwcadmin`, to set the `client.enablecustomization` parameter to `true`. For example:

   ```
   ./iwcadmin -W <password file> -o client.enablecustomization -v true
   ```

2. Create the customization directory, `/c11n`.
   All customizations must be located under this directory.

   ```
   /iwc_static/c11n
   ```

3. Create the configuration file, `config.js`, for the customization.

```
/c11n/config.js
```

You can create this file by copying an example of the `config.js` file from the sample customization directory:

```
/iwc_static/c11n_sample
```

In this example, the `config.js` file enables JavaScript customization (`jsEnabled: true`) across all domains (`module: "allDomain"`). Additionally, you can enable the following options:

- theme customization (`themeEnabled: true`) to customize themes
- i18n (`i18nEnabled: true`) to customize localized versions and to pick up the button. For this specific example, JavaScript customization and i18n are enabled.

c11n/config.js

```
dojo.provide("c11n.config");

c11n.config = {

        // allDomain configuration
        allDomain: {
                module: "allDomain",           // module name
                themeEnabled: false,           // true if
theme is customized
                i18nEnabled: true,            // true if i18n
is customized
                jsEnabled: true                // true if js
is customized

                // the last entry must not end with comma

        }
}
```

4. Create the following subdirectory under the customization (`c11n`) directory:

```
iwc_static/c11n/<domain>/js
```

where *domain* is the domain in which the customized banner will be available.

5. Create a `customize.js` file, as follows:

```
iwc_static/c11n/<domain>/js/customize.js
```

You can create this file by copying an example of the `customize.js` file from the sample customization directory, `/iwc_static/c11n_sample`. For example, you could copy the file from:

```
iwc_static/c11n_sample/allDomain/js
```

At runtime, this file is responsible for loading the javascript customizations (the custom widget files) in this subdirectory. For details, see How the `customize.js` File Manages Customized Widgets.

6. Create a service subdirectory:

```
iwc_static/c11n/<domain>/js/service
```

7. Create a subdirectory to contain the custom widgets:

```
iwc_static/c11n/<domain>/js/widget
```

8. Create the following three files in these subdirectories:

```
iwc_static/c11n/<domain>/js/service/HelloConvergence.js

iwc_static/c11n/<domain>/js/widget/helloConvergence/Navigator.js

iwc_static/c11n/<domain>/js/widget/helloConvergence/ViewerContainer.j
```

9. Add the following code to the `iwc_static/c11n/<domain>/js/customize.js` file.

```
// Add HelloConvergence as a barebone service

     iwc.topicNames["helloConvergence"] = {
         startupComplete: "service/startupComplete",
         serviceReady: "service/serviceReady",
         last: ""
     };

     helloConvergenceService = {
       name: "helloConvergence",
       enabled: true,
       displayName: "Hello Convergence",
       packageName: "c11n.allDomain.js.service.HelloConvergence",
       className: "c11n.allDomain.js.service.HelloConvergence",
       options:{
       }
     };

     dojo.require("c11n.allDomain.js.service.HelloConvergence");
     iwc.api.addService(helloConvergenceService);

iwc.api.setServiceMenuDisplayOrder(helloConvergenceService.name,
1);
```

10. Add the following dojo statements to the `c11n/allDomain/js/service/HelloConvergence.js` file.

```
dojo.provide("c11n.allDomain.js.service.HelloConvergence");

dojo.["require"]("iwc.api");
dojo.["require"]("iwc.service.ServiceBase");

dojo.require("c11n.allDomain.js.widget.helloConvergence.Navigator"); (
[iwc.service.ServiceBase],
  {
    // new properties
  }
);
```

11. Add the following `dojo.declare` statement to the `c11n/allDomain/js/widget/helloConvergence/Navigator.js` file.

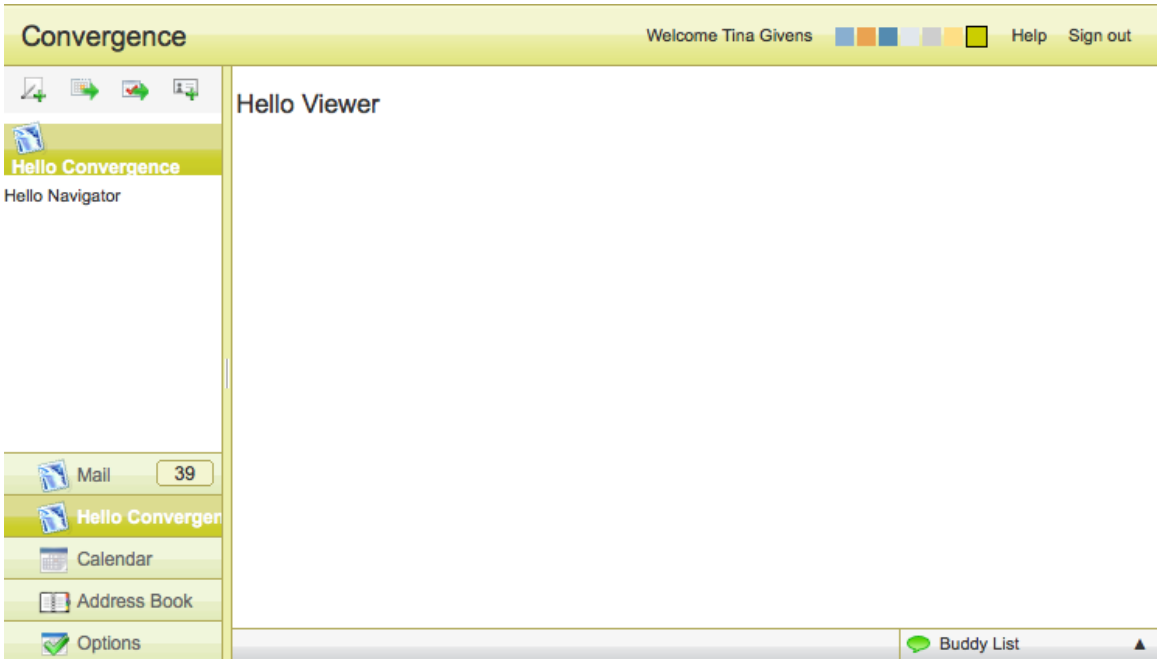
```
dojo.declare(
     "c11n.allDomain.js.widget.helloConvergence.Navigator",
     [ dijit.layout._LayoutWidget, dijit._Templated ],
     {
       // new properties
     }
);
```

12. Add the following `dojo.declare` statement to the `c11n/allDomain/js/widget/helloConvergence/ViewerContainer.js` file.

```
dojo.declare(
"c11n.allDomain.js.widget.helloConvergence.ViewerContainer",
[ dijit.layout._LayoutWidget, dijit._Templated ],
  {
   // new properties
  }
);
```

For user who wants to experiment adding new service, there is a simple bare bone service named HelloConvergence. This sample gives user a starting point to try out the Convergence service architecture. The following figure shows an example of integrating a bare-bones, proof-of-concept application into the Convergence UI.

Figure 2 Integrating a Simple, Proof-of-Concept 3rd-Party Application into Convergence:


Convergence
Welcome Tina Givens
Help
Sign out
Hello Viewer
Hello Convergence
Hello Navigator
Mail
39
Hello Converger
Calendar
Address Book
Options
Buddy List

Chapter 35. Making the Customized Logo a Clickable Link in the Banner

Customization Example - Making the Customized Logo a Clickable Link in the Banner

This example extends the customization scenarios outlined in Customizing Themes and Banner in Convergence 2.x to make the logo in a clickable link in the banner.

As with other Convergence elements, your customization can apply to a particular domain, or it can be applied to all domains. The following example uses the `allDomain` directory to customize the banner in all domains in the deployment:

1. Add a logo to your banner. Follow the instructions in Adding a Logo.
2. Be sure that customization is enabled in the Convergence Server.
   Use the command-line utility, `iwcadmin`, to set the `client.enablecustomization` parameter to `true`. For example:

   ```
   ./iwcadmin -W <password file> -o client.enablecustomization -v true
   ```

3. Be sure the following directories have been created under the *appserver-domain-root* `/docroot/iwc_static/` directory to hold the customization files.

   ```
   c11n/
   c11n/allDomain
   c11n/allDomain/js
   c11n/allDomain/js/widget
   ```

4. Be sure that the general customization configuration file has been created: `config.js`. (You can copy this file from the `c11n_sample` directory.)
5. In the `config.js` configuration file, set the `jsEnabled` flag to `true` across all domains (`"allDomain"`). For example:

c11n/config.js

```
dojo.provide("c11n.config");

c11n.config = {

        // allDomain configuration
        allDomain: {
                module: "allDomain",            // module name
                themeEnabled: false,            // true if theme is
customized
                i18nEnabled: false,             // true if i18n is
customized
                jsEnabled: true                 // true if js is
customized

                // the last entry must not end with comma

        }
}
```

6. Create the domain specific customization file.

c11n/allDomain/js/customize.js

```
dojo.provide("c11n.allDomain.js.customize");

//Enable the Banner.js to make the logo in a clickable link in the
banner .
dojo.require("c11n.allDomain.js.widget.Banner");
```

7. Create the JavaScript file to make the logo in a clickable link in the banner by using either of the following `postCreate` functions, and optionally, either of the following `logoOnClick` functions:
   - The first `postCreate` function makes the logo clickable in the banner.
   - The second `postCreate` function (which is commented out in the example) makes the mouse pointer clickable when it hovers over the logos. Uncomment the function in order to make the mouse pointer clickable when it hovers over logos. Comment out the first `postCreate` function.
   - The first `logoOnClick` function creates an alert when the logo has been clicked.
   - The second `logoOnClick` function (which is commented out in the example) links the logo to a URL. Uncomment the function in order to link the logo to a URL. Comment out the first `logoOnClick` function.

c11n/allDomain/js/widget/Banner.js

```
dojo.provide("c11n.allDomain.js.widget.Banner");
dojo.require("iwc.widget.Banner");

dojo.declare("iwc.widget.Banner", iwc.widget.Banner, {
	//postCreate #1: makes logo clickable
	postCreate: function() {
		this.inherited(arguments);

		// find the logo element
		var elem = dojo.query(".Banner-Logo",
this.domNode);
		if (elem.length == 1) {
			this.connect(elem[0], "onclick",
"logoOnClick");
		}
	},

	//postCreate #2: makes the mouse pointer clickable when
hovering over links. Comment out the postCreate #1 function
and uncomment the following function:
	//postCreate: function() {
	//	this.inherited(arguments);
	//
	//	var elem = dojo.query(".Banner-Logo",
this.domNode);  // find the logo element
	//	if (elem.length == 1) {
	//		elem[0].style.cursor = "pointer";
// change the cursor to pointer
	//		this.connect(elem[0], "onclick",
"logoOnClick");
	//	}
	// },

	// logoOnClick #1: creates a clickable logo
	logoOnClick: function() {
		alert("your logo is clicked!");
	},

	// logoOnClick #2: links the logo to a URL. Comment out
the logoOnClick #1 function and uncomment the following
function, where http://www.example.com is the URL to which the
logo is linked:
	// logoOnClick: function() {
	//	window.open("http://www.example.com");
	// },

	last: ""

});
```

8. Restart GlassFish Server and clear the browser cache to see the change.

Chapter 36. Removing Change Password, Mail Vacation Message, and Calendar Notification Options

Customization Example - Removing the Change Password, Vacation Message, and Calendar Notification Options

This Convergence example assumes that the reader has thoroughly read the Convergence Customization Guide.

Note
To remove change password and vacation message options in Convergence 1.x, see Removing Change Password and Vacation Message Options in Convergence 1.x.

The following example removes the "Change Password" and "Vacation Message" menu items in the Convergence Options tab. This is a useful example for sites that need to restrict the ability to modify specific account settings through the Convergence interface.

1. Enable client customizations.

```
./iwcadmin -o client.enablecustomization -v true
```

2. Create the following directory structure under the *appserver domain root* `/docroot/iwc_static/` directory to hold the customization files.

```
c11n/
c11n/allDomain
c11n/allDomain/js
c11n/allDomain/js/service
```

3. Generate the general customization configuration file.
   This configuration file enables JavaScript customization (`jsEnabled: true`) across all domains (`module: "allDomain"`).

c11n/config.js

```
dojo.provide("c11n.config");

c11n.config = {

        // allDomain configuration
        allDomain: {
                module: "allDomain",           // module name
                themeEnabled: false,           // true if theme is
customized
                i18nEnabled: false,            // true if i18n is
customized
                jsEnabled: true                // true if js is
customized

                // the last entry must not end with comma

        }
}
```

4. Create the domain specific customization file (if it does not already exist).

c11n/allDomain/js/customize.js

```
dojo.provide("c11n.allDomain.js.customize");

dojo.require("c11n.allDomain.js.widget.option.Navigator");
```

5. Create the JavaScript file that removes the following options: Change Password, Mail Vacation Message, and Calendar Notifications.

c11n/allDomain/js/widget/option/Navigator.js

```
dojo.provide("c11n.allDomain.js.widget.option.Navigator");

dojo["require"]("iwc.api");
dojo["require"]("iwc.widget.option.Navigator");

dojo.declare("iwc.widget.option.Navigator",
iwc.widget.option.Navigator,
       {
                postCreate: function() {
                        // remove the change password option
                        iwc.api.removeOption("/Global/Change
Password");

                        // remove the Mail vacation message option
                        iwc.api.removeOption("/Mail/Local
Account/Vacation Message");

                        // remove the Calendar Notifcations option

iwc.api.removeOption("/Calendar/Notifications");

                        // call the superclass postCreate
                        this.inherited(arguments);
                },

                last: ""
       }
);
```

6. Restart GlassFish Server and clear the browser cache to see the change.

Chapter 37. Removing Reply-To Address Option

Customization Example - Removing the Reply-To: Address Option

Comms Community Verified Contribution
This article came from the Comms Community and has been verified by the Oracle Communications Unified Communications Suite product team.

This Convergence example assumes that the reader has thoroughly read the Convergence Customization Guide.

Note
To remove reply-to: address options in Convergence 1.x, see Removing Reply-To Address Option in Convergence 1.x.

The following example hides the `Reply-To:` Account Setting in the Mail option for users in all domains.

1. Enable client customizations

```
./iwcadmin -o client.enablecustomization -v true
```

2. Create the following directory structure under <appserver domain root>/docroot/iwc_static/ to hold the customization files.

```
c11n/
c11n/allDomain
c11n/allDomain/js
c11n/allDomain/js/widget
c11n/allDomain/js/widget/mail
c11n/allDomain/js/widget/mail/option
```

   The `mail.option.Identity` widget that is being modified in this example can be viewed here.

3. Generate the general customization configuration file.
   This configuration file enables JavaScript customization (`jsEnabled: true`) across all domains (`module: "allDomain"`).

c11n/config.js

```
dojo.provide("c11n.config");

c11n.config = {

        // allDomain configuration
        allDomain: {
                module: "allDomain",           // module name
                themeEnabled: false,           // true if theme is
customized
                i18nEnabled: false,            // true if i18n is
customized
                jsEnabled: true                // true if js is
customized

                // the last entry must not end with comma

        }
}
```

4. Create the domain specific customization file (if it doesn't already exist).

c11n/allDomain/js/customize.js

```
dojo.provide("c11n.allDomain.js.customize");

// Hide replyTo Option from the Mail Account Settings
dojo.require("c11n.allDomain.js.widget.mail.option.Identity");
```

5. Create the JavaScript file which hides the `Reply-To:` option. This example makes use of the `dojoAttachPoint="replyTo"` value in the `<appserver domain root>/docroot/iwc_static/js/debug/iwc/widget/templates/mail/option/Identi` file to determine the appropriate location in the DOM-tree to make the modification.

c11n/allDomain/js/widget/mail/option/Identity.js

```
dojo.provide("c11n.allDomain.js.widget.mail.option.Identity");
dojo.require("iwc.widget.mail.option.Identity");
dojo.declare("iwc.widget.mail.option.Identity",
iwc.widget.mail.option.Identity, {

        buildRendering: function() {
                this.inherited(arguments);

                // Hide the replyTo option field and the associated
text
                var replyToParent =
this.replyTo.domNode.parentNode;

                // go up one level because this.replyTo is a
container widget
                replyToParent.parentNode.style.display = "none";

        },

        last: ""

});
```

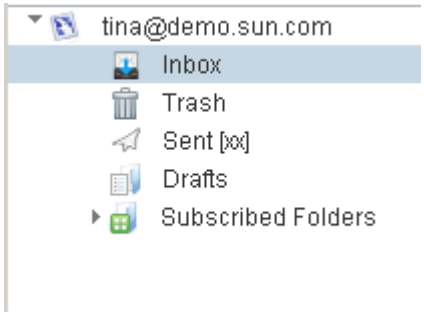
6. Restart GlassFish Server and clear the browser cache to see the change.

Chapter 38. Modifying Mail Folder Icons in the Service Navigator

Customization Example - Modifying Mail Folder Icons in the Service Navigator

This example describes how to customize the mail folder icons in the service navigator.

Default Mail Folder Icons
The following screenshot shows the default mail folder icons:



As with other Convergence elements, your customization can apply to a particular domain, or it can be applied to all domains. The following example uses the `allDomain` directory to modify the mail folder icons in all domains in the deployment:

1. Enable customization in the Convergence Server.
   Before you can customize the UI, the Convergence Server must be enabled for customization. Use the command-line utility, `iwcadmin`, to set the `client.enablecustomization` parameter to `true`. For example:

   ```
   ./iwcadmin -W <password file> -o client.enablecustomization -v true
   ```

2. Create the following directory structure under the *appserver-domain-root* `/docroot/iwc_static/` directory to hold the customization files.

   ```
   c11n/
   c11n/allDomain
   c11n/allDomain/js
   c11n/allDomain/layout
   c11n/allDomain/layout/css
   ```

3. Create the general customization configuration file, `config.js`. (You can copy this file from the `c11n_sample` directory.)
4. In the `config.js` configuration file, set the `jsEnabled` flag to `true` across all domains (`"allDomain"`). For example:

c11n/config.js

```
dojo.provide("c11n.config");

c11n.config = {

        // allDomain configuration
        allDomain: {
                module: "allDomain",            // module name
                themeEnabled: false,            // true if theme is
customized
                i18nEnabled: false,             // true if i18n is
customized
                jsEnabled: true                 // true if js is
customized

                // the last entry must not end with comma

        }
}
```

5. Create the domain specific customization file.

c11n/allDomain/js/customize.js

```
dojo.provide("c11n.allDomain.js.customize");

// Load Customized CSS File Helper
var loadCustomizedCssFile = function(url, id) {
        if (!id){
                id = url.replace(/\.\./gm, "").replace(/\//gm,
"_").replace(/\./gm, "_");
        }
        var link = window.document.getElementById(id);
        if (! link) {
                link = window.document.createElement("link");
                link.setAttribute("id", id);
                link.setAttribute("type", "text/css");
                link.setAttribute("rel", "stylesheet");
                var head =
window.document.getElementsByTagName("head")[0];
                head.appendChild(link);
        }

        link.setAttribute("href", url + "?" + djConfig.cacheBust);
}

// Load customized css file c11n_icons
loadCustomizedCssFile("../c11n/allDomain/layout/css/c11n_icons.css");
```

6. Add new icons or an icon sprite (a single image that contains multiple icons) to the `c11n/allDomain/layout/images/` directory.
   The following table lists the default mail icons and equivalent new, red mail folder icons for the service navigator that are being added in this example:

Default and New Icons in Service Navigator Example

Icon	Default Image	New Image	New Image File Name
Root Folder			`options.IconMail_new_red.png`
Inbox Folder			`optionsIconMail_new_red.png`
Shared Folder and Trash Folder Sprite	   	   	`MailFolders_new_red.png`
Sent Folder			`treeMailSent_new_red.png`
Drafts Folder			`treeMailDrafts_new_red.png`

7. To change all the folder icons, create a new css file: `clln_icons.css` in the `c11n/allDomain/layout/css/` directory. In this example, the `background-image` points to the new images in `c11n/allDomain/layout/images/`:

> **Comms Community Contribution**
> The following example (To Change the Personal Mail Folder Icon) has been received from the Comms Community. It has not yet been verified by Oracle. The rest of the examples in this step have been verified by Oracle.

To Change the Personal Mail Folder Icon

c11n/allDomain/layout/css/c11n_icons.css

```
/* Personal Mail Icon */
.FolderIcons {
   width: 16px;
   height: 16px;
   background-repeat:no-repeat;
   background-position:top left;

background-image:url("../images/treeMailFolderPersonal_new_red.png");
```

To Change the Root Folder Icon

c11n/allDomain/layout/css/c11n_icons.css

```
/* Root Folder Icon */
.FolderIcons_Root {
  background-image: url("../images/optionsIconMail_new_red.png");
  background-repeat: no-repeat;
  background-position: center center;
  background-color: transparent;
}
```

To Change the Inbox Folder Icon

c11n/allDomain/layout/css/c11n_icons.css

```
/* Inbox Folder Icon */
.FolderIcons_Inbox {
  background-image: url("../images/treeMailInbox_new_red.png");
  background-repeat: no-repeat;
  background-position: center center;
  background-color: transparent;
}
```

To Change the Shared Folder Icon

c11n/allDomain/layout/css/c11n_icons.css

```
/* Trash Folder Icon */
.FolderIcons_Shared {
  background-image: url("../images/MailFolders_new_red.png");
  background-repeat: no-repeat;
  background-position: 0 0px;
  background-color: transparent;
}
```

To Change the Trash Folder Icon

c11n/allDomain/layout/css/c11n_icons.css

```
/* Trash Folder Icon */
.FolderIcons_Trash {
  background-image: url("../images/MailFolders_new_red.png");
  background-repeat: no-repeat;
  background-position: 0 -102px;
  background-color: transparent;
}
```

To Change the Sent Folder Icon

c11n/allDomain/layout/css/c11n_icons.css

```
/* Sent Folder Icon */
.FolderIcons_Sent {
  background-image: url("../images/treeMailSent_new_red.png");
  background-repeat: no-repeat;
  background-position: center center;
  background-color: transparent;
}
```

To Change the Drafts Folder Icon

c11n/allDomain/layout/css/c11n_icons.css

```
/* Drafts Folder Icon */
.FolderIcons_Drafts {
  background-image: url("../images/treeMailDrafts_new_red.png");
  background-repeat: no-repeat;
  background-position: center center;
  background-color: transparent;
}
```

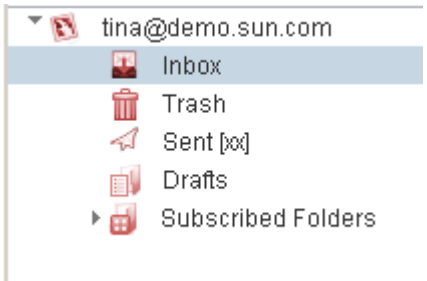
To Change the Subscribe to Folder Icon

c11n/allDomain/layout/css/c11n_icons.css

```
/* Subscribe to Folder Icon */
.FolderIcons_Subscribed {
  background-image: url("../images/MailFolders_new_red.png");
  background-repeat: no-repeat;
  background-position: 0 -68px;
  background-color: transparent;
}
```

8. Restart GlassFish Server to clear the browser cache and view the change.

After Changing the Mail Icons in the Service Navigator
This screenshot shows the new mail icons in the service navigator:



Chapter 39. Removing Reservations from the New Event Tab

Customization Example - Removing Reservations from the New Event Tab

The following example removes the Reservations from the New Event Tab:

1. Enable client customizations.

```
./iwcadmin -o client.enablecustomization -v true
```

2. Create the following directory structure under the *appserver-domain-root* `/docroot/iwc_static/` directory to hold the customization files.

```
c11n/
c11n/allDomain
c11n/allDomain/js
c11n/allDomain/js/widget
```

3. Generate the general customization configuration file.
   This configuration file enables JavaScript customization (`jsEnabled: true`) across all domains (`module: "allDomain"`).

c11n/config.js

```
dojo.provide("c11n.config");

c11n.config = {

        // allDomain configuration
        allDomain: {
                module: "allDomain",           // module name
                themeEnabled: false,           // true if theme is
customized
                i18nEnabled: false,            // true if i18n is
customized
                jsEnabled: true                // true if js is
customized

                // the last entry must not end with comma

        }
}
```

4. Create the domain specific customization file (if it doesn't already exist).

c11n/allDomain/js/customize.js

```
dojo.provide("c11n.allDomain.js.customize");

// Remove Reservations from the New Event Tab
dojo.require("c11n.allDomain.js.widget.calendar.CreateEvent");
```

5. Create the JavaScript file that removes Reservations from the New Event Tab and the Reservations label.

c11n/allDomain/js/widget/calendar/CreateEvent.js

```
dojo.provide("c11n.allDomain.js.widget.calendar.CreateEvent");
dojo["require"]("iwc.widget.calendar.CreateEvent");
dojo.declare("iwc.widget.calendar.CreateEvent",
iwc.widget.calendar.CreateEvent, {

         postCreate: function () {
                this.inherited(arguments);

                dojo.style(this.reservationNode.domNode,
"display", "none");

                // Remove the "Reservations" Label
                var reservationParent =
this.reservationNode.domNode.parentNode;

reservationParent.removeChild(dojo.query("h3",reservationParent)[1]);
},

         last: ""

});
```

- Restart GlassFish Server and clear the browser cache to see the change.

Chapter 40. Removing the Attachment Button in the New Task Tab

Customization Example - Removing the Attachment Button in the New Task Tab

The following example removes the Attachment Button in the New Task Tab:

1. Enable client customizations.

```
./iwcadmin -o client.enablecustomization -v true
```

2. Create the following directory structure under the *appserver-domain-root* `/docroot/iwc_static/` directory to hold the customization files.

```
c11n/
c11n/allDomain
c11n/allDomain/js
c11n/allDomain/js/widget
```

3. Generate the general customization configuration file.
   This configuration file enables JavaScript customization (`jsEnabled: true`) across all domains (`module: "allDomain"`).

c11n/config.js

```
dojo.provide("c11n.config");

c11n.config = {

        // allDomain configuration
        allDomain: {
                module: "allDomain",           // module name
                themeEnabled: false,           // true if theme is
customized
                i18nEnabled: false,            // true if i18n is
customized
                jsEnabled: true                // true if js is
customized

                // the last entry must not end with comma

        }
}
```

4. Create the domain specific customization file (if it doesn't already exist).

c11n/allDomain/js/customize.js

```
dojo.provide("c11n.allDomain.js.customize");

// Remove the Attachment Button in a New Task Tab
dojo.require("c11n.allDomain.js.widget.calendar.TaskDetail");
```

5. Create the JavaScript file that removes the Attachment Button in a New Task Tab.

c11n/allDomain/js/widget/calendar/TaskDetail.js

```
dojo.provide("c11n.allDomain.js.widget.calendar.TaskDetail");
dojo["require"]("iwc.widget.calendar.TaskDetail");
dojo.declare("iwc.widget.calendar.TaskDetail",
iwc.widget.calendar.TaskDetail, {

        postCreate: function () {
                this.inherited(arguments);

                dojo.style(this.attachButton.domNode, "display",
"none");

        },

        last: ""

});
```

6. Restart GlassFish Server and clear the browser cache to see the change.

Chapter 41. Removing the Language Selection Pull-Down Menu and Removing Languages from the Options Page

Customization Example - Removing the Language Selection Pull-Down Menu and Removing Languages from the Options Page

This Convergence example assumes that the reader has thoroughly read the Convergence Customization Guide.

To remove the language selection pull-down menu and to remove languages from the options page in Convergence 1.x, see Customization Example - Removing the Language Selection Pull-Down Menu and Removing Languages from the Options Page in Convergence 1.x.

Comms Community Verified Contribution
This article came from the Comms Community and has been verified by the Oracle Communications Unified Communications Suite product team.

The following examples describe how to customize language options in the Convergence UI:

- Removing the Language Selection Pull-Down Menu from the Options Page
- Removing Specific Languages from the Language Selection Pull-down Menu in the Options Page

Removing the Language Selection Pull-Down Menu from the Options Page

The following example removes the language selection pull-down menu from the Convergence Options Page. This is a useful example for sites which need to restrict the ability to modify specific account settings via the Convergence interface.

1. Enable client customizations:

```
./iwcadmin -o client.enablecustomization -v true
```

2. Create the following directory structure under <appserver domain root>/docroot/iwc_static/ to hold the customization files.

```
c11n/
c11n/allDomain
c11n/allDomain/js
c11n/allDomain/js/service
```

3. Create - if not already done - the general customization configuration file.
   This configuration file enables JavaScript customization (`jsEnabled: true`) across all domains (`module: "allDomain"`).

c11n/config.js

```
dojo.provide("c11n.config");

c11n.config = {

        // allDomain configuration
        allDomain: {
                module: "allDomain",           // module name
                themeEnabled: false,           // true if theme is
customized
                i18nEnabled: false,            // true if i18n is
customized
                jsEnabled: true                // true if js is
customized

                // the last entry must not end with comma

        }
}
```

4. Create the domain specific customization file (if it doesn't already exist).

c11n/allDomain/js/customize.js

```
dojo.provide("c11n.allDomain.js.customize");
dojo.require("c11n.allDomain.js.widget.option.General");
```

5. Overwrite the global options JavaScript file `option.General.js` with the following code. This file is available at the following path: `c11n/allDomain/js/widget/`.

"c11n/allDomain/js/widget/option/General.js"

```
dojo.provide("c11n.allDomain.js.widget.option.General");

dojo["require"]("iwc.widget.option.General");

dojo.declare("iwc.widget.option.General",
       iwc.widget.option.General,
       {
               buildRendering: function() {
                       this.inherited(arguments);

                       var elems = dojo.query("h2",
this.form.domNode);
                       if (elems[0]) dojo.style(elems[0],
"display", "none");

                       elems = dojo.query(".FormField",
this.form.domNode);
                       if (elems[0]) dojo.style(elems[0],
"display", "none");

               },

               last: ""
       }
);
```

6. Restart GlassFish Server and clear the browser cache to see the change.

Removing Specific Languages from the Language Selection Pull-down Menu in the Options Page

The following example removes specific default languages from the language selection pull-down menu in the Options Page. This is a useful example for sites which need to restrict the ability to modify specific account settings via the Convergence interface.

1. Enable client customizations:

```
./iwcadmin -o client.enablecustomization -v true
```

2. Create - if not already done - the following directory structure under <appserver domain root>/docroot/iwc_static/ to hold the customization files.

```
c11n/
c11n/allDomain
c11n/allDomain/js
c11n/allDomain/js/service
```

3. Create - if not already done - the general customization configuration file.
   This configuration file enables JavaScript customization (`jsEnabled: true`) across all domains (`module: "allDomain"`).

c11n/config.js

```
dojo.provide("c11n.config");

c11n.config = {

        // allDomain configuration
        allDomain: {
                module: "allDomain",           // module name
                themeEnabled: false,           // true if theme is
customized
                i18nEnabled: false,            // true if i18n is
customized
                jsEnabled: true                // true if js is
customized

                // the last entry must not end with comma

        }
}
```

4. Create the domain specific customization file (if it doesn't already exist).

c11n/allDomain/js/customize.js

```
dojo.provide("c11n.allDomain.js.customize");
// Hide a few default languages
dojo.require("c11n.allDomain.js.widget.option.General");
```

5. Extend the JavaScript file `option/General.js` with the following code. This file must be available at the following location: `c11n/allDomain/js/widget/`.
   For example, let us assume you're a customer based in Europe and you only want the European languages to be selectable.

c11n/allDomain/js/widget/option/General.js

```
dojo.provide("c11n.allDomain.js.widget.option.General");

dojo["require"]("iwc.widget.option.General");

dojo.declare("iwc.widget.option.General",
       iwc.widget.option.General,
       {
               buildRendering: function() {
                       this.inherited(arguments);

                       // remove your languages here
                       this.language.removeOption(["ja", "ko",
"zh-CN", "zh-TW"]);

               },

               last: ""
       }
);
```

6. Restart GlassFish Server and clear the browser cache to see the change.

Chapter 42. Using the Convergence Customization Example

Using the Convergence Customization Example

> **Note**
> This example applies to Convergence 2.x and later.

> **Note**
> For Convergence 1.x, see Using the Convergence 1.x Customization Example.

The Convergence directory includes an example of customized code provided in the *Application Server base*`/docroot/iwc_static/c11n_sample` directory. This directory is not live; that is, Convergence does not use the javascript and css files in this directory when it loads files at runtime.

The `clln_sample` directory includes the following customization samples:

- Themes
- i18n samples
- A mail.CreateMessage widget
- helloConvergence service

This page covers the following topics:

Enabling Customization in the Convergence Server

Before you can use the customization example, the Convergence Server must be enabled for customization. Use the command-line utility, `iwcadmin`, to set the `client.enablecustomization` parameter to `true`. For example:

```
./iwcadmin -o client.enablecustomization -v true
```

Customization Example Directory Structure

The easiest way to create your own customization is to follow the samples provided in this directory. You can copy all the files and subdirectories from `/c11n_sample` directory to the live directory, *Application Server base*`/docroot/iwc_static/c11n`. Then restart the Convergence server so it can recognize and register the `/iwc_static/c11n` directory. After that, customization is ready to go.

Convergence c11n_sample directory

The following list shows the structure of the `/c11n_sample` directory:

```
c11n_sample/
    config.js                              (configuration file)
    allDomain/
         js/                               (contains widgets and custom
applications)
              service/                     (contains additional services)
              widget/                      (contains UI widgets)
              customize.js                 (general customization
JavaScript file for adding new services, widgets, and so on)
         layout/
         nls/                              (contains language-specific
files)
         themes/                           (contains themes)
              customize.js                 (customization JavaScript file
specific to adding and removing themes.  This file is different from
js/customize.js.)

    example_com/
         js/                               (contains example_com widgets
and custom applications)
              widget/                      (contains example_com UI
widgets)
              customize.js                 (example_com customization
JavaScript file for adding new services, widgets, and so on)
         nls/                              (example_com language-specific
files)
         themes/                           (contains example_com themes)
              customize.js                 (customization JavaScript file
specific to adding and removing themes.  This file is different from
js/customize.js.)
```

Customizing Different Domains

The sample customization directory contains two subdirectories:

- `allDomain`
- `example_com`

The `allDomain` directory contains customizations that affect all domains in a hosted-domain deployment. If the deployment is in a single domain, `allDomain` contains customizations that affect this single domain.

The `example_com` directory contains customizations specific to a domain named `example.com`. Note that the directory is named `example_com` instead of `example.com`. This is done for ease of programming. In your LDAP directory, all data that names or is related to the domain, `example.com`, should remain `example.com`. Do not change the dot '.' to an underscore ' _ ' in LDAP. The Convergence customization code converts the dot '.' to an underscore ' _ ' whenever the directory name is used.

At runtime, Convergence uses the following rules to load the customizations:

- The customizations for specific domains are loaded first.
- If there are no customizations for specific domains, customizations for all domains are loaded.
- The new customizations codes override the base client code.

For example, if a non example.com user logs in to Convergence, the customizations are applied from the allDomain directory. However, if an example.com user logs in to Convergence, the customizations are applied from the example_com directory.

Defining Which UI Components Are Customized

The config.js. configuration file defines the types of customization (such as theme, javascript codes, and i18n) that are enabled. The config.js file is the first one loaded from the customization directory.

The following example config.js file shows a configuration for allDomain, which has enabled the theme and javascript. It also includes a configuration for the domain example.com, which has only enabled the theme.

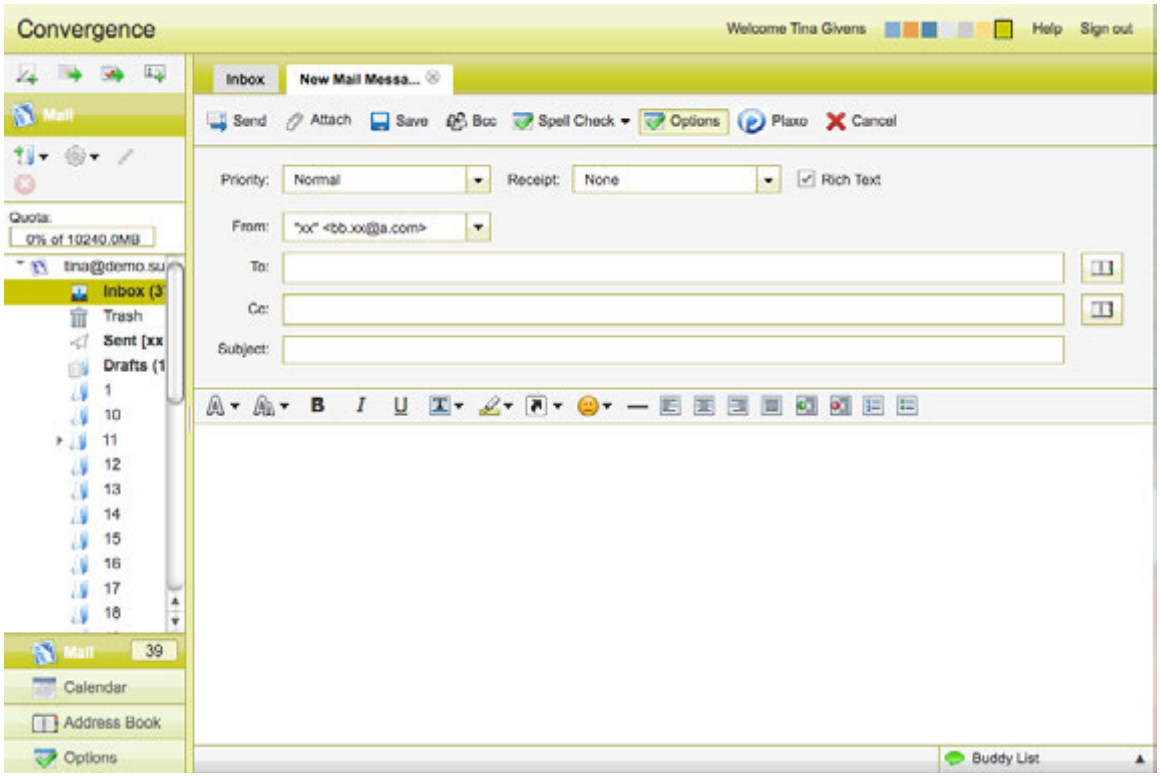
```
dojo.provide("c11n.config");

c11n.config={
         // allDomain configuration
         allDomain:{
            module: "allDomain",      //module name
            themeEnabled: true,       //true if theme is customized
            i18nEnabled: true,        //true if i18n is customized
            jsEnabled: true           //true if js is customized

                  //the last entry must not end with a comma
         }

         //replace example.com for each domain configuration, change
         //domain name example example.com to example_com for internal
programming
         example_com:{
            module: "example_com",    //module name
            themeEnabled: true,       //true if theme is customized
            i18nEnabled: false,       //true if i18n is customized
            jsEnabled: false          //true if js is customized

                  //the last entry must not end with a comma
         }
}
```

Note
Do not add a comma to the last entry of each configuration in the config.js file. Otherwise, your customization will not properly load, and you might see a c11n.config error. In the above example, there is no comma after the jsEnabled customization setting in either the allDomain or example_com configuration.

Custom Button for mail.CreateMessage Widget

The following figure shows an example of a custom button (Plaxo) for the mail.CreateMessage widget.

Figure 1: Compose Tab with a new Customized Button (Plaxo)



To Enable the `mail.CreateMessage` Widget

1. Change to the `iwc_static` directory and copy the `c11n_sample` directory to the `c11n` directory.

```
For example:

cd /opt/SUNWappserver/domains/domain1/docroot/iwc_static
cp -r c11n_sample c11n
# Run the ls command to confirm that the directory was created.
```

2. Edit the `c11n/config.js` file so that `i18nEnabled` and `jsEnabled` are both set to `true` in the `allDomain` section:

```
dojo.provide("c11n.config");
c11n.config = {

    // allDomain configuration
    allDomain: {
        module: "allDomain",        // module name
        themeEnabled: false,         // true if theme is customized
        i18nEnabled: true,        // true if i18n is customized
        jsEnabled: true            // true if js is customized

        // the last entry must not end with comma
    },
    ...
```

3. Uncomment the following section in the `/iwc_static/c11n/allDomain/js/customize.js` file.

```
var loadCustomizedCssFile = function(url, id) {
        if (!id){
                id = url.replace(/../gm, "").replace(///gm,
"_").replace(/./gm, "_");
        }
        var link = window.document.getElementById(id);
        if (! link) {
                link = window.document.createElement("link");
                link.setAttribute("id", id);
                link.setAttribute("type", "text/css");
                link.setAttribute("rel", "stylesheet");
                var head =
window.document.getElementsByTagName("head")[0];
                head.appendChild(link);
        }

        link.setAttribute("href", url + "?" + djConfig.cacheBust);
}
loadCustomizedCssFile("../c11n/allDomain/layout/css/c11n.css")
dojo.require("c11n.allDomain.js.widget.mail.CreateMessage");
```

4. Clear browser cache and open new Compose tab to view modifications.

Chapter 43. Removing the SMS option from the Calendar Notification Options and the New Event Reminder Dialog

Customization Example - Removing the SMS option from the Calendar Notification Options and the New Event Reminder Dialog

Note
This Convergence example assumes that you have read the Convergence Customization Guide.
This sample can also be found in the `c11n_sample` directory.

Note
To remove the SMS Option in Convergence 1.x, see Removing the SMS Option of the Calendar Notifications and the New Event reminder section in Convergence 1.x.

The following example removes the SMS Option in the Calendar Notifications Options tab and from the New Event Reminder dialog.

- Enable client customizations.

```
./iwcadmin -o client.enablecustomization -v true
```

- Create the following directory structure under the *appserver-domain-root* `/docroot/iwc_static/` directory to hold the customization files.

```
c11n/
c11n/allDomain
c11n/allDomain/js
c11n/allDomain/js/widget
```

- Generate the general customization configuration file.
  This configuration file enables JavaScript customization (`jsEnabled: true`) across all domains (`module: "allDomain"`).

c11n/config.js

```
dojo.provide("c11n.config");

c11n.config = {

        // allDomain configuration
        allDomain: {
                module: "allDomain",           // module name
                themeEnabled: false,           // true if theme is
customized
                i18nEnabled: false,            // true if i18n is
customized
                jsEnabled: true                // true if js is
customized

                // the last entry must not end with comma

        }
}
```

- Create the domain specific customization file (if it doesn't already exist).

c11n/allDomain/js/customize.js

```
dojo.provide("c11n.allDomain.js.customize");

// Remove the SMS Option from the possible Calendar Notifications
dojo.require("c11n.allDomain.js.widget.calendar.option.Notification")
```

- Create the JavaScript file that removes the SMS Options.

c11n/allDomain/js/widget/calendar/option/Notification.js

```
dojo.provide("c11n.allDomain.js.widget.calendar.option.Notification")
iwc.widget.calendar.option.Notification, {

        buildRendering: function() {
                this.inherited(arguments);

                dojo.style(this.notificationSMS.domNode.parentNode,
"display", "none");

        },

        last: ""

});
```

c11n/allDomain/js/widget/calendar/NotificationDialog.js

```
dojo.provide("c11n.allDomain.js.widget.calendar.NotificationDialog");
iwc.widget.calendar.NotificationDialog, {

            buildRendering: function() {
                this.inherited(arguments);

                this.methodSelector.removeOption("sms");
                this.methodSelectorAbs.removeOption("sms");
            },

            last: ""

});
```

- Restart GlassFish Server and clear the browser cache to see the change.

Chapter 44. Removing the Move Button in the Mail Open Folder

Customization Example - Removing the Move Button in the Mail Open Folder

This Convergence example assumes that the reader has thoroughly read the Convergence Customization Guide.

Note
To remove the Move Button in Mail Open Folder in Convergence 1.x, see Removing the Move Button in Mail Open Folder in Convergence 1.x.

Comms Community Verified Contribution
This article came from the Comms Community and has been verified by the Oracle Communications Unified Communications Suite product team.

The following example deletes the `Move` Button in Account Setting in the Mail Open Folder for users in all domains.

1. Enable client customizations

```
./iwcadmin -o client.enablecustomization -v true
```

2. Create the following directory structure under `<appserver domain root>/docroot/iwc_static/` to hold the customization files.

```
c11n/
c11n/allDomain
c11n/allDomain/js
c11n/allDomain/js/widget
```

3. Generate the general customization configuration file.
   This configuration file enables JavaScript customization (`jsEnabled: true`) across all domains (`module: "allDomain"`).

c11n/config.js

```
dojo.provide("c11n.config");

c11n.config = {

        // allDomain configuration
        allDomain: {
                module: "allDomain",           // module name
                themeEnabled: false,           // true if theme is
customized
                i18nEnabled: false,            // true if i18n is
customized
                jsEnabled: true                // true if js is
customized

                // the last entry must not end with comma

        }
}
```

4. Create the domain specific customization file (if it doesn't already exist).

c11n/allDomain/js/customize.js

```
dojo.provide("c11n.allDomain.js.customize");

// Delete Move Button in Mail Option Folder
dojo.require("c11n.allDomain.js.widget.mail.OpenFolder");
```

5. Create the JavaScript file which deletes the `Move` button.

c11n/allDomain/js/widget/mail/OpenFolder.js

```
dojo.provide("c11n.allDomain.js.widget.mail.OpenFolder");

dojo["require"]("iwc.widget.mail.OpenFolder");

dojo.declare("iwc.widget.mail.OpenFolder",
iwc.widget.mail.OpenFolder, {
        buildRendering: function() {
                // invoke the original buildRendering()
                this.inherited(arguments);

                dojo.addClass(this.moveButton.domNode,
"dijitHidden");       // remove the button
                dojo.addClass(this.moveMenuItem.domNode,
"dijitHidden");     // remove the menu item

        },

        last: ""

});
```

6. Restart GlassFish Server and clear the browser cache to see the change.

Chapter 45. Customization Example - Handling Large Logos in Gradient Themes

Customization Example - Handling Large Logos in Gradient Themes

Topics:

When adding a logo to a theme, the logo's height typically does not exceed 40px in height. If you choose to add a logo that is larger than 40px, the banner might not display properly in a gradient image theme. Themes with gradient images include:

- theme_yellow
- theme_dark_blue
- theme_light_blue
- theme_green
- theme_grey

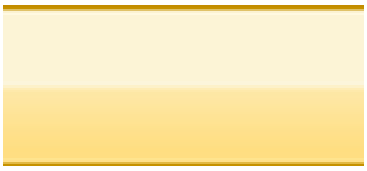
Themes that use solid color (for example, theme_blue, theme_orange) do not need to be modified for larger logos since the color will fill the entire height of the masthead. However, themes with gradient images require adding a new image to fill the entire height of the masthead when a larger logo is inserted. The theme needs to be customized to use the larger background image for the masthead.

The following example shows an 80px logo in a 40px gradient image theme. Product name, Welcome *Convergence Display Name*, theme buttons, help, and sign out text are stuck in the middle of the banner, making that text difficult to read:

Re-sized Gradient Banner Samples

Use the following banner samples when manipulating large logos (>40px) in gradient themes. To copy the actual resizable gradient banners, refer to the following file: Convergence Customization Resizable Gradient Banners:

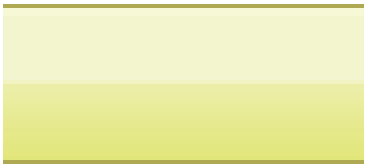
- Yellow (yellow_masthead.png):



- Dark Blue (masthead.png):

- Light Blue (light_blue_masthead.png):

- Green (green_masthead.png):



- Grey (grey_masthead.png):

Requirements for Customizing Large Logos in Gradient Themes

Prior to customizing large logos in gradient themes:

- Review Customizing Themes and Banner in Convergence 2.x.
- Enable theme customization as described in Before You Begin.
- Know the height of your new logo. The default logo in the banner is 40 pixels in height.

Customizing the Dark Blue Theme

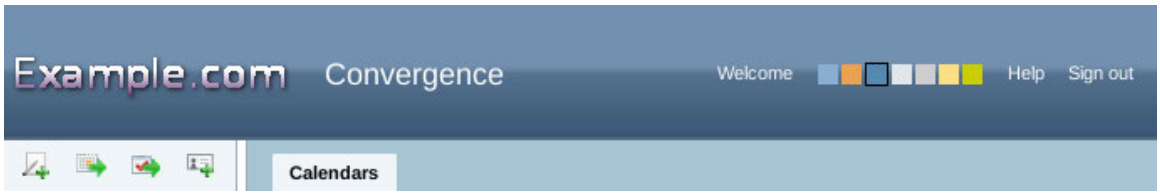
To handle large logos in gradient themes, you need to create a new image to be used for the `mastheadBackground` which fits the desired height for each affected theme. This method ensures a properly fitted image for the larger icon. To add the resized background image to the gradient theme:

1. Copy the following image (masthead.png) into the `c11n/allDomain/themes/dark_blue/images/` directory.

2. Follow the steps in theme customization for modifying a specific theme.
3. In step 6 of modifying a specific theme, set the `mastheadBackground` property to use the new image in the dark blue theme configuration (`theme.json`) file in the `c11n/allDomain/themes/dark_blue/` directory:

```
mastheadBackground: "#7291B0
url('../c11n/allDomain/themes/dark_blue/images/masthead.png')
repeat-x center center"
```

4. Refresh your web browser.



5. If necessary, refine the image so to create the right look and feel.

Chapter 46. custom-useroptions.properties Mapping File

`custom-useroptions.properties` Mapping File

Topics:

Overview of `custom-useroptions.properties` Mapping File

With the exception of system-generated attributes, `custom-useroptions.properties` defines mappings between LDAP attributes which can be retrieved through the `get_allprefs.iwc` command from your LDAP directory for any customization purpose. It allows site administrators to map custom LDAP attributes. For example, you can obtain employee numbers from your LDAP directory through the `get_allprefs.iwc` command, which can be used to customize the UI with the `custom-useroptions.properties` file.

File location: `/var/opt/sun/comms/iwc/config/custom-useroptions.properties`
Used with: `iwc.protocol.iwcp.setUserPrefs` to set custom attributes

Structure of `custom-useroptions.properties` Mapping File

The following file syntax describes how to map custom LDAP attributes in `custom-useroptions.properties`. `<mapping-name>` is the parameter name to be passed to `setUserprefs`. `<ldap-attribute>` is the LDAP attribute to be mapped.

custom-useroptions.properties

```
# This property file is a mapping of request parameter name for a custom
user
# preference to its corresponding LDAP attribute name.
# Format:
# <mapping-name>=<ldap-attribute>
# Where:
# - <mapping-name> must start with "custom." (without quotes).
# ex: custom.mysmsnumber=<ldap-attribute>
# - <ldap-attribute> is a name of the ldap attribute in user's LDAP
entry.
#   + To use a single valued attribute, just provide the name of the
attribute.
#   ex: custom.mysmsnumber=smsNumber
#   + To use a multi valued attribute with sub-attributes, delimit the
ldap
#     attribute with a ":" followed by the sub-attribute name.
#     ex: custom.mysmsnumber=contactMode:sms (for multi valued ldap
attribute
#     contactMode with values as say - contactMode: sms=<value> etc
#   + To use a multi valued attribute, delimit the ldap attribute with a
":"
#     followed by a *.
#     ex: custom.mysmsnumber=smsNumbers:*
# - In all the above cases, the xml response to the client would have
the
# mapping name (excluding "custom.") as the element name and the user
preferenc
# as the value of a <value> element (child of mapping element).
```

Example of **`custom-useroptions.properties`**

```
custom.name=cn
```

To use a custom user option in a Convergence customization, set the option, as shown in the following example:

```
iwc.userPrefs.custom.name
```

To update a custom user option in a Convergence customization, use `setUserPrefs`, as shown in the following example:

```
/* where data is a array of objects. For example: */
var data = [];
data.push({name:<param name>,value:<param value>});

var deferred = iwc.protocol.iwcp.setUserPrefs(data, true /* always sync
*/);
deferred.addCallback(this, function() {
   //success
});
```

You must set the user option before you can call it using `setUserPrefs`.

Chapter 47. Customization Example - Redirecting User to Another Page to Change Password

Customization Example - Redirecting User to Another Page to Change Password

This customization example describes how you might redirect a user to another page to change the login password for Convergence.

1. Enable client customizations.

```
./iwcadmin -o client.enablecustomization -v true
```

2. Create the following directory structure under the *appserver domain root* `/docroot/iwc_static/` directory to hold the customization files.

```
c11n/
c11n/allDomain
c11n/allDomain/js
c11n/allDomain/js/option
```

3. Generate the general customization configuration file.
   This configuration file enables JavaScript customization (`jsEnabled: true`) across all domains (`module: "allDomain"`).

c11n/config.js

```
dojo.provide("c11n.config");

c11n.config = {

        // allDomain configuration
        allDomain: {
               module: "allDomain",           // module name
               themeEnabled: false,           // true if theme is
customized
               i18nEnabled: false,            // true if i18n is
customized
               jsEnabled: true                // true if js is
customized

               // the last entry must not end with comma

        }
}
```

4. Create the domain specific customization file (if it does not already exist).

c11n/allDomain/js/customize.js

```
dojo.provide("c11n.allDomain.js.customize");

dojo.require("c11n.allDomain.js.widget.option.Password");
```

5. Create a JavaScript file that redirects the change password option to www.example.com.

c11n/allDomain/js/widget/option/Password.js

```
dojo.provide("c11n.allDomain.js.widget.option.Password");
dojo["require"]("iwc.widget.option.Password");
dojo["require"]("iwc.widget.form.BorderContainerForm");
dojo["require"]("iwc.widget.option.ViewerTabPane");

dojo.declare("iwc.widget.option.Password",
[iwc.widget.option.ViewerTabPane], {
        templateString: '<div><form
dojoType="iwc.widget.form.BorderContainerForm"
dojoAttachPoint="form"><iframe src="http://www.example.com"
width=100% height=1000px></iframe></form></div>',

        postCreate: function() {
                dojo.addClass(this.form.buttonNode.domNode,
"dijitHidden");
        },

        last: ""

});
```

6. Restart GlassFish Server and clear the browser cache to see the change.

Chapter 48. Changing From Address to Only Include Email Address

Customization Example - Changing From: Address to Only Include Email Address (not cn)

This customization example describes how to change the From: address to only include the email address (not cn):

1. Enable client customizations.

```
./iwcadmin -o client.enablecustomization -v true
```

2. Create the following directory structure under the appserver domain root/docroot/iwc_static/ directory to hold the customization files.

```
c11n/
c11n/allDomain
c11n/allDomain/js
c11n/allDomain/js/mail
```

3. Generate the general customization configuration file.
   This configuration file enables JavaScript customization (`jsEnabled: true`) across all domains (module: `"allDomain"`).

c11n/config.js

```
dojo.provide("c11n.config");

c11n.config = {

        // allDomain configuration
        allDomain: {
               module: "allDomain",           // module name
               themeEnabled: false,           // true if theme is
customized
               i18nEnabled: false,            // true if i18n is
customized
               jsEnabled: true                // true if js is
customized

               // the last entry must not end with comma

        }
}
```

4. Create the domain specific customization file (if it does not already exist).

c11n/allDomain/js/customize.js

```
dojo.provide("c11n.allDomain.js.customize");

dojo.require("c11n.allDomain.js.widget.mail.CreateMessage");
```

5. Create a JavaScript file that changes the From: address to only include the email address.

c11n/allDomain/js/widget/mail/CreateMessage.js

```
dojo.provide("c11n.allDomain.js.widget.mail.CreateMessage");
dojo["require"]("iwc.widget.mail.CreateMessage");
dojo.declare("iwc.widget.mail.CreateMessage",
iwc.widget.mail.CreateMessage, {

	postCreate: function() {
		// remove the sender id displayname

dojo.forEach(iwc.userPrefs.senderidentities.identity, function(id)
{
			id.displayname = "";
		});

		this.inherited(arguments);
	},

	last: ""

});
```

6. Restart Glassfish Server and clear the browser cache to see the change.

Chapter 49. Customization Example - Displaying a Password Policy in the Convergence UI

Customization Example - Displaying a Password Policy in the Convergence UI

When changing the Convergence UI password, a user might need information on your site's password policy, the rules designed to enhance security by employing strong passwords. This can be achieved through a customization:

1. Enable client customizations.

```
./iwcadmin -o client.enablecustomization -v true
```

2. Create the following directory structure under the *appserver domain root* `/docroot/iwc_static/` directory to hold the customization files.

```
c11n/
c11n/allDomain
c11n/allDomain/js
c11n/allDomain/js/option
```

3. Generate the general customization configuration file.
This configuration file enables JavaScript customization (`jsEnabled: true`) across all domains (`module: "allDomain"`).

c11n/config.js

```
dojo.provide("c11n.config");

c11n.config = {

        // allDomain configuration
        allDomain: {
                module: "allDomain",           // module name
                themeEnabled: false,           // true if theme is
customized
                i18nEnabled: false,            // true if i18n is
customized
                jsEnabled: true                // true if js is
customized

                // the last entry must not end with comma

        }
}
```

4. Create the domain specific customization file (if it does not already exist).

c11n/allDomain/js/customize.js

```
dojo.provide("c11n.allDomain.js.customize");

dojo.require("c11n.allDomain.js.widget.option.Password");
```

5. Create the JavaScript file allows you to add password policy information on the Change Password page.

c11n/allDomain/js/widget/option/Password.js

```
dojo.provide("c11n.allDomain.js.widget.option.Password");
dojo["require"]("iwc.widget.option.Password");
dojo.declare("iwc.widget.option.Password",
        iwc.widget.option.Password,
        {
                postCreate: function() {
                        this.inherited(arguments);
                        dojo.place("<h2>Password
Policy:</h2><ul><li>Password must at least be 8
characters</li><li>Password must not be the same as the previous
passwords</li></ul>", this.form.containerNode, "last");

                },

                last: ""
        }
);
```

6. Restart GlassFish Server and clear the browser cache to see the change.

Chapter 50. Adding and Modifying Calendar 7 Timezones

Customization Example - Adding and Modifying Calendar 7 Timezones

> **Note**
> To administer timezones in Calendar Server 7, see Administering Timezones.

The following example describes how to add and modify calendar timezones with Calendar Server 7 and Convergence with the following tasks:

- Adding a new calendar timezone for Saskatchewan in Canada
- Modifying the existing "America Los Angeles" timezone to display as the more generic "America Pacific Time"

1. Enable Convergence client customizations.

   ```
   ./iwcadmin -o client.enablecustomization -v true
   ```

2. Create the following directory structure under the *appserver-domain-root* `/docroot/iwc_static/` to hold the required customization files.

   ```
   c11n/
   c11n/allDomain
   c11n/allDomain/nls
   ```

3. Generate the general customization configuration file.
   This configuration file enables modifications and additions to the list of translation strings (`i18nEnabled: true`). This allows for calendar server TZID string to City Name translations. The changes apply to all user domains (`module: "allDomain"`).

c11n/config.js

```
dojo.provide("c11n.config");

c11n.config = {

        // allDomain configuration
        allDomain: {
                module: "allDomain",           // module name
                themeEnabled: false,           // true if theme is
customized
                i18nEnabled: true,            // true if i18n is
customized
                jsEnabled: false                // true if js is
customized

                // the last entry must not end with comma

        }
}
```

4. For instructions on adding and modifying new WCAP timezones in Calendar Server 7, see Adding New WCAP Timezones and Timezone Aliases in Calendar Server 7.
5. Create the `resources.js` file which maps and displays TZID to city name. For example, `Saskatchewan` is the value displayed in the UI for the `Saskatchewan` TZID. Similarly, `Pacific Time` displays in the UI when the TZID is `Los_Angeles`:

c11n/allDomain/nls/resources.js

```
options_city_Saskatchewan: "Saskatchewan",
options_city_Los_Angeles: "Pacific Time",
last: ""
}
```

Note
TZIDs defined in the `timezones.ics` file that have no corresponding `options_city_*` translation string are ignored and not displayed as an option. Similarly, if you want to change the name of a TZID (for example Los Angeles to Pacific Time), and you don't specify the change in `resources.js`, the original name (in this case, Los Angeles) displays in the UI.

6. Restart GlassFish Server and clear the browser cache to see the change.

Chapter 51. Changing the Mail Forward Default from As Attachment to Inline

Changing the Mail Forward Default from As Attachment to Inline

When you click the Forward button to forward an email message, a two options display: As Attachment and Inline. If you do not choose either of these options, the default, As Attachment, is selected. This Convergence customization example describes how to change the mail forward default from As Attachment to Inline:

1. Enable client customizations.

```
./iwcadmin -o client.enablecustomization -v true
```

2. Create the following directory structure under the appserver domain root/docroot/iwc_static/ directory to hold the customization files.

```
c11n/
c11n/allDomain
c11n/allDomain/js
c11n/allDomain/js/widget
c11n/allDomain/js/widget/mail
```

3. Generate the general customization configuration file.
   This configuration file enables JavaScript customization and i18n customization (since labels are also updated in `resources.js`) across all domains (module: "`allDomain`").

c11n/config.js

```
dojo.provide("c11n.config");

c11n.config = {

        // allDomain configuration
        allDomain: {
               module: "allDomain",          // module name
               themeEnabled: false,          // true if theme is
customized
               i18nEnabled: true,            // true if i18n is
customized
               jsEnabled: true               // true if js is
customized

               // the last entry must not end with comma

        }
}
```

4. Create the domain specific customization file (if it does not already exist).

c11n/allDomain/js/customize.js

```
dojo.provide("c11n.allDomain.js.customize");

dojo.require("c11n.allDomain.js.widget.mail.OpenFolder");
dojo.require("c11n.allDomain.js.widget.mail.OpenMessage");
```

5. Create a JavaScript file that changes the mail forward default from As Attachment to Inline in the mail.OpenFolder widget.

c11n/allDomain/js/widget/mail/OpenFolder.js

```
dojo.provide("c11n.allDomain.js.widget.mail.OpenFolder");

dojo["require"]("iwc.widget.mail.OpenFolder");

dojo.declare("iwc.widget.mail.OpenFolder",
iwc.widget.mail.OpenFolder, {
        postCreate: function() {
                this.inherited(arguments);

                this._origOnForwardMessageInline =
this.onForwardMessageInline;
                this._origOnForwardMessageAttach =
this.onForwardMessageAttach;

                this.onForwardMessageInline = dojo.hitch(this,
function() {

this._origOnForwardMessageAttach(arguments);
                });

                this.onForwardMessageAttach = dojo.hitch(this,
function() {

this._origOnForwardMessageInline(arguments);
                });
        },


        last: ""

});
```

6. Create a JavaScript file that changes the mail forward default from As Attachment to Inline in the mail.OpenMessage widget.

c11n/allDomain/js/widget/mail/OpenMessage.js

```
dojo.provide("c11n.allDomain.js.widget.mail.OpenMessage");

dojo["require"]("iwc.widget.mail.OpenMessage");

dojo.declare("iwc.widget.mail.OpenMessage",
iwc.widget.mail.OpenMessage, {
        postCreate: function() {
                this.inherited(arguments);

                this._origForwardMessageInline =
this.forwardMessageInline;
                this._origForwardMessageAttach =
this.forwardMessageAttach;

                this.forwardMessageInline = dojo.hitch(this,
function() {
                        this._origForwardMessageAttach(arguments);
                });

                this.forwardMessageAttach = dojo.hitch(this,
function() {
                        this._origForwardMessageInline(arguments);
                });
        },


        last: ""

});
```

7. Create an `nls` directory if it doesn't already exist:

```
iwc_static/c11n/allDomain/nls
```

8. Create a default resource file named `resources.js` to create new labels in the UI.

```
iwc_static/c11n/allDomain/nls/resources.js
```

9. In the `resources.js` file, add the following text strings:

iwc_static/c11n/allDomain/nls/resources.js

```
{
    forward_attach_item: "Inline",
    forward_inline_item: "As Attachment",
}
```

10. Restart GlassFish Server and clear the browser cache to see the change.