

Oracle® Communications Convergence Customization Guide



Release 3.0.3
F99830-02
December 2024

ORACLE®

Copyright © 2008, 2024, Oracle and/or its affiliates.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software," "commercial computer software documentation," or "limited rights data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle®, Java, MySQL, and NetSuite are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

Preface

Audience	viii
Documentation Accessibility	viii
Diversity and Inclusion	viii

1 Convergence Customization Concepts

Customization Overview	1-1
Directory Placeholders Used in This Guide	1-1
Key Features of Customization	1-2
Skills Required for Customizing Convergence	1-3
About Convergence Customization Workflow	1-3
Enabling the Customization Framework	1-3
Customizing Convergence	1-4
Restarting the Application Server	1-4
Possible Problems	1-5
About the Customization Examples	1-5
Enabling Customization in the Convergence Server	1-5
Customization Directory Structure	1-5
Creating the Customization Directory	1-6
Convergence c11n_sample directory	1-6
Customizing Different Domains	1-6
Defining Which UI Components Are Customized	1-7
To Enable the mail.CreateMessage Widget	1-8
Technical Overview	1-8
About How Topics Apply to My Customizations	1-9
About Convergence Architecture Customization Support	1-9
Convergence Customization Booting	1-9
Customization Loading Order at Run Time	1-9
Loading Order for Multiple Domains	1-10
Loading Order Across All Domains	1-10
Directory Layout	1-10
Dojo Basics	1-11
Using the Debugging Directory to Customize Convergence	1-11

About Dojo Statements Map to the Convergence Directory Structure	1-12
Preserving Custom Widgets During Upgrades	1-12
Preserving Themes During Upgrades	1-13
Preserving Dojo Widgets During Upgrades	1-13
Consolidating Convergence Customizations to Preserve Client Performance	1-13
About the custom-useroptions.properties Mapping File	1-14
Structure of custom-useroptions.properties Mapping File	1-14

2 Enabling and Disabling Customization

About Enabling Customization for the Deployment, Domains, and Users	2-1
Enabling Customization for the Convergence Deployment	2-2
Enabling or Disabling Customization for an Individual User	2-2
Enabling or Disabling Customization for an Individual Domain	2-2

3 About Convergence UI Widgets

Location of Javascript Widgets	3-1
Common Widgets	3-1
Mail Widgets	3-3
Address Book Widgets	3-6
Calendar Widgets	3-11
Options Widgets	3-26

4 Working with the Convergence UI

Customization Requirements	4-1
Theme Customization Features	4-1
Default Themes Included with Convergence	4-2
About the Basic Theme	4-4
Basic Theme Properties	4-5
JSON Reference for Customizing Themes	4-7
Example Theme.json File	4-8
Customizing Layout HTML Pages	4-11
Creating and Customizing login.html	4-11
Creating and Customizing login.html in a Hosted Domain	4-12
Modifying the Login Page Welcome Message	4-12
Creating and Customizing main.html	4-13
Configuring the Per-Domain Main Page	4-13
Creating and Customizing calendar.html	4-14
Setting a Theme in an Anonymous Calendar	4-14
Customizing Anonymous Calendar Date and Time Format	4-15

Integrating Third-Party Applications	4-16
Integrating HelloConvergence into Convergence	4-17
About Adding a New Language	4-19
Adding a New Language in Convergence	4-19
Adding a New Language that Does Not Currently Exist in the Dojo Toolkit	4-20
Sample Custom I10n Resource File	4-20
Adding a Label for the New Language to the Global Options Language Menu	4-21
Adding a Label for the New Language to the Convergence Login Page	4-22
Setting Help for Unsupported Locales in the Convergence Banner	4-23

5 Convergence UI Customization Examples

Customization Requirements	5-1
Modifying a Specific Theme	5-1
Hiding a Single Theme	5-2
Creating a New Theme	5-2
Making a Newly Created Theme the Default	5-3
Adding a Logo to All Themes	5-4
Adding a Logo to the Right Side of the Banner	5-4
Making the Banner Logo a Clickable Link	5-6
Handling Large Logos in Gradient Themes	5-8
Re-Sized Gradient Banner Samples	5-8
Customizing the Dark Blue Theme	5-9
Adding and Removing Fonts from the Editor Menu	5-9
Changing an Icon in the Service Selector	5-11
Displaying and Printing the Japanese Yen Symbol	5-13
Modifying the Document Title and the Convergence Text in the Banner	5-17
Changing Names and Labels in the Convergence UI	5-19
Removing or Changing the Product Name on the Mail HTML Page	5-20
Displaying a Password Policy in the Convergence UI	5-21
Hiding the Quick Actions Menu	5-22

6 Convergence Messaging Customization Examples

Customization Requirements	6-1
Changing the Mail Forward Default from As Attachment to Inline	6-1
Changing Default Folder Mappings for Sent and Deleted Messages	6-3
Changing From: Address to Only Include Email Address	6-4
Changing or Removing the Signature Separator	6-5
Modifying Mail Folder Icons in the Service Navigator	6-6
Removing Folder Sharing and Subscribing Menu Options	6-9
Removing the Local Account Mail Forwarding Option	6-11

Removing the Move Button in the Mail Open Folder	6-12
Removing the Reply-To Address Option	6-13
Restricting Outgoing Mail with Local Account Identity Parameters	6-14
Hiding User-Created Folder Names	6-15
Adding Additional Spell Checker Dictionaries	6-16
Customizing the Attachment Blacklist and Whitelist	6-18
Increasing Corporate Address Book Entries in Email Autocomplete	6-20

7 Convergence Calendar Customization Examples

Customization Requirements	7-1
Displaying a Complete Title in Calendar List Views	7-1
Adding or Modifying Calendar Time Zones	7-2
Categorizing Calendar Events with Text or Background Colors	7-4
Disabling Event Balloon User Input Saving as Event Description	7-6
Disabling Quick Parsing Calendar Capabilities	7-7
Removing the Attachment Button in the New Task Tab	7-8
Removing Reservations from the New Event Tab	7-9
Disabling Calendar Event Notification by SMS	7-10

8 Convergence Address Book Customization Examples

Customization Requirements	8-1
Changing the Corporate Directory Name	8-1
Displaying Additional Address Book Attributes When Adding Contacts to an Invitation	8-2
Displaying Additional Address Book Attributes When Adding Resources to an Invitation	8-5
Removing the Copy To Button	8-5
Removing the Google Maps Link	8-6
Importing or Exporting Address Book Information in a Custom Language	8-7

9 Convergence Options Customization Examples

Customization Requirements	9-1
Disabling External POP Account Access	9-1
Enabling or Disabling the Modification of Identity Settings	9-2
Redirecting Users to Another Page to Change Password	9-4
Removing Change Password, Vacation Message, and Calendar Notification Options	9-5
Removing Default Language List in General Options	9-6
Removing Languages from Language List in General Options	9-7
Customizing the Default Alert Sounds	9-8

Preface

This guide explains how to customize the look and feel of Convergence. Although the product architecture permits almost unlimited customization, this guide focuses on concepts, reference, examples, and explanations on how to perform the most commonly used customizations.

Audience

This document is intended for Convergence system administrators, customizers, and developers. This guide assumes that you have a working knowledge of the following concepts:

- Oracle WebLogic Server administration
- System administration and networking

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc>.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info> or visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs> if you are hearing impaired.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

1

Convergence Customization Concepts

This guide describes how to customize the look and feel of Convergence. Although the product architecture permits an almost unlimited customization, this guide focuses on concepts, reference, examples, and explanations on how to perform the most commonly used customizations.

Customization Overview

You can customize the Convergence UI, giving you the power to flexibly change the look and feel of the UI, control the services available, manage the display of advertisements, and change the UI in various other ways. The Convergence UI is built specifically to support customization in the following ways:

- Change the look and feel of the UI (implemented by customizing themes).
- Brand the product as you'd like, altering the banner, text, icons, and so on.
- Customize each domain in its own way.
- Manage the display and location of advertisements.
- Control which back-end services are made available to end users.
- Localize the UI for languages in addition to those supported automatically by the product.
- Integrate third-party applications, in addition to the standard services such as mail and calendar.

Note:

Convergence 3.0.2 and above supports the traditional two-pane layout and a new three-pane layout. All the customization pre-existing on a two-pane layout may not be rendered in the three-pane layout.

The three-pane layout is the default layout in Convergence 3.0.2.0.0 and above. You can change the layout using the following **iwadmin** command and restart the Oracle WebLogic Server:

```
iwadmin -o user.common.layoutPreference -v classic
```

Directory Placeholders Used in This Guide

[Table 1-1](#) lists the directory placeholders used in this guide:

Table 1-1 Convergence Directory Placeholders

Placeholder	Description
<i>Convergence_Home</i>	Specifies the installation location for the Convergence software. The default is /opt/sun/comms/iwc .
<i>WebLogic_Home</i>	The directory in which Oracle WebLogic Server software is installed. For example: WLS_HOME.
<i>Convergence_Domain</i>	The application server directory containing the configuration files for the domain in which Convergence is deployed. Convergence_Domain is created in <i>WLS_HOME/Oracle_Home/user_projects/domains</i> . Convergence_Domain for Oracle WebLogic Server Deployment is: <ul style="list-style-type: none"> <i>WLS_HOME/Oracle_Home/user_projects/domains/base_domain</i>
<i>data_directory</i>	Specifies the location in which all data elements are stored: /var/opt/sun/comms/iwc
<i>c11n_Home</i>	The directory in which all Convergence customization files and directories are created for WebLogic Server. <ul style="list-style-type: none"> The <i>c11n_Home</i> directory for Oracle WebLogic Server is: data_directory/web-src/client/iwc_static/c11n

Key Features of Customization

The following list briefly describes the main customization features provided by Convergence.

Theme

You can customize the look and feel of the UI by adding new themes. Each theme constitutes a set of icons, colors, fonts, and so on, which can be made available to end users through the banner of the UI. End users can then choose from a selection of site-defined themes.

Branding and Banner

You can change the brand and banner of the UI. In a hosted-domain environment, different banners and brands can be developed for each domain.

Tailoring Customizations for Individual Domains and End Users

You can control the *level* of customization. Most customization features can be targeted for

- The entire site
- A particular domain in a hosted-domain environment
- A particular user or set of users, according to the way they are provisioned

For example, the theme, banners and other widgets, availability of services, and language can be customized for all three levels: site-wide, domain-specific, and individual end-users.

Controlling Which Services Are Available to End Users

You can enable and disable the services available to end users. For example, if you do not want users to use the calendar service, you can disable it. Services include mail, address book and calendar; and so on.

Further, you can control the subset of services available to the whole site, to an individual domain, or to particular end users. See *Convergence System Administrator's Guide* for more information.

Customizing Locales

You can make languages available to end users, in addition to those provided by Convergence, by adding your own language resources to the source code.

Integrating Advertisements Within Convergence

Convergence provides predefined advertisement space (which may include banner, text, images, and so on), which you can customize. You can specify the set of events (such as refreshing a page or reading a message) that trigger a call to the advertising API. Ads can be located in the top, bottom, left, and right panels of the UI, or at the top, bottom, left, and right of the email message viewer.

Integrating Third-Party Applications into Convergence

You can enable new services, in addition to the services provided by Convergence (mail, calendar and address book). A new service can be made available in the UI and integrated with the existing services. For example, you could add a tab in the UI and integrate an end user mailing list management application within Convergence.

Skills Required for Customizing Convergence

Customization in Convergence requires different sets of skills, depending on which aspects of the UI you intend to customize. For theme customization, you should have expertise in CSS styling and graphic design.

For Javascript customization, you should have expertise in Javascript, dojo, and dijit. In addition, you will need to become familiar with the Convergence architecture and code.

For information about the Convergence architecture, directory structure, and the mapping of dojo declarations, see "[Technical Overview](#)" for more information.

About Convergence Customization Workflow

Customizing Convergence for the first time consists of the following general steps:

1. Enabling the customization framework in Convergence.
See "[Enabling the Customization Framework](#)" for more information.
 2. Creating one or more customizations.
See "[Customizing Convergence](#)" for more information.
 3. Restarting the Oracle WebLogic Server.
See "[Restarting the Application Server](#)" for more information.
- See "[Possible Problems](#)" for information about dealing with problems with your customizations.

Enabling the Customization Framework

1. Use the Convergence **iwadmin** command-line utility to verify that Convergence customizations are not enabled (the default).

```
iwadmin -o client.enablecustomization
```

The command line displays **client.enablecustomization = false** if customization is disabled.

2. Log in with Convergence client and view the user landing page.

3. Verify the `iwcc.log` and (optionally) Firebug console output.

- **`iwcc.log`:** Check for "Client customization service is disabled for the deployment:"

```
PROTOCOL: WARN from
com.sun.comms.client.protocol.delegate.agent.ClientOptionsAgent Thread
httpSSLWorkerThread-9000-0 at 11:21:16,948 - client preferences not found for
domain: sfbay.sun.com
.
.
.
PROTOCOL: INFO from com.sun.comms.client.entity.user.sun.CommsUser Thread
httpSSLWorkerThread-9000-0 at 11:04:40,021 - Client customization service is
disabled for the deployment
```

- **Firebug console:** Check for no console entries about customization when it is off. Here is the console entry that would precede the customization service entry:

```
Scroll bar dimensions: Object w=13 h=13
```

You should not see:

```
Loading Service: Customization Object enabled=true displayName=Customization
name=c11n
```

4. Enable customizations using the `iwccadmin` command:

```
iwccadmin -o client.enablecustomization -v true
```

5. Verify that the `iwcc_admin.log` shows the change:

```
ADMIN: INFO from com.sun.comms.client.admin.mbeans.CommonMBean Thread RMI TCP
Connection(7900)-129.145.185.117 at 11:49:40,804 - Value of config parameter
client.enablecustomization modified to true
```

See "[Enabling and Disabling Customization](#)" for more information about enabling and disabling customizations for domains and users.

Customizing Convergence

Customize Convergence, as outlined in this guide.

Restarting the Application Server

Customization changes do not take effect until you stop and start the application server. If you log in to the Convergence client before restarting the Oracle WebLogic Server, the content remains the same.

1. Stop the Oracle WebLogic Managed Server.
2. Start the Oracle WebLogic Managed Server.
3. Log in to Convergence.
4. Verify that the **`iwcc.log`** entry "Client customization service is disabled..." does not appear. The **`iwcc.log`** entry previously noted should not appear. In the entry, look soon after "ClientOptionsAgent."
5. Verify that Firebug console contains the following:

```
Loading Service: Customization Object enabled=true displayName=Customization
name=c11n
```

Possible Problems

If the Convergence customization does not appear to be active after making the **iwadmin** command change and restarting the Oracle WebLogic Server, the most likely cause of problems is that the server did not stop completely. View the Oracle WebLogic Server log for errors. For example:

For Oracle WebLogic Server 12.2.1.4, you can use the following tail command on the log file referenced by the output of start script:

```
# tail -f WLS_HOME/Oracle_Home/ user_projects/domains/base_domain/servers/mserver/logs/  
mserver.log  
.  
.  
.  
[BEA-149060] [Module iwc.war of application Convergence successfully transitioned from  
STATE_ADMIN to STATE_ACTIVE on server mserver.]  
[BEA-149060] [Module iwc_static.war of application Convergence_Client successfully  
transitioned from STATE_ADMIN to STATE_ACTIVE on server mserver.]
```

About the Customization Examples

The Convergence directory includes an example of customized code provided in Oracle WebLogic Server:

- **For Oracle WebLogic Server:** *data_directory/web-src/client/iwc_static/c11n_sample*
For example: */var/opt/sun/comms/iwc/web-src/client/iwc_static/c11n_sample* directory.

The directories for Oracle WebLogic Server are not live; that is, Convergence does not use the javascript and CSS files in these directories when they load files at run time.

The **c11n_sample** directory includes the following customization samples:

- Themes
- i18n samples
- A mail.CreateMessage widget
- helloConvergence service

Enabling Customization in the Convergence Server

Before you can use the customization example, the Convergence Server must be enabled for customization. Use the Convergence **iwadmin** command-line utility, to set the **client.enablecustomization** parameter to **true**.

```
iwadmin -o client.enablecustomization -v true
```

Customization Directory Structure

The location of the customization directory, called *c11n_Home* for Oracle WebLogic Server is as follows: *data_directory/web-src/client/iwc_static/c11n*

The custom files in *c11n_Home* extend the code base, overwriting the standard elements with the customized elements. Note the following information about *c11n_Home*:

- *c11n_Home* does not exist by default in the Oracle WebLogic Server.

- For Oracle WebLogic Server, you must create *c11_Home* in */var/opt/sun/comms/iwc/web-src/client/iwc_static* and the directory name must be **c11n**
- In *c11n_Home*, if **config.js** exists, customizations are booted based on how they are defined in this file.

For more information about the directory placeholders used in this guide, see [Table 1-1](#), "Convergence Directory Placeholders" for more information.

Creating the Customization Directory

The easiest way to create *c11n_Home* is to copy the sample customization directory and name it **c11n** in Oracle WebLogic Server, the sample customization directory is: */var/opt/sun/comms/iwc/web-src/client/iwc_static/c11n_sample*.

After creating the sample customization directory, restart the Oracle WebLogic Server so it can recognize and register the *c11n_Home* directory. After that, customization is ready to go.

The following example shows how to copy the **c11n_sample** to make *c11n_Home* for Oracle WebLogic Server:

```
cd /var/opt/sun/comms/iwc/web-src/client/iwc_static/c11n
cp -r c11n_sample c11n
```

You can also manually create *c11n_Home*.

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:

- **c11n_sample/allDomain**

- **c11n_sample/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**. 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 run time, 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 customization 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: false,     //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.

To Enable the mail.CreateMessage Widget

1. Create *c11n_Home*.
2. In *c11n_Home*, edit **config.js** 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. In *c11n_Home/allDomain/js*, create or modify **customize.js** and uncomment the following section:

```
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.

Technical Overview

This section discusses the following topics:

- [About How Topics Apply to My Customizations](#)
- [About Convergence Architecture Customization Support](#)

- [Dojo Basics](#)
- [About Dojo Statements Map to the Convergence Directory Structure](#)
- [Preserving Custom Widgets During Upgrades](#)

About How Topics Apply to My Customizations

The first topic, "[About Convergence Architecture Customization Support](#)" applies to all types of customization.

The next two topics, "[Dojo Basics](#)" and "[About Dojo Statements Map to the Convergence Directory Structure](#)" apply only to more "advanced" types of customization.

You must use dojo to customize Convergence widgets. For example, to add features to the banner, or to integrate a third-party application, you will customize widgets.

To integrate an outside application, for example, you may want to add a button or icon (or both) to the UI to provide access to the application. Buttons and icons are widgets; to create them, you must write custom javascript with dojo.

If you only want to change the theme (look and feel) of the Convergence UI, or offer alternate themes to end users, you customize the **theme.json** file. You do not need to write dojo code. You do not need the information in the second and third sections.

Finally, "[Preserving Custom Widgets During Upgrades](#)" mainly concerns preserving dojo widget customizations, but also briefly describes how CSS or **theme.json** files are preserved during upgrades.

About Convergence Architecture Customization Support

The Convergence architecture separates the standard UI elements from the corresponding customized elements. To customize the UI, you do *not* alter the standard UI definitions; these always remain in the code. Instead, you *add* customizations in a separate directory. The relationship between them is set up as follows:

- A single style-sheet file, named **template.css**, defines all the UI elements. These elements determine the look and feel of the standard UI.
- All images in the **template.css** file are defined as background images. If no customized images exist, the background images are loaded, and the default UI appears to the user.
- A *theme* is a collection of styles that affect the background color, background images, border color, and font color. Rather than modifying the CSS files directly in a theme, Convergence uses a **theme.json** file to map UI styling into CSS format. See "[Working with the Convergence UI](#)" for more information.
- Your other customized definitions, which create custom Convergence and dojo widgets, redefine the default UI elements at run time without altering the default source code.

Convergence Customization Booting

Convergence searches for `c11n_Home/config.js`. If this file exists, the customization service boots your customization code based on the configuration setup defined in the **config.js** file.

Customization Loading Order at Run Time

The technique of customization begins with the loading order of the script and CSS files.

First, the files containing all the Convergence source code are loaded. These default files include the HTML, CSS, image, and javascript files.

Second, if customization is enabled, the customization code is loaded. The customization files also include CSS, image, and javascript files.

Third, the actual Convergence UI and widgets are created. The creation process uses both the customized code and standard source code to build the UI. That is, the creation process combines all your customized elements with the standard elements defined in the default code. For example, if you customize only the banner, the creation process combines the customized banner with the standard elements in the rest of the UI. The components of the standard banner that have been customized are now suppressed.

Loading Order for Multiple Domains

Customization is designed to work either in a single-domain or hosted-domain environment. For hosted domains, customization is applied as follows:

1. The settings defined in the base code are loaded and applied.
2. Customizations for the specific domain in which the end user has logged in are loaded and applied. Different domains can display the UI in different languages.

Thus, the loading order follows the same principle of applying the more general definitions first, followed by the more specific customizations.

Loading Order Across All Domains

You must use a reserved keyword, **allDomain**, to create customizations that apply across all domains in the Convergence deployment.

The loading order for customizations in the **allDomain** directory is the same as for any other domain.

Thus, at run time, Convergence behaves as follows:

1. It loads the base-code settings.
2. It loads **either** the customizations defined in **allDomain** *or* the customizations defined in a specific domain (if customizations have been created for that domain).

Directory Layout

When Convergence is installed, the client component of the Convergence software is installed in *Convergence_Domain/docroot/iwc_static*.

All client source files, including the customization files, reside in the **iwc_static** directory. By default, **iwc_static** contains the following directories:

```
/layout  
/js/dojotoolkit  
/js/iwc  
/c11n_sample
```

If you create customizations, they must be stored in the *c11n_Home* directory, which must be named **c11n** under directory **iwc_static**. Thus, the *c11n_Home* directory must be

```
/iwc_static/c11n
```

When you patch or upgrade Convergence, files in the standard Convergence directories are updated and replaced. However, the patch or upgrade does not touch other directories and files not originally installed by the installer. The *c11n_Home* directory is left untouched.

Dojo Basics

Dojo code is required for customizing Convergence widgets such as the banner, buttons, icons, and so on. You may want to add or alter widgets to customize the banner or integrate a third-party application.

This release of the Convergence customization framework is built on Dojo 1.3.2. See the Dojo web site for more information:

<http://docs.dojocampus.org/>

Table 1-2 shows the dojo statements that provide the foundation for creating dojo classes:

Table 1-2 Dojo Statements

dojo Statement	Function
<code>dojo.provide("x.y.z")</code>	Tells dojo that the module x.y.z is defined, and puts x.y.z into the dojo internal hash.
<code>dojo.require("x.y.z")</code>	Finds module x.y.z from the dojo internal hash. If the module cannot be found in the internal hash, load it.
<code>dojo.declare(className, superclass, properties)</code>	Creates a new className from the superclass and passes in the properties associated with the new class.

Using the Debugging Directory to Customize Convergence

Dojo minifies or compresses *layers*, single JavaScript files which combine all of the JavaScript code from multiple source files, including dependencies.

According to the dojo documentation, minification makes your JavaScript code smaller by:

- Removing extra spaces and blank lines
- Removing any comments
- Making internal variable names shorter

Minification can make the layers load faster, and can take less time for the browser's JavaScript engine to parse.

However, minified files (for example, for Oracle WebLogic Server: *data_directory/web-src/client/iwc_static/js/iwc*) are often difficult to read and debug because they are compressed.

Therefore, debugging directories are provided in the Convergence customization libraries for Oracle WebLogic Server: *data_directory/web-src/client/iwc_static/jsdebug/iwc*.

The debugging directory provides the original, uncompressed content which includes source code, HTML templates, and the original Convergence file directory structure that corresponds to the source code. The debugging directory provides clear references to the Convergence source code, making it easier to read than in the compressed format.

From Convergence 3.0.3.4, the **buildscripts** folder and few jars under the **shrinksafe** folder are removed from the *data_directory/web-src/client/iwc_static/jsdebug/dojotoolkit/util* folder. Therefore, before building the code under the debugging directory *data_directory/web-*

src/client/iwc_static/jsdebug, download the dojo toolkit from the dojo website and unzip it: <https://download.dojotoolkit.org/release-1.3.2/dojo-release-1.3.2-src.zip> file and unzip it.

- Copy the **<unzip_location>/dojo-release-1.3.2-src/util/buildscripts** folder to **data_directory/web-src/client/iwc_static/jsdebug/dojotoolkit/util/**
- Copy **<unzip_location>dojo-release-1.3.2-src/util/shrinksafe/*.jar** to **data_directory/web-src/client/iwc_static/jsdebug/dojotoolkit/util/shrinksafe .**

 **Note:**

When you have completed your customization, you will need uncompressed JavaScript code for debugging. You can load the uncompressed JavaScript code for debugging by loading the **login.html** page with the **isdebug** parameter to true. You can customize Convergence using the widgets available under the **jsdebug** folder. After you complete customizing, remove the **jsdebug** folder from **data_directory/web-src/client/iwc_static/**.

About Dojo Statements Map to the Convergence Directory Structure

Dojo code maps to the following directories in Convergence:

For example:

```
dojo.require("iwc.widget.Banner")
```

refers to a file located in

```
iwc_static/js/iwc/widget/Banner.js
```

For example:

```
dojo.require("c11n.allDomain.js.widget.Banner")
```

refers to a file located in

```
iwc_static/c11n/allDomain/js/widget/Banner.js
```

Preserving Custom Widgets During Upgrades

The Convergence base code, in particular the dojo javascript files that define the UI widgets, are designed to allow you to create custom files that can be preserved automatically when the Convergence software is upgraded.

When you create custom files in *c11n_Home*, those files are not touched during upgrades. Therefore, they are always preserved.

However, when you create customized dojo widgets, you can use a few different methods to copy or extend the base dojo code. Each will have different ramifications when the Convergence software is upgraded.

Before you begin coding, you should plan which approach to take. Choose the one that best suits your goals.

First, let's briefly look at the simplest type of customization: how custom themes are handled during upgrades.

Preserving Themes During Upgrades

It is not possible to upgrade themes created for Convergence 1.x. Themes created for Convergence 2.x are automatically preserved when upgrading to later versions of Convergence. However, if in future releases, new variables are added to the dojo template file or to specific themes, you will need to add those values to your **theme.json** files and any new images.

Preserving Dojo Widgets During Upgrades

By default, custom widgets residing in the *c11n_Home* directory are preserved during Convergence upgrade. However, changes are made to the base code directory and to file names, your custom widgets may need to be modified. To do so:

1. Map your custom widgets to the new widget name: see "[About Convergence UI Widgets](#)" to determine new widget names.
2. Review the examples in the **c11n_sample** directory and in this guide to determine if the functionality of your custom widgets is similar to the provided examples. You can then modify your custom widgets to match the provided examples.
3. If you are unable to match your custom widget to the provided examples, change the widget name, **dojo.require**, and **dojo.declare** lines.
 - Review the base code widget to determine if the DOM node(s) and **dojoAttachPoint** have same names.
 - After changing your widget names, you can run a tool such as Firebug for Firefox to debug your custom widgets.
 - If after following these steps, you are still unable to retain your widget customization, contact your support channels.

Consolidating Convergence Customizations to Preserve Client Performance

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 individual files, you can add the customizations directly to **customize.js**. You can have an unlimited number of customizations, but still only one file will be required.

For example, instead of:

c11n_Home/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_Home/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_Home/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.

About the 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 you can use to customize the UI with the **custom-useroptions.properties** file.

The **custom-useroptions.properties** file exists in the following directory: **/var/opt/sun/comms/iwc/config/**

Use **custom-useroptions.propertiesUsed with:iwc.protocol.iwcp.setUserPrefs** to set custom attributes. Once defined, the custom properties can be accessed in the **iwc.userPrefs.custom** javascript object. You can update the custom properties by using **iwc.protocol.iwcp.setUserPrefs**.

Structure of custom-useroptions.properties Mapping File

The following file syntax describes how to map custom LDAP attributes in **custom-useroptions.properties**, where *<mapping-name>* is the parameter name to be passed to **setUserprefs** and *<ldap-attribute>* is the LDAP attribute to be mapped.

```
# 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 preference  
# 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  
});
```

2

Enabling and Disabling Customization

You can control whether to enable customization for an entire deployment, for specific domains, or for specific users.

Before you can enable customization for an individual user or domain, you must enable customization for the entire Convergence deployment.

About Enabling Customization for the Deployment, Domains, and Users

Since you can enable or disable customization at all three levels (whole deployment, domain, and individual user), it is important to understand how the settings at each level affect one another.

The following rules and guidelines explain these relationships:

- By default, customization for the whole Convergence deployment is disabled.
- If you enable customization for the whole deployment, customization is enabled for all domains and all users within the domains.
- However, if you disable customization for a domain, users in that domain have no access to customization, even when customization is enabled for the whole deployment.
- Similarly, if you disable customization for a user, that user has no access to customization, even when the settings are enabled at the higher levels.
- If you disable customization at the deployment level, customization is disabled for all domains and all users, no matter what value is set at the domain or user level.
- At any level, disabling customization overrides enabling it.

You enable or disable customization at the domain and user levels by setting LDAP attribute values. Thus, there are three possibilities at these levels:

- Do not add the customization LDAP attribute to the domain or user entry.
- Add the attribute and set it to **true**.
- Add the attribute and set it to **false**.

If you explicitly set the customization LDAP attribute to **true** or **false**, the rules and guidelines described previously apply.

If you do not add the LDAP attribute to a user or domain, the settings at the higher level(s) apply.

Since you **must** enable customization for the whole deployment to enable customization for any individual domain or user, setting the customization LDAP attribute to **true** does not change the status for that domain or user.

In practice, the way to target access to customization per domain or per user is to turn on customization for the deployment and then set the LDAP attribute to **false** for each domain or user to whom you want to prevent access to customization.

Enabling Customization for the Convergence Deployment

1. Enable the Convergence Server for customization using the **iwcadm** command:

```
iwcadm -o client.enablecustomization -v true
```
2. Populate the *c11n_Home* directory with the required directories and customization files. One approach is to copy the sample customization files from the sample customization directory to the live directory. Copy

```
/iwc_static/c11n_sample
```


to

```
/iwc_static/c11n
```

Enabling or Disabling Customization for an Individual User

1. Enable customization for Convergence. See "[Enabling Customization for the Convergence Deployment](#)" for more information.
2. Add the following LDAP attribute to the user entry:

```
sunUCExtendedUserPrefs: ClientCustomizationEnabled=true
```

To disable customization for the user, set the value of the attribute to false:

```
sunUCExtendedUserPrefs: ClientCustomizationEnabled=false
```



Note:

Before adding the **sunUCExtendedUserPrefs** attribute, ensure the domain entry contains the **sunucpreferences** object class.

Enabling or Disabling Customization for an Individual Domain

1. Enable customization for Convergence. See "[Enabling Customization for the Convergence Deployment](#)" for more information.
2. Add the following LDAP attribute to the domain entry:

```
sunUCExtendedClientPrefs: ClientCustomizationEnabled=true
```

To disable customization for the domain, set the value of the attribute to false:

```
sunUCExtendedClientPrefs: ClientCustomizationEnabled=false
```

3

About Convergence UI Widgets

This chapter identifies many UI elements in Oracle Communications Convergence.

Location of Javascript Widgets

The widgets are located in the *Convergence_Domain/docroot/iwc_static/js/iwcl/widget* directory.

Widgets for each service are located in separate directories:

- Mail widgets: **../widget/mail**
- Calendar widgets: **../widget/calendar**
- Address Book widgets: **../widget/addressBook**

Audio/visual widgets are located in the following directory:

- Common audio/visual widgets: **../widget**

Option widgets are located within each service directory:

- Mail Options: **../widget/mail/option**
- Calendar Options: **../widget/calendar/option**
- Address Book Options: **../widget/addressBook/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 home directory. For example:

c11n_Home/Domain/js/widget

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

c11n_Home/allDomain/js/widget

See "[Technical Overview](#)" for more information.

Common Widgets

The common Convergence widgets are shown in the following figures:

- [Figure 3-1](#), Common Widgets in Convergence UI
- [Figure 3-2](#), Convergence UI Recipient Widget

[Figure 3-1](#) shows the location of the following widgets:

1. The Banner widget (**Banner.js**)

2. The QuickActions widget (**QuickActions.js**)
3. The ServiceMenu widget (**ServiceMenu.js**)
4. The SaveNotification widget (**SaveNotification.js**)
5. The SaveNotificationMessage widget (**SaveNotificationMessage.js**)

Figure 3-1 Common Widgets in Convergence UI

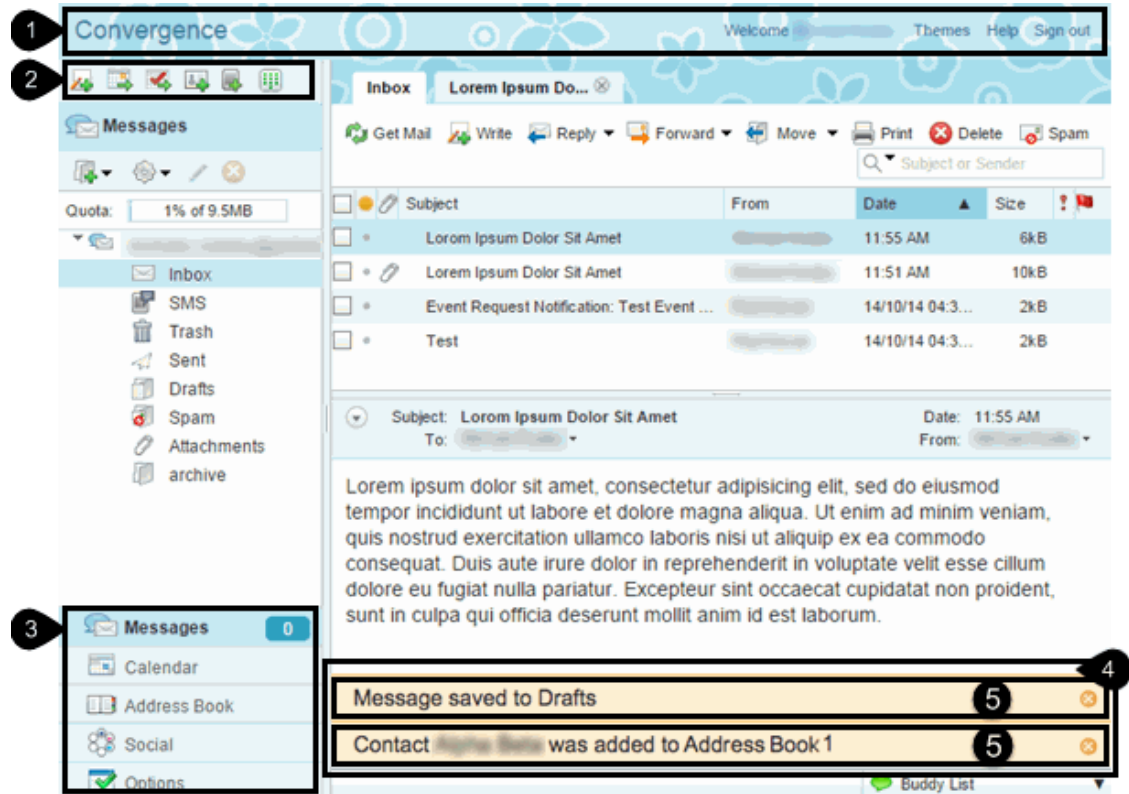
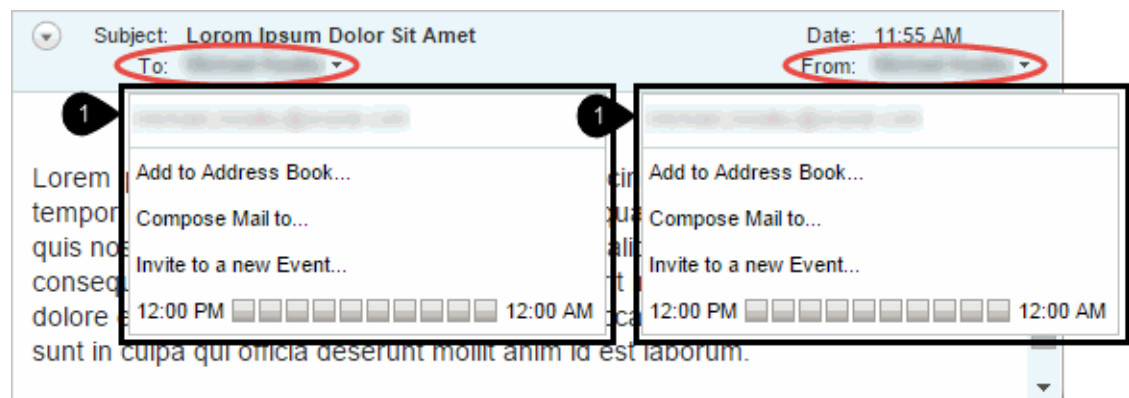


Figure 3-2 shows the location of the following widgets:

1. The Recipient widget (**Recipient.js**)

Figure 3-2 Convergence UI Recipient Widget



Mail Widgets

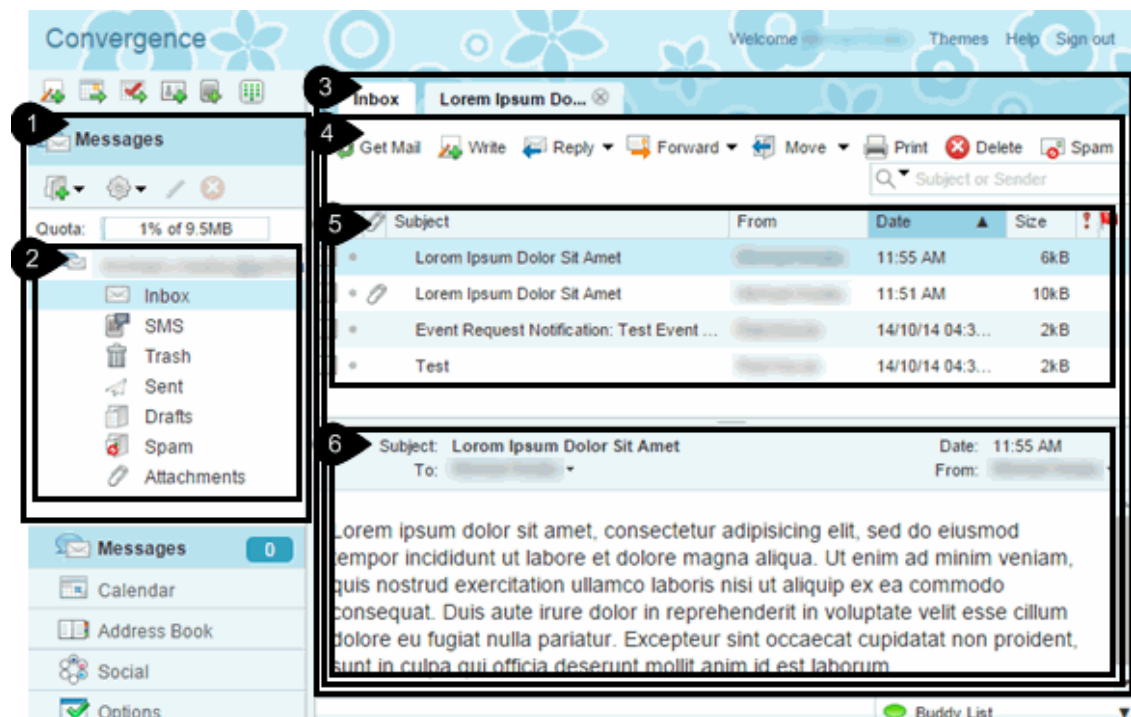
The Convergence mail widgets are shown in the following figures:

- [Figure 3-3](#) Common Convergence Mail Widgets
- [Figure 3-4](#) Convergence UI Recipient Widget
- [Figure 3-5](#) Convergence Mail Open Message Widget
- [Figure 3-6](#) Convergence Mail Advanced Search and Select Folder Input Widgets
- [Figure 3-7](#) Convergence Mail Print Message Widget
- [Figure 3-8](#) Convergence Mail Folder Properties Dialog Widget
- [Figure 3-9](#) Convergence Mail Folder Dialog Widget

[Figure 3-3](#) shows the location of the following widgets:

1. The Navigator widget (**mail.Navigator.js**)
2. The FolderTree widget (**mail.FolderTree.js**)
3. The ViewerContainer widget (**mail.ViewerContainer.js**)
4. The OpenFolder widget (**mail.OpenFolder.js**)
5. The Grid widget (**mail.Grid.js**)
6. The MessageViewer widget (**mail.MessageViewer.js**)

Figure 3-3 Common Convergence Mail Widgets



[Figure 3-5](#) shows the location of the following widgets:

1. The CreateMessage widget (**mail.CreateMessage.js**)
2. The EmailComboTextarea widget (**addressBook.EmailComboTextarea.js**)

Figure 3-4 Convergence UI Recipient Widget

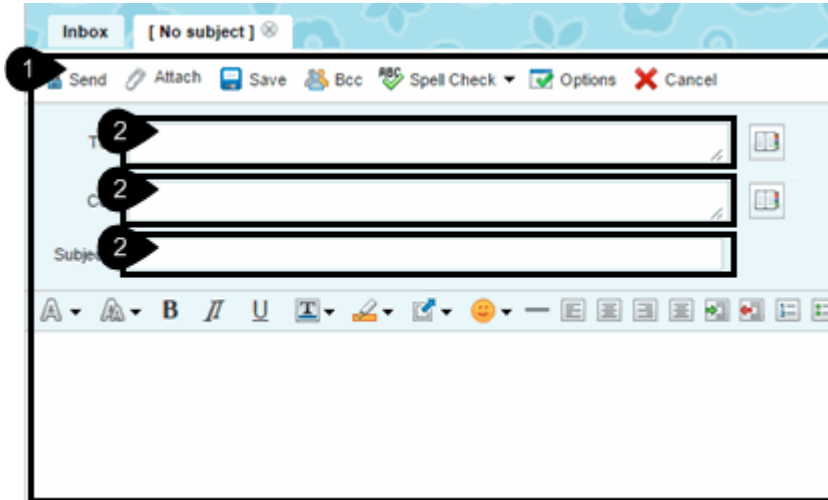


Figure 3-5 shows the location of the following widgets:

1. The OpenMessage widget (**mail.OpenMessage.js**)

Figure 3-5 Convergence Mail Open Message Widget



Figure 3-6 shows the location of the following widgets:

1. The SelectFolderInput widget (**mail.SelectFolderInput.js**)
2. The AdvancedSearch widget (**mail.AdvancedSearch.js**)

Figure 3-6 Convergence Mail Advanced Search and Select Folder Input Widgets

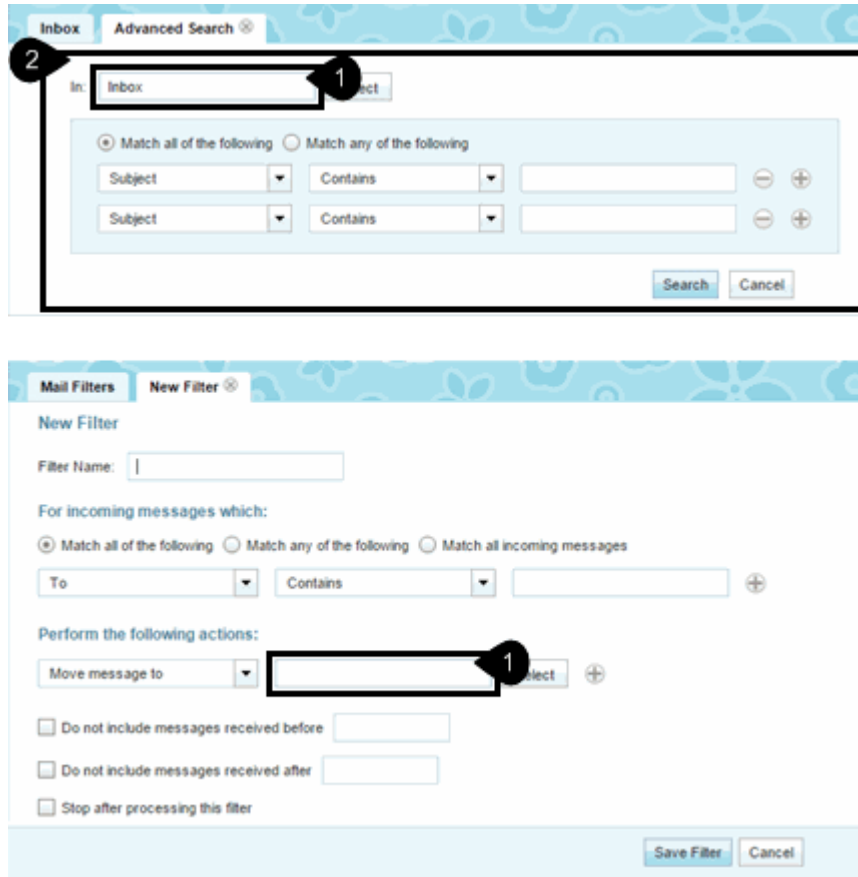


Figure 3-7 shows the location of the following widgets:

1. The PrintMessage widget (`mail.PrintMessage.js`)

Figure 3-7 Convergence Mail Print Message Widget

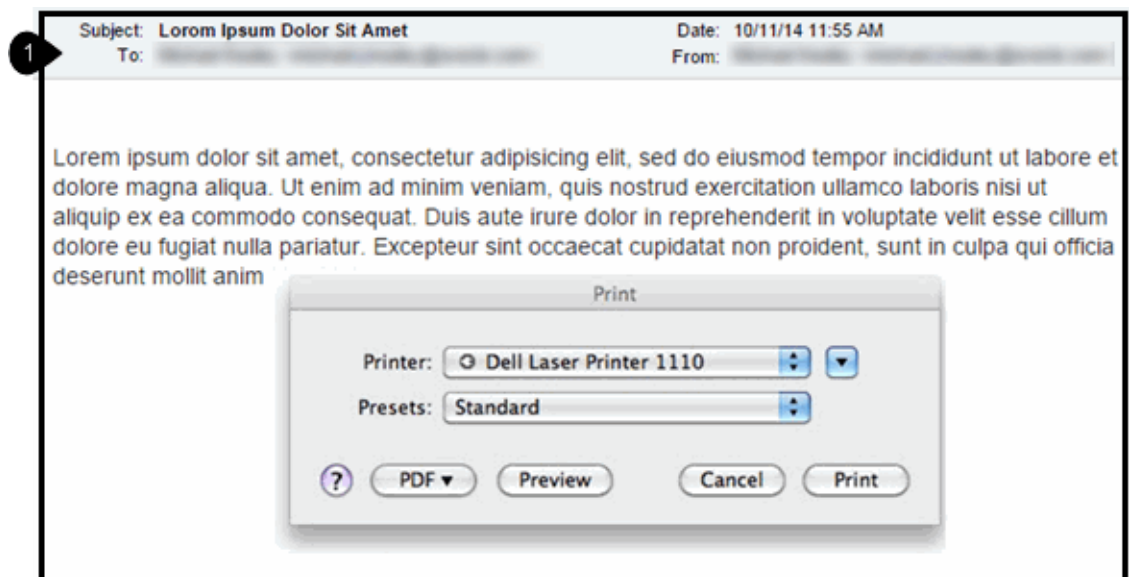


Figure 3-8 shows the location of the following widgets:

1. The FolderPropertiesDialog widget (**mail.FolderPropertiesDialog.js**)

Figure 3-8 Convergence Mail Folder Properties Dialog Widget

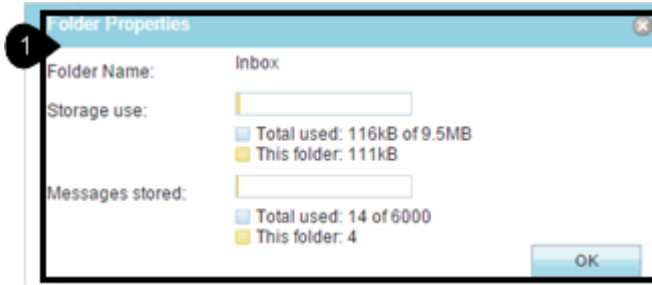
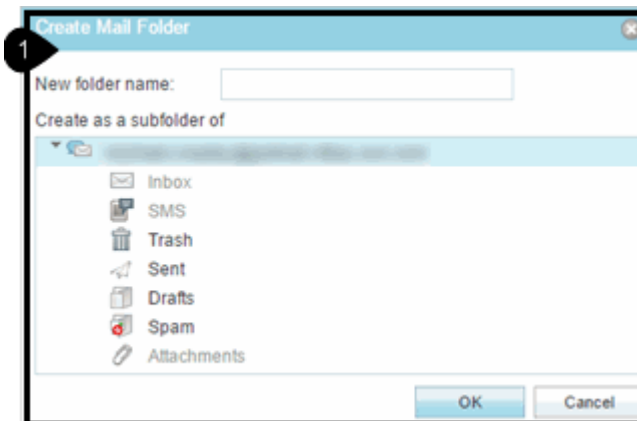


Figure 3-9 shows the location of the following widgets:

1. The FolderDialog widget (**mail.FolderDialog.js**)

Figure 3-9 Convergence Mail Folder Dialog Widget



Address Book Widgets

The Convergence address book widgets are shown in the following figures:

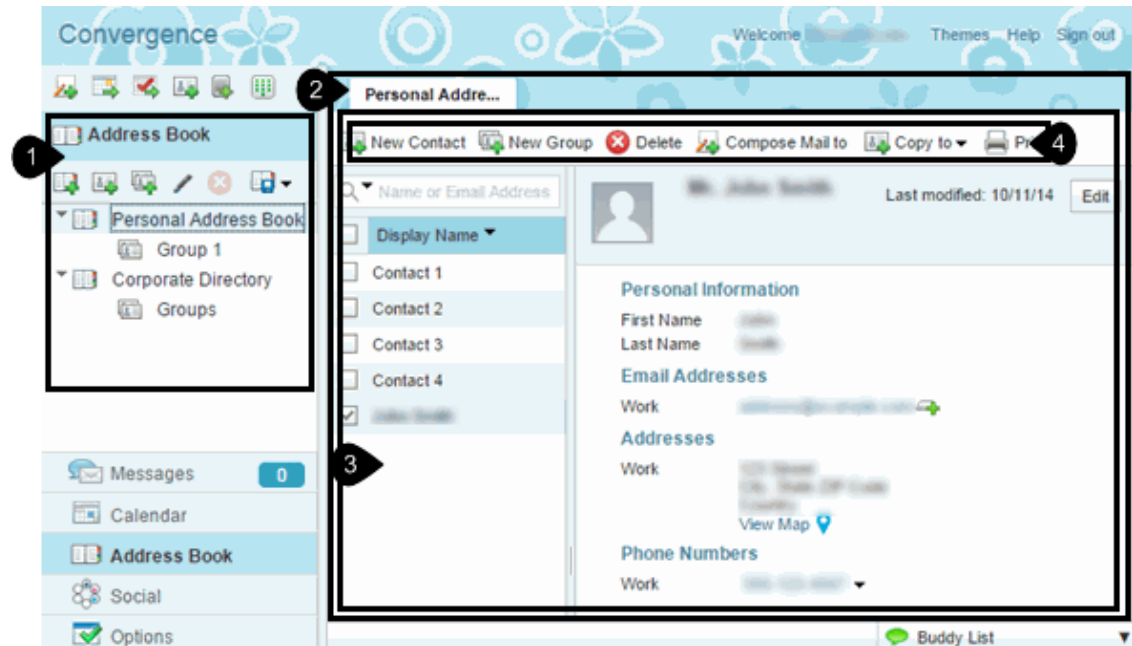
- [Figure 3-10](#) Common Convergence Address Book Widgets
- [Figure 3-11](#) Convergence Address Book Corporate Book Browser Widget
- [Figure 3-12](#) Convergence Address Book Create Contact Widget
- [Figure 3-13](#) Convergence Address Book Create Group Widget
- [Figure 3-14](#) Convergence Address Book Export Contacts Dialog
- [Figure 3-15](#) Convergence Address Book Import Contacts Dialog Widget
- [Figure 3-16](#) Convergence Address Book Create Contact Dialog Widget
- [Figure 3-17](#) Convergence Address Book Book Store Item Selector Widget

- [Figure 3-18](#) Convergence Address Book Resource Store Item Selector Widget

[Figure 3-10](#) shows the location of the following widgets:

1. The Navigator widget (**addressBook.Navigator.js**)
2. The ViewerContainer widget (**addressBook.ViewerContainer.js**)
3. The PersonalBookBrowser widget (**addressBook.PersonalBookBrowser.js**)
4. The BookBrowserToolbar widget (**addressBook._BookBrowserToolbar.js**)

Figure 3-10 Common Convergence Address Book Widgets



[Figure 3-11](#) shows the location of the following widgets:

1. The CorporateBookBrowser widget (**addressBook.CorporateBookBrowser.js**)

Figure 3-11 Convergence Address Book Corporate Book Browser Widget

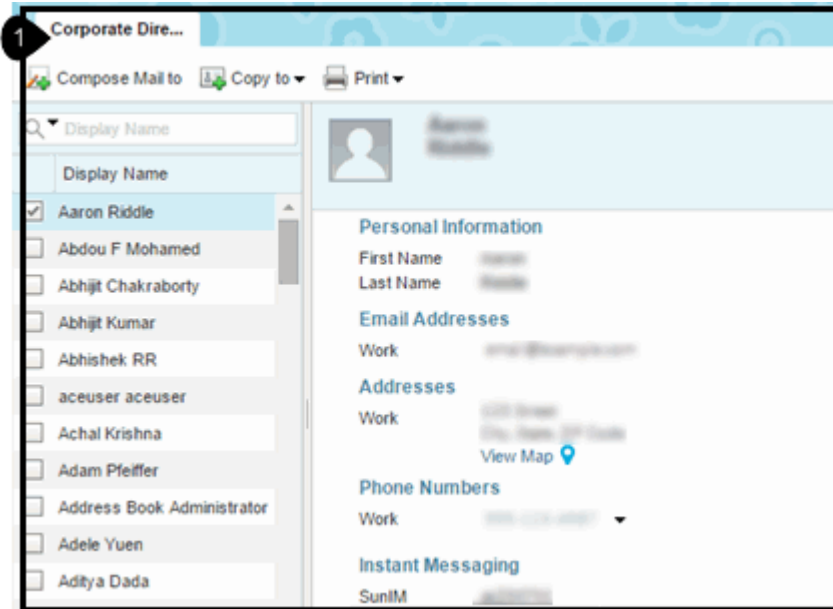


Figure 3-12 shows the location of the following widgets:

1. The CreateContact widget (`addressBook.CreateContact.js`)

Figure 3-12 Convergence Address Book Create Contact Widget

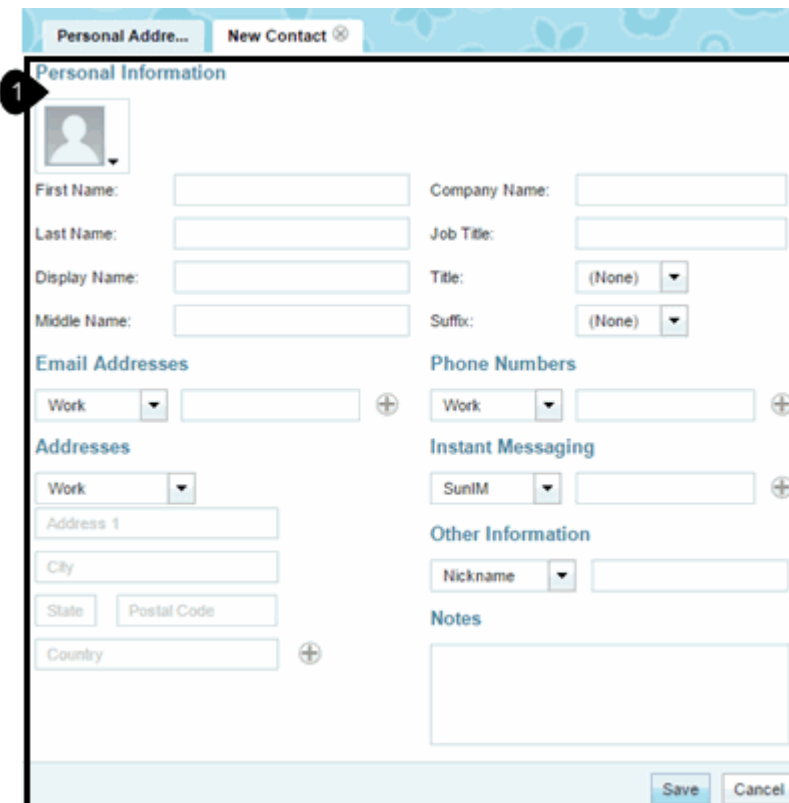


Figure 3-13 shows the location of the following widgets:

1. The CreateGroup widget (`addressBook.CreateGroup.js`)

Figure 3-13 Convergence Address Book Create Group Widget

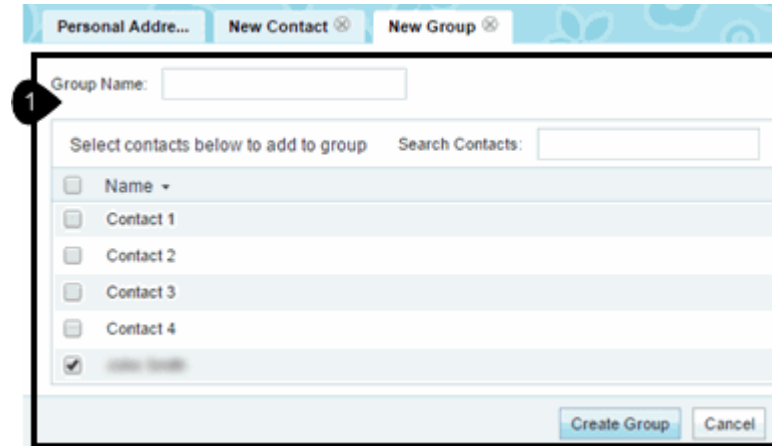


Figure 3-14 shows the location of the following widgets:

1. The ExportContactsDialog widget (`addressBook.ExportContactsDialog.js`)

Figure 3-14 Convergence Address Book Export Contacts Dialog Widget

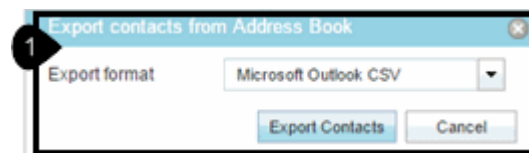


Figure 3-15 shows the location of the following widgets:

1. The ImportContactsDialog widget (`addressBook.ImportContactsDialog.js`)

Figure 3-15 Convergence Address Book Import Contacts Dialog Widget

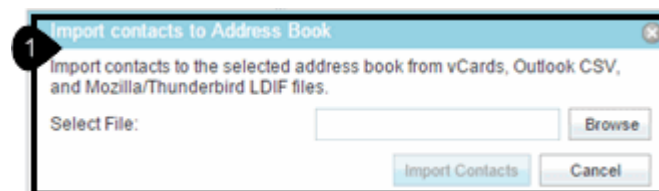


Figure 3-16 shows the location of the following widgets:

1. The CreateContactDialog widget (`addressBook.CreateContactDialog.js`)

Figure 3-16 Convergence Address Book Create Contact Dialog Widget

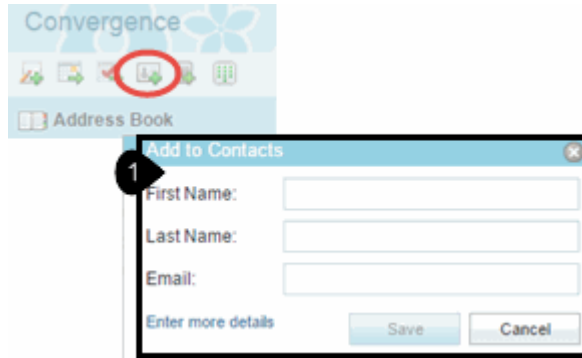


Figure 3-17 shows the location of the following widgets:

1. The BookStoreItemSelector widget (`addressBook.BookStoreItemSelector.js`)

Figure 3-17 Convergence Address Book Book Store Item Selector Widget

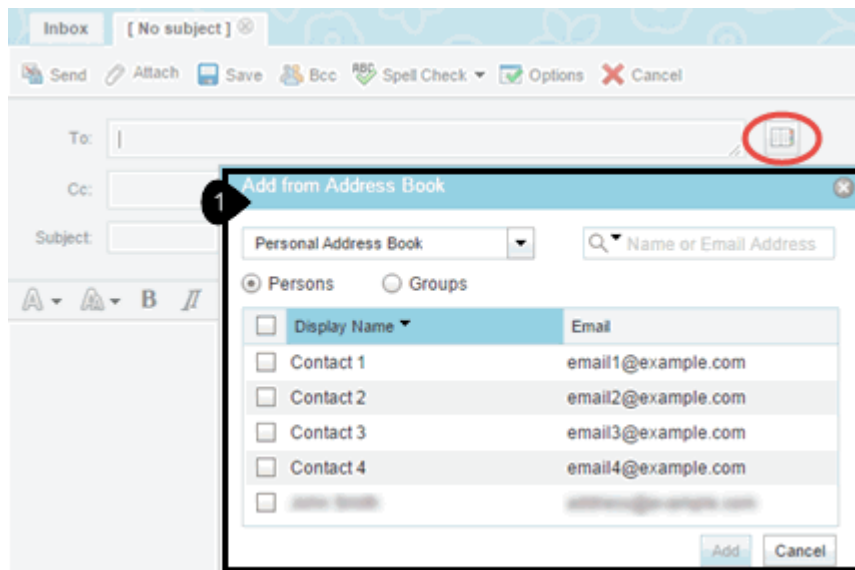
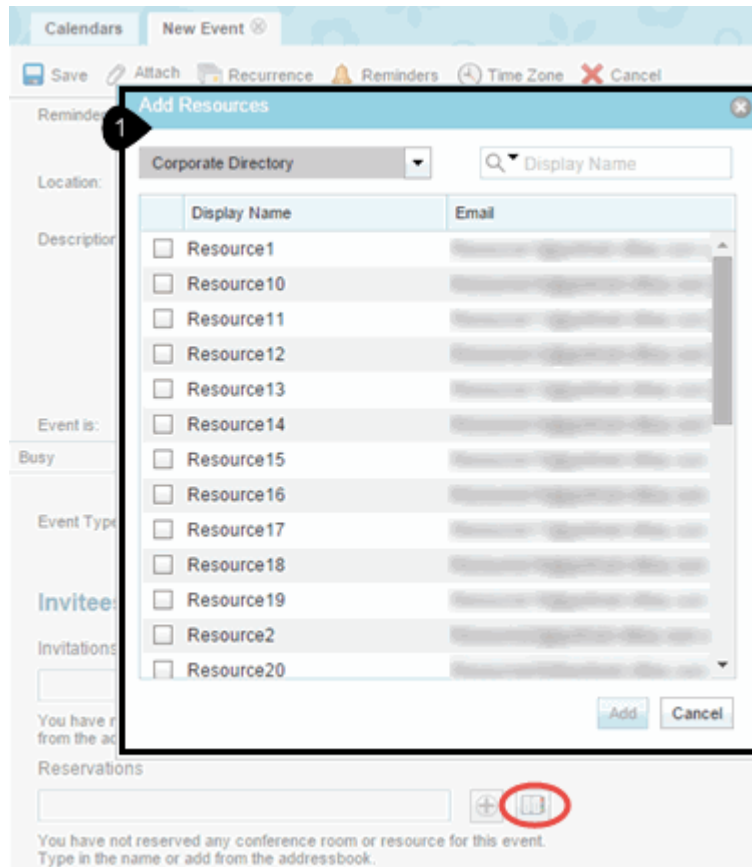


Figure 3-18 shows the location of the following widgets:

1. The ResourceStoreItemSelector widget (`addressBook.ResourceStoreItemSelector.js`)

Figure 3-18 Convergence Address Book Resource Store Item Selector Widget



Calendar Widgets

The Convergence Calendar widgets are shown in the following figures:

- [Figure 3-19](#) Common Convergence Calendar Widgets
- [Figure 3-20](#) Convergence Calendar Day View Widget
- [Figure 3-21](#) Convergence Calendar Week View Widget
- [Figure 3-22](#) Convergence Calendar Next 7 View Widget
- [Figure 3-23](#) Convergence Calendar Month View Widget
- [Figure 3-24](#) Convergence Calendar List View and View Event Item Widgets
- [Figure 3-25](#) Convergence Calendar List View and View Invites Item Widgets
- [Figure 3-26](#) Convergence Calendar List View and View Task Item Widgets
- [Figure 3-27](#) Convergence Calendar Event Widget
- [Figure 3-28](#) Convergence Calendar Monthly Events Widget
- [Figure 3-29](#) Convergence Calendar Create Events and Invitees Widgets
- [Figure 3-30](#) Convergence Calendar Recurrence Dialog Widget
- [Figure 3-31](#) Convergence Calendar Create Task Dialog Widget
- [Figure 3-32](#) Convergence Calendar Task Detail Widget

- [Figure 3-33](#) Convergence Calendar Event Balloon Widget
- [Figure 3-34](#) Convergence Calendar View Event Widget
- [Figure 3-35](#) Convergence Calendar Task Detail Widget
- [Figure 3-36](#) Convergence Calendar Availability Widget
- [Figure 3-37](#) Convergence Calendar Notification (Reminder) Dialog Widget
- [Figure 3-38](#) Convergence Calendar Print Dialog Widget
- [Figure 3-39](#) Convergence Calendar Print Widget
- [Figure 3-40](#) Convergence Calendar Time Zone Dialog Widget
- [Figure 3-41](#) Convergence Calendar Export Dialog Widget
- [Figure 3-42](#) Convergence Calendar Import Dialog Widget
- [Figure 3-43](#) Convergence Calendar Subscribe Widget

Figure 3-19 shows the location of the following widgets:

1. The Navigator widget (`calendar.Navigator.js`)
2. The Calendar widget (`digit._Calendar.js`)
3. The ViewerContainer widget (`calendar.ViewerContainer.js`)
4. The ViewDispatcher widget (`calendar.ViewDispatcher.js`)
5. The Event widget (`calendar.Event.js`)

Figure 3-19 Common Convergence Calendar Widgets

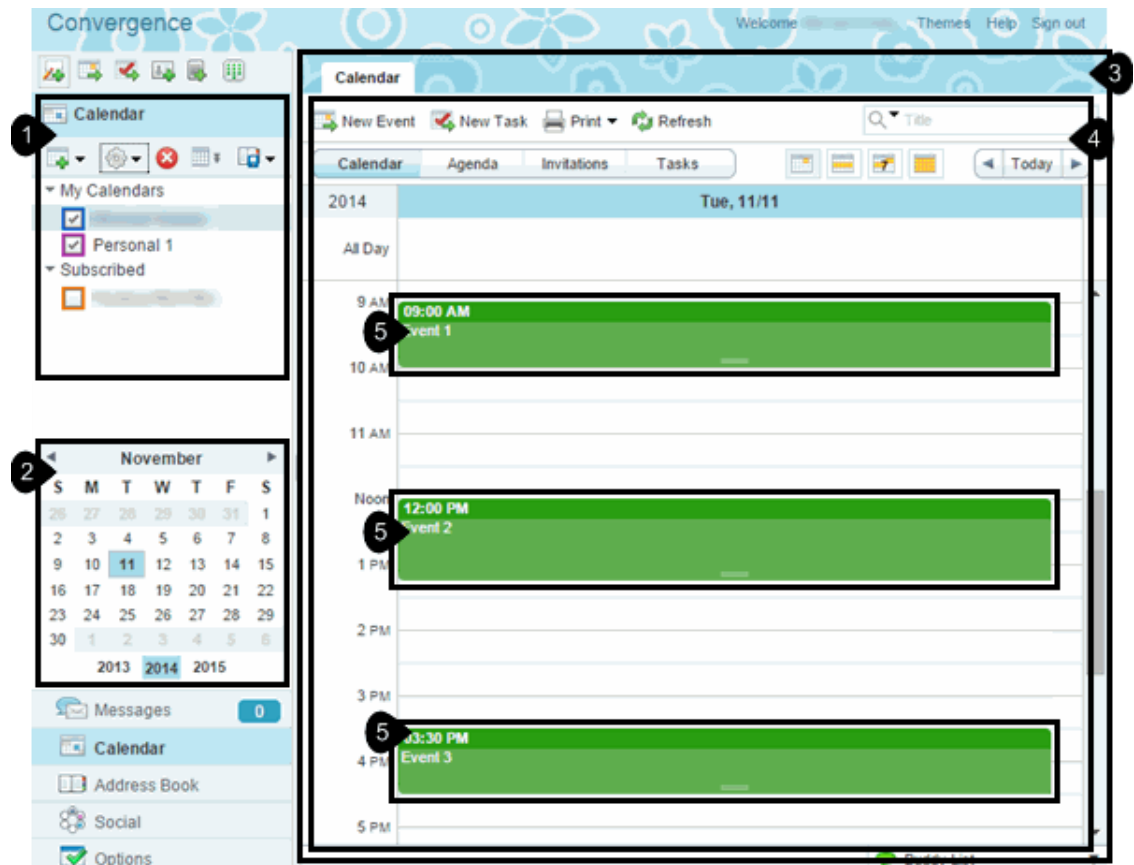


Figure 3-20 shows the location of the following widgets:

1. The DayView widget (`calendar.DayView.js`)

Figure 3-20 Convergence Calendar Day View Widget

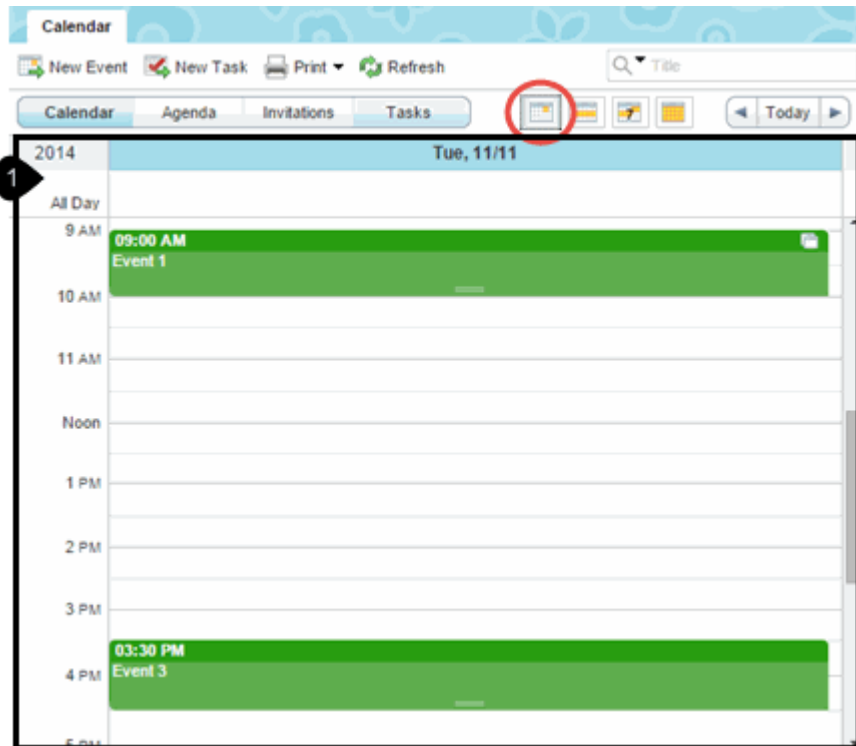


Figure 3-21 shows the location of the following widgets:

1. The WeekView widget (`calendar.WeekView.js`)

Figure 3-21 Convergence Calendar Week View Widget

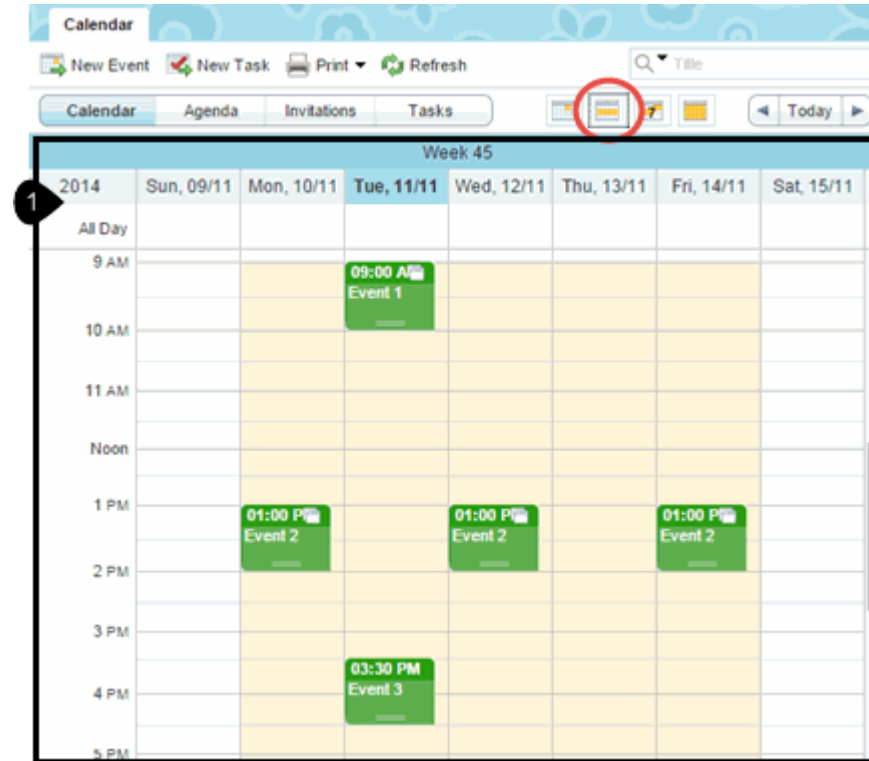


Figure 3-22 shows the location of the following widgets:

1. The Next7View widget (`calendar.Next7View.js`)

Figure 3-22 Convergence Calendar Next 7 View Widget

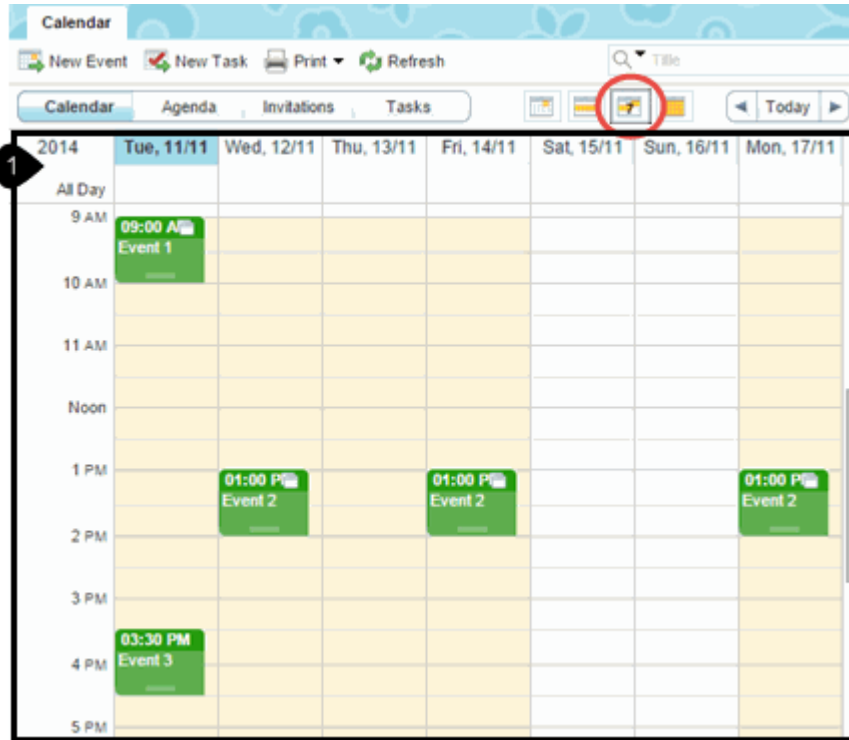


Figure 3-23 shows the location of the following widgets:

1. The MonthView widget (`calendar.MonthView.js`)

Figure 3-23 Convergence Calendar Month View Widget

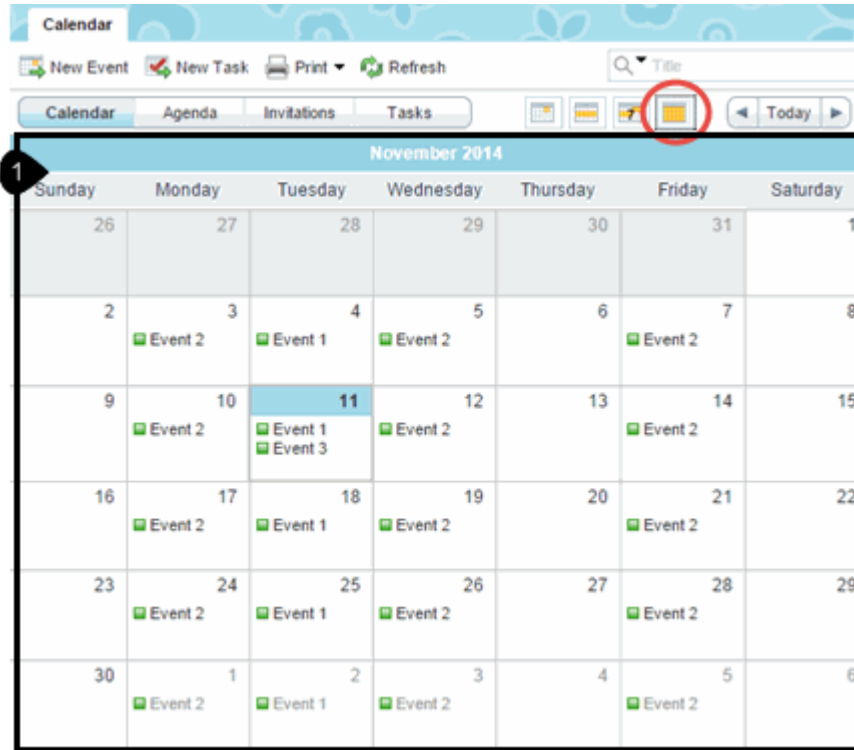


Figure 3-24 shows the location of the following widgets:

1. The ListView widget (`calendar.MonthView.js`)
2. The ListItemEvent widget (`calendar.ListItemEvent.js`)

Figure 3-24 Convergence Calendar List View and View Event Item Widgets

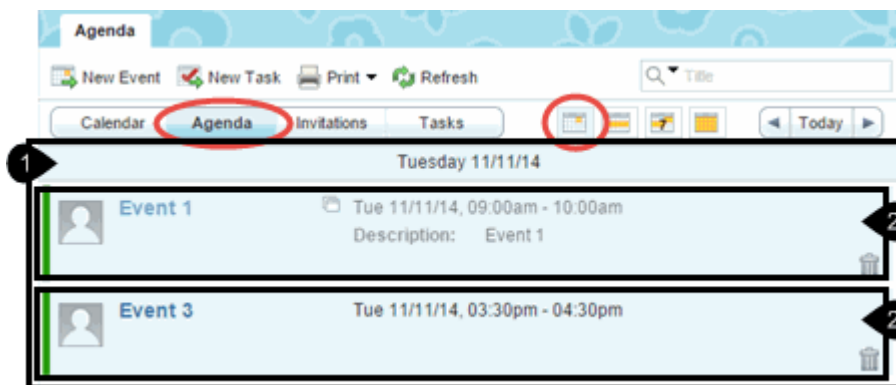


Figure 3-25 shows the location of the following widgets:

1. The ListView widget (`calendar.MonthView.js`)
2. The ListItemInvite widget (`calendar.ListItemInvite.js`)

Figure 3-25 Convergence Calendar List View and View Invites Item Widgets

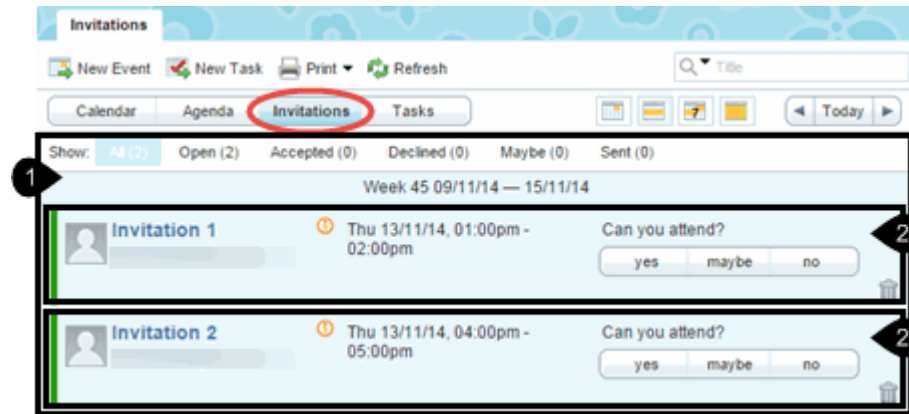


Figure 3-26 shows the location of the following widgets:

1. The ListView widget (`calendar.MonthView.js`)
2. The ListItemTask widget (`calendar.ListItemTask.js`)

Figure 3-26 Convergence Calendar List View and View Task Item Widgets

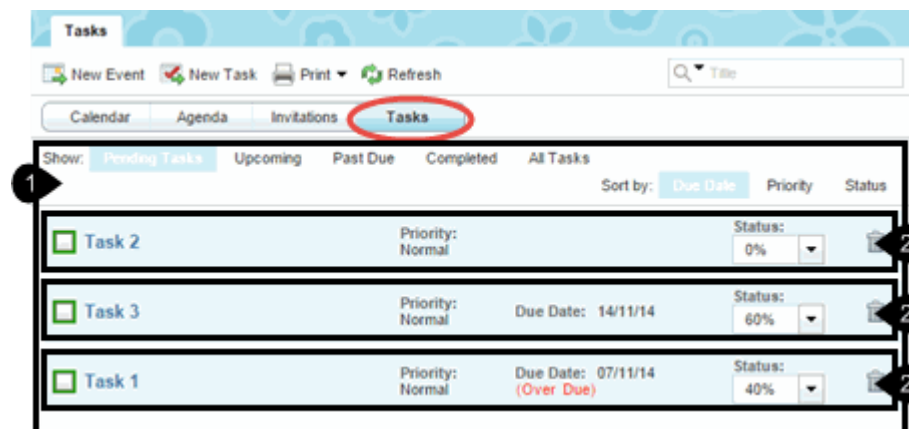


Figure 3-27 shows the location of the following widgets:

1. The Event widget (`calendar.Event.js`)

Figure 3-27 Convergence Calendar Event Widget

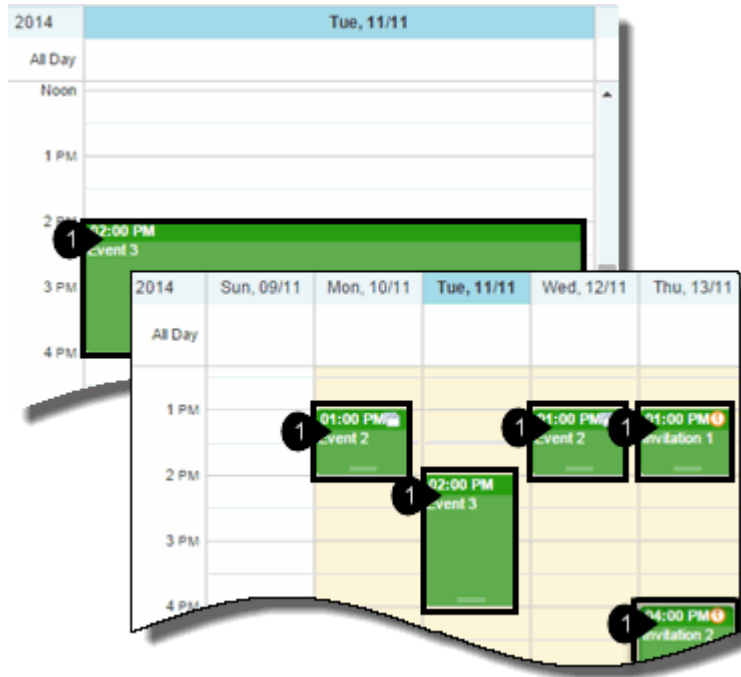


Figure 3-28 shows the location of the following widgets:

1. The MonthlyEvents widget (`calendar.MonthlyEvent.js`)

Figure 3-28 Convergence Calendar Monthly Events Widget

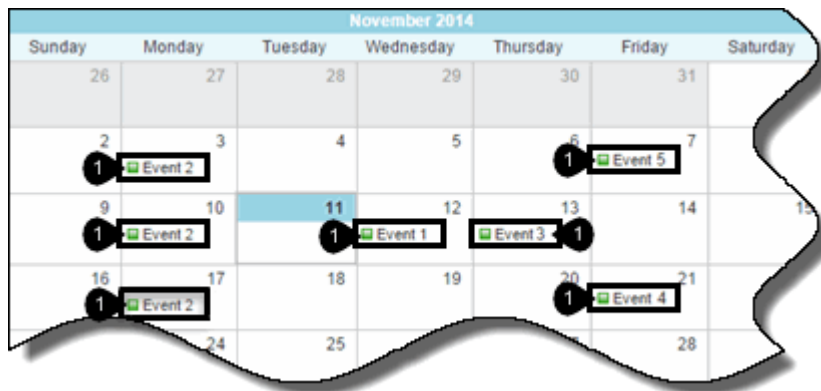


Figure 3-29 shows the location of the following widgets:

1. The CreateEvent widget (`calendar.CreateEvent.js`)
2. The Invitees widget (`calendar.Invitees.js`)

Figure 3-29 Convergence Calendar Create Events and Invitees Widgets

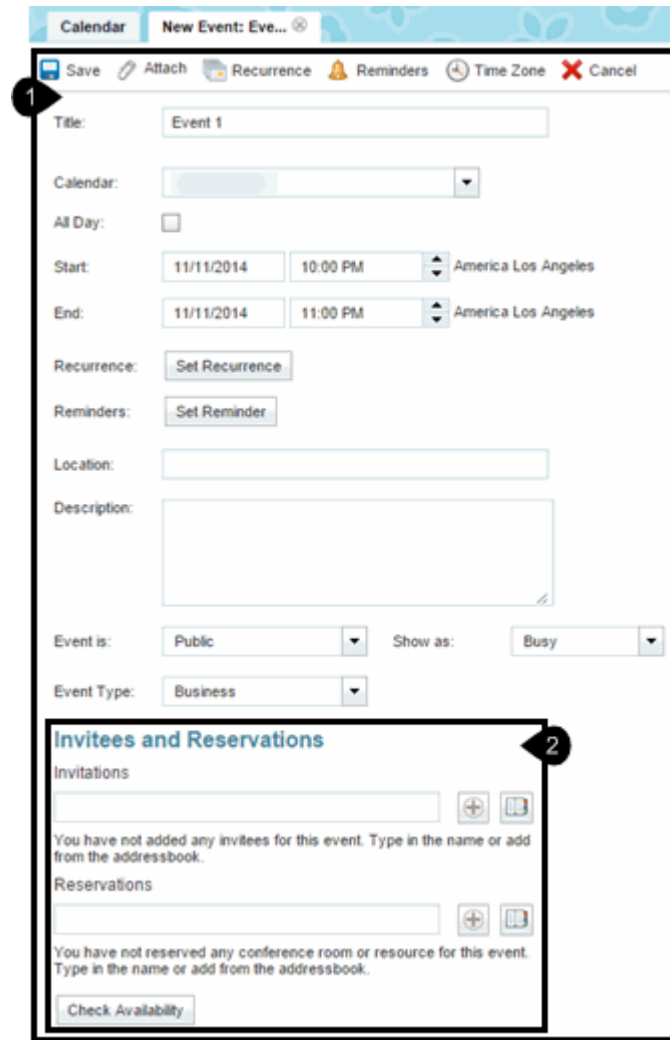


Figure 3-30 shows the location of the following widgets:

1. The RecurrenceDialog widget (`calendar.RecurrenceDialog.js`)

Figure 3-30 Convergence Calendar Recurrence Dialog Widget

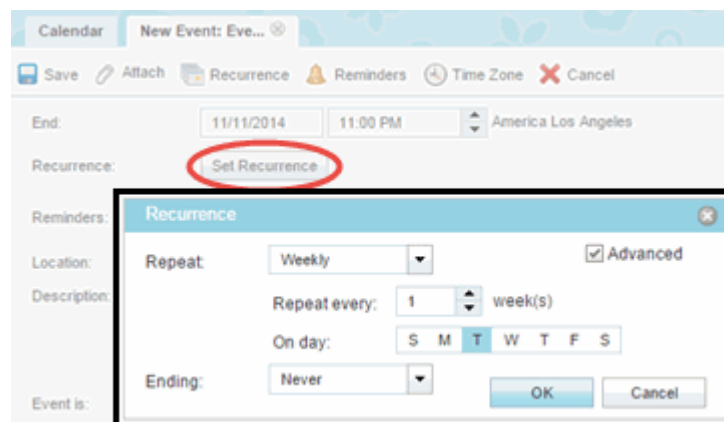


Figure 3-31 shows the location of the following widgets:

1. The CreateTaskDialog widget (`calendar.CreateTaskDialog.js`)

Figure 3-31 Convergence Calendar Create Task Dialog Widget

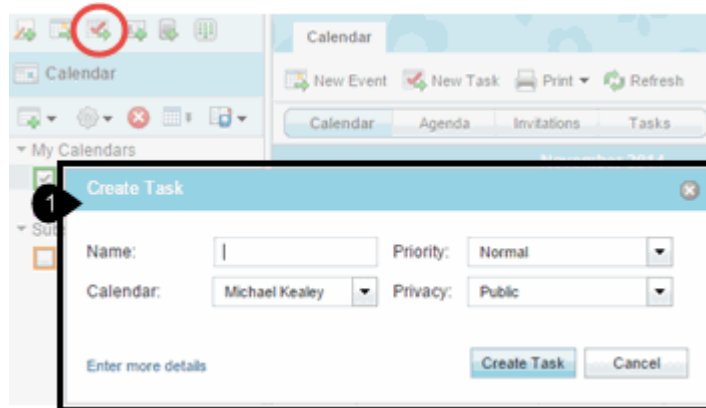


Figure 3-32 shows the location of the following widgets:

1. The TaskDetail widget (`calendar.TaskDetail.js`)

Figure 3-32 Convergence Calendar Task Detail Widget

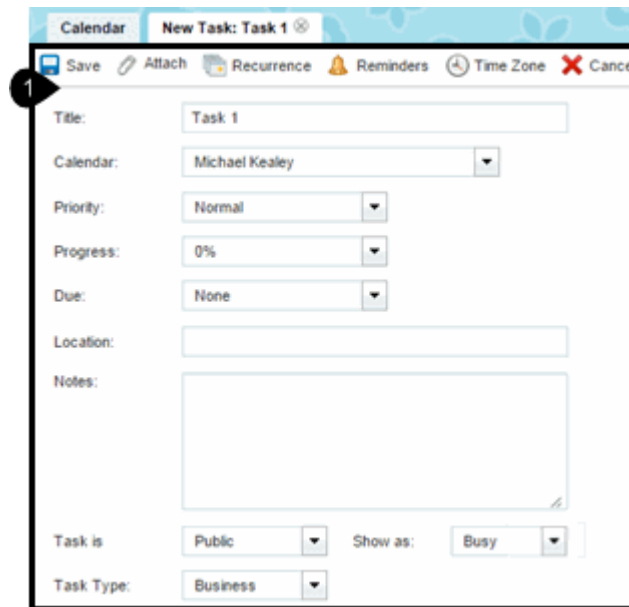


Figure 3-33 shows the location of the following widgets:

1. The EventBalloon widget (`calendar.EventBalloon.js`)

Figure 3-33 Convergence Calendar Event Balloon Widget

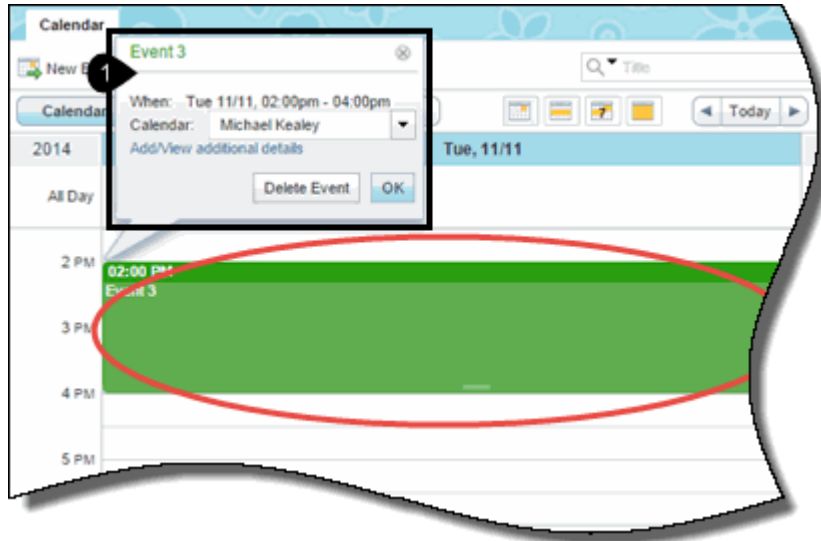


Figure 3-34 shows the location of the following widgets:

1. The ViewEvent widget (`calendar.ViewEvent.js`)

Figure 3-34 Convergence Calendar View Event Widget

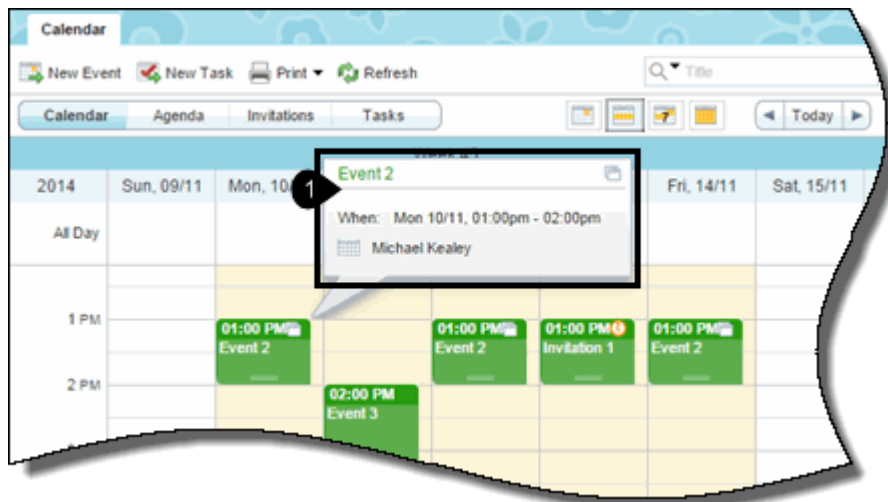


Figure 3-35 shows the location of the following widgets:

1. The QuickEventBalloon widget (`calendar.QuickEventBalloon`, in `calendar.TaskDetail.js`)

Figure 3-35 Convergence Calendar Task Detail Widget

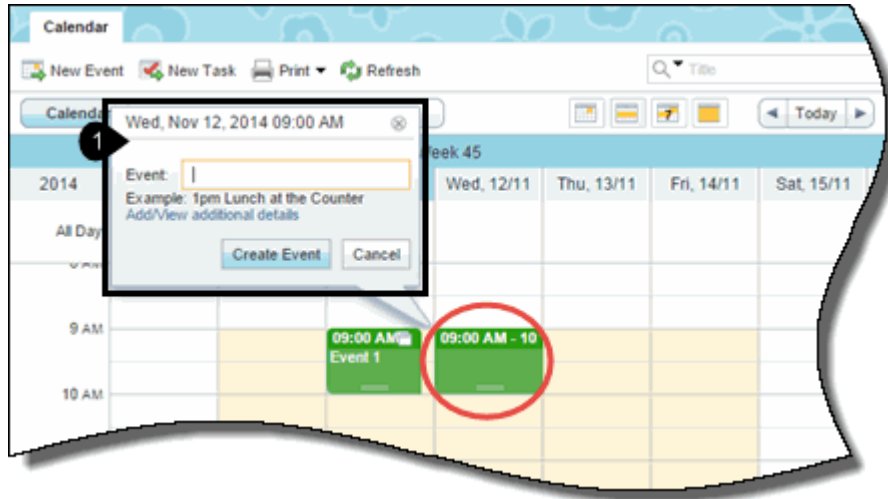


Figure 3-36 shows the location of the following widgets:

1. The Availability widget (`calendar.Availability.js`)

Figure 3-36 Convergence Calendar Availability Widget

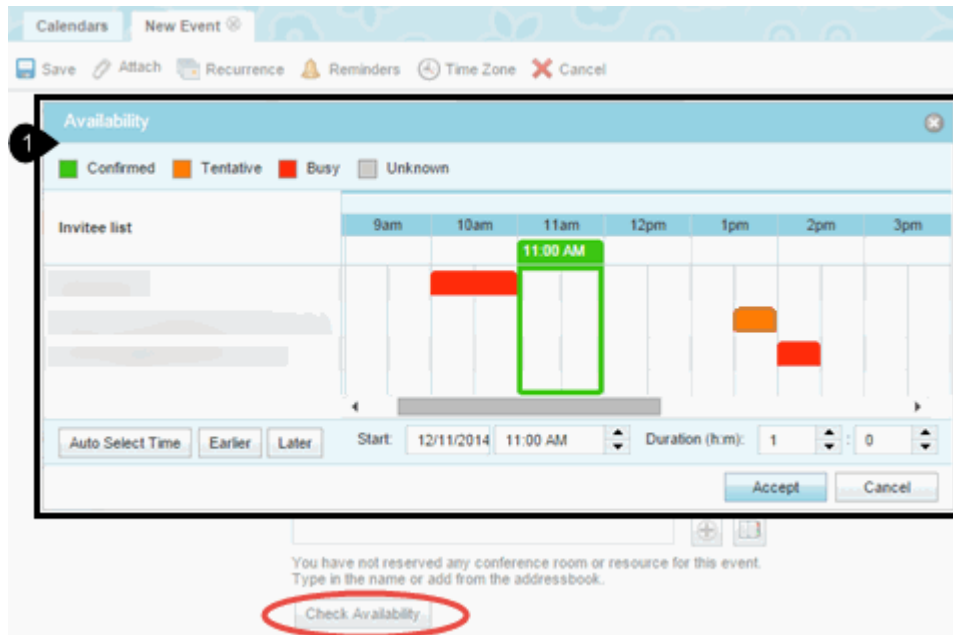


Figure 3-37 shows the location of the following widgets:

1. The NotificationDialog widget (`calendar.NotificationDialog.js`)

Figure 3-37 Convergence Calendar Notification (Reminder) Dialog Widget

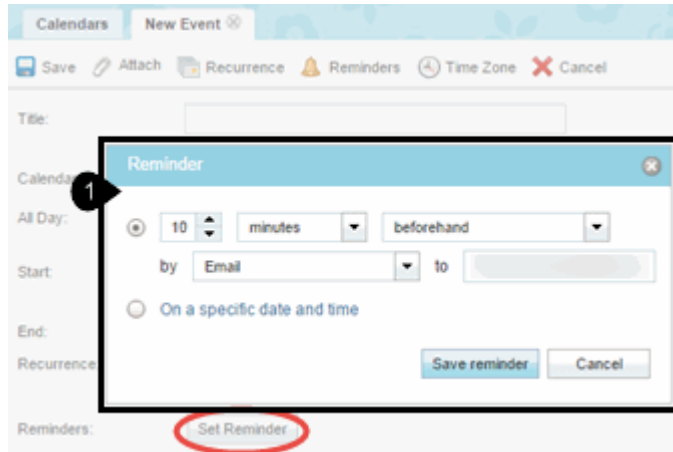


Figure 3-38 shows the location of the following widgets:

1. The PrintDialog widget (`calendar.PrintDialog.js`)

Figure 3-38 Convergence Calendar Print Dialog Widget

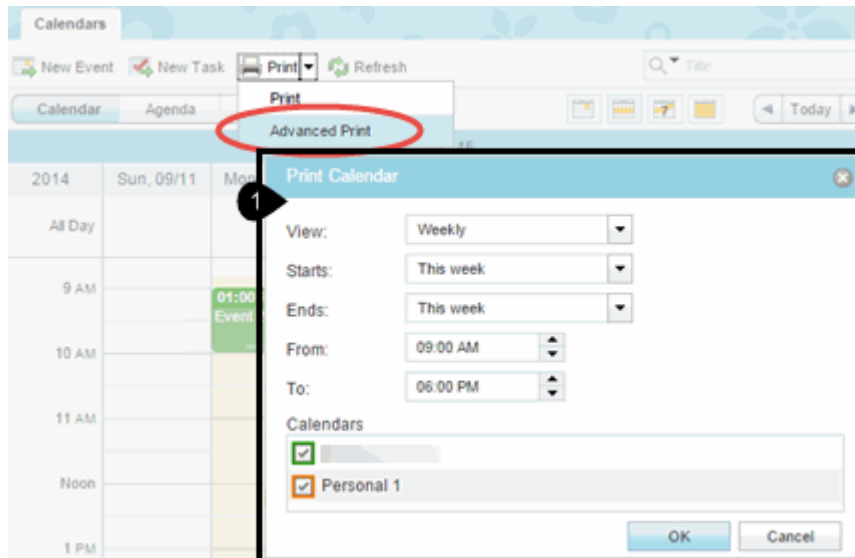


Figure 3-39 shows the location of the following widgets:

1. The Print widget (`calendar.Print.js`)

Figure 3-39 Convergence Calendar Print Widget

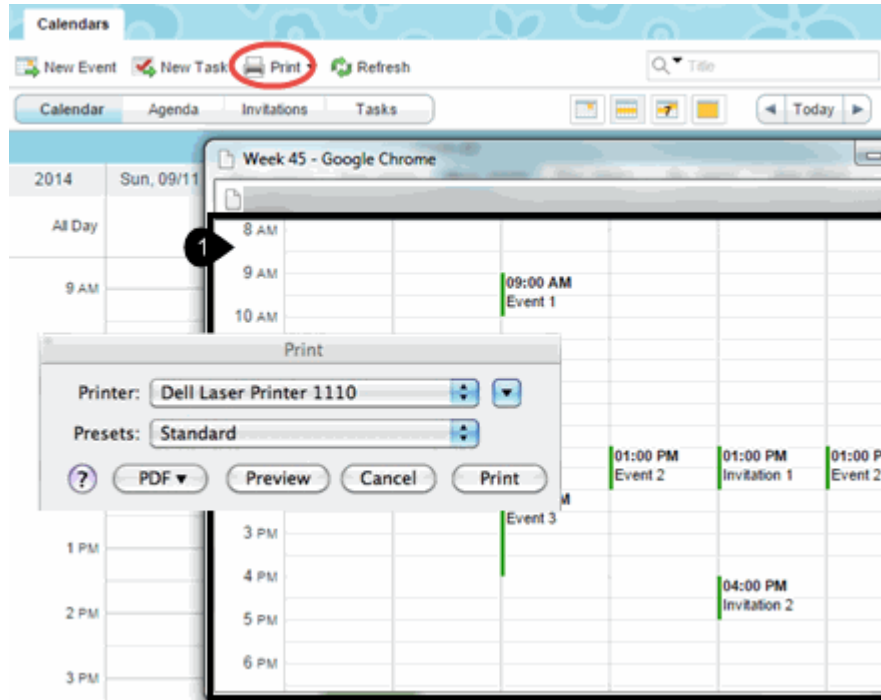


Figure 3-40 shows the location of the following widgets:

1. The TimezoneDialog widget (`calendar.TimezoneDialog.js`)

Figure 3-40 Convergence Calendar Time Zone Dialog Widget

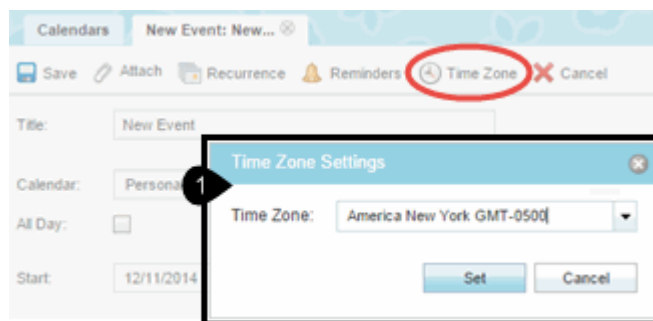


Figure 3-41 shows the location of the following widgets:

1. The ExportDialog widget (`calendar.ExportDialog.js`)

Figure 3-41 Convergence Calendar Export Dialog Widget

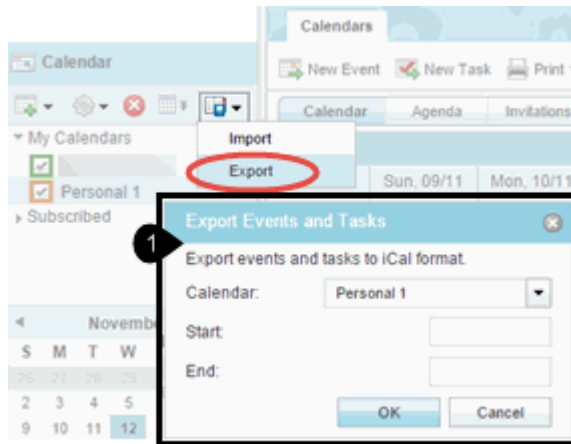


Figure 3-42 shows the location of the following widgets:

1. The ImportDialog widget (`calendar.ImportDialog.js`)

Figure 3-42 Convergence Calendar Import Dialog Widget

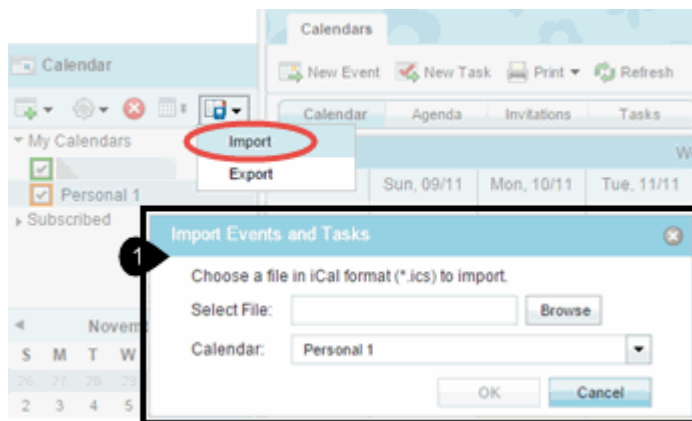
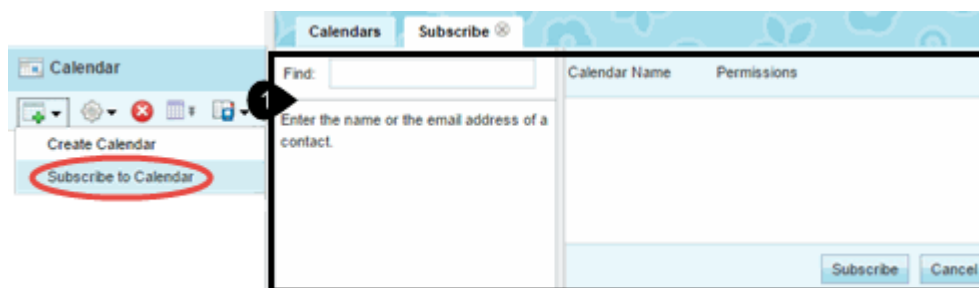


Figure 3-43 shows the location of the following widgets:

1. The Subscribe widget (`calendar.Subscribe.js`)

Figure 3-43 Convergence Calendar Subscribe Widget



Options Widgets

The Convergence Option widgets are shown in the following figures:

- [Figure 3-44](#) Common Convergence Option Widgets
- [Figure 3-45](#) Convergence Global Option General Widget
- [Figure 3-46](#) Convergence Global Option Date and Time Widget
- [Figure 3-47](#) Convergence Global Option Password Widget
- [Figure 3-48](#) Convergence Global Option Audio Alerts Widget
- [Figure 3-49](#) Convergence Mail Option General Widget
- [Figure 3-50](#) Convergence Mail Option Layout Widget
- [Figure 3-51](#) Convergence Mail Option Forwarding Widget
- [Figure 3-52](#) Convergence Mail Option Filter List Widget
- [Figure 3-53](#) Convergence Mail Option (New) Filter Widget
- [Figure 3-54](#) Convergence Mail Option Vacation Message Widget
- [Figure 3-55](#) Convergence Mail Option (Local Account) Identity Widget
- [Figure 3-56](#) Convergence Calendar Option General Widget
- [Figure 3-57](#) Convergence Calendar Option Event Widget
- [Figure 3-58](#) Convergence Calendar Option Notification Widget

[Figure 3-44](#) shows the location of the following widgets:

1. The Navigator widget (**`option.Navigator.js`**)
2. The ViewContainer widget (**`option.ViewContainer.js`**)

Figure 3-44 Common Convergence Option Widgets

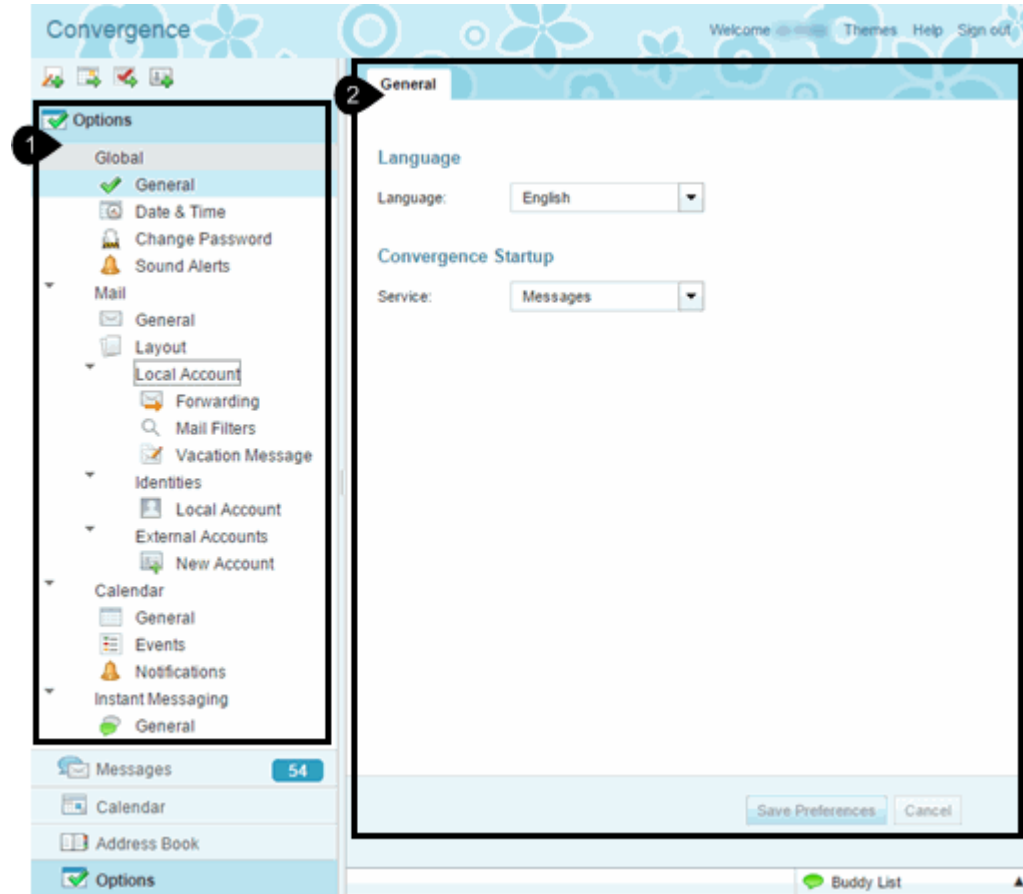


Figure 3-45 shows the location of the following widgets:

1. The General widget (`option.General.js`)

Figure 3-45 Convergence Global Option General Widget

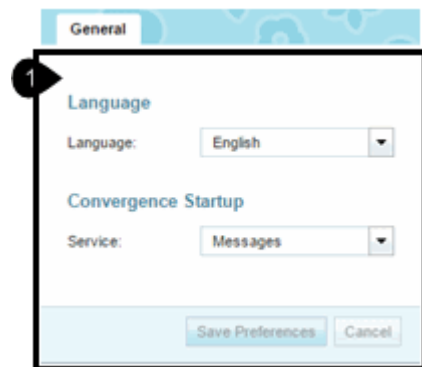


Figure 3-46 shows the location of the following widgets:

1. The DateAndTime widget (`option.DateAndTime.js`)

Figure 3-46 Convergence Global Option Date and Time Widget

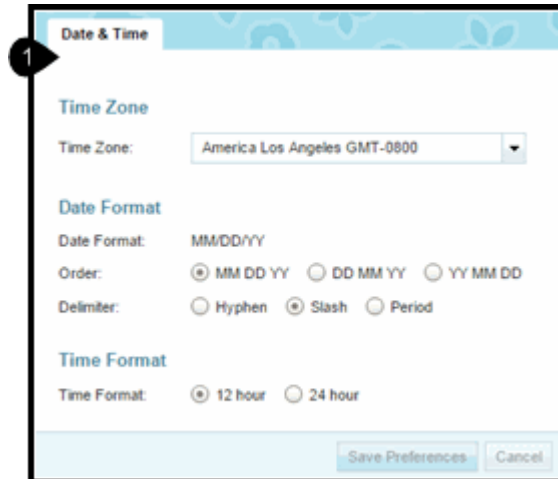


Figure 3-47 shows the location of the following widgets:

1. The Password widget (`option.Password.js`)

Figure 3-47 Convergence Global Option Password Widget

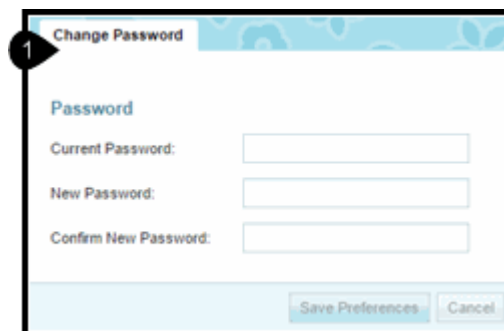


Figure 3-48 shows the location of the following widgets:

1. The AudioAlerts.Options widget (`option.AudioAlerts.Options.js`)

Figure 3-48 Convergence Global Option Audio Alerts Widget

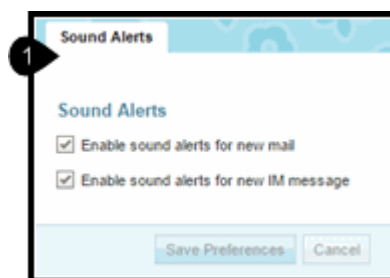


Figure 3-49 shows the location of the following widgets:

1. The mail.option.General widget (**mail.option.General.js**)

Figure 3-49 Convergence Mail Option General Widget

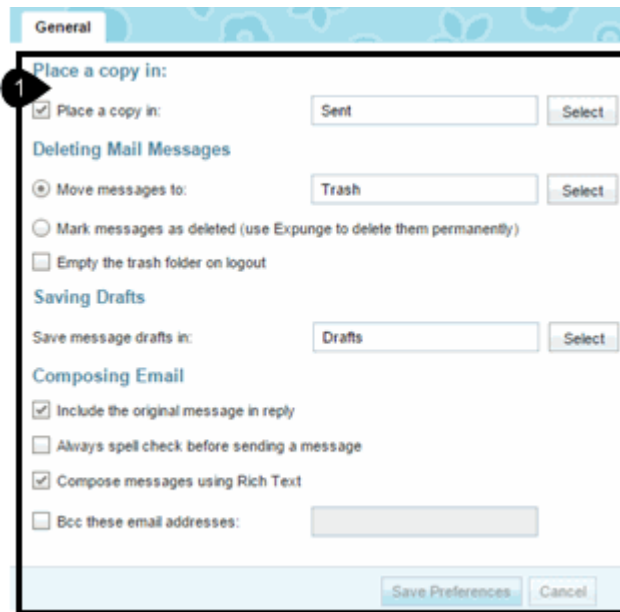


Figure 3-50 shows the location of the following widgets:

1. The mail.option.Layout widget (**mail.option.Layout.js**)

Figure 3-50 Convergence Mail Option Layout Widget

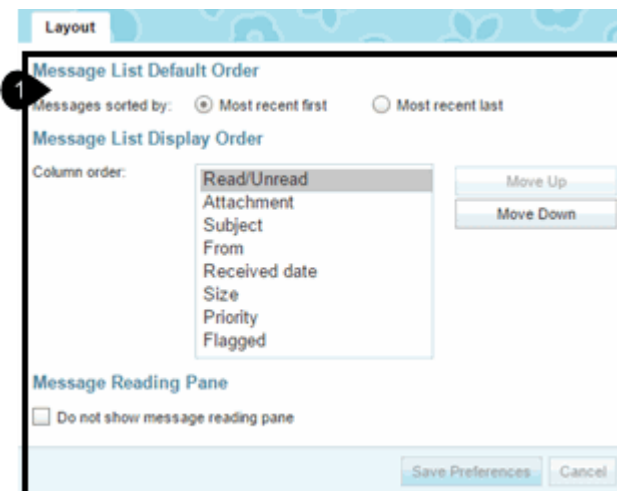


Figure 3-51 shows the location of the following widgets:

1. The mail.option.Forwarding widget (**mail.option.Forwarding.js**)

Figure 3-51 Convergence Mail Option Forwarding Widget

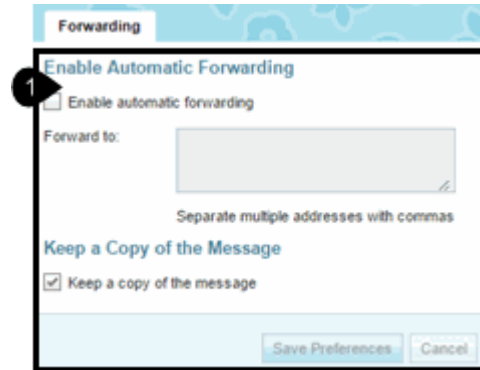


Figure 3-52 shows the location of the following widgets:

1. The mail.option.FilterList widget (`mail.option.FilterList.js`)

Figure 3-52 Convergence Mail Option Filter List Widget

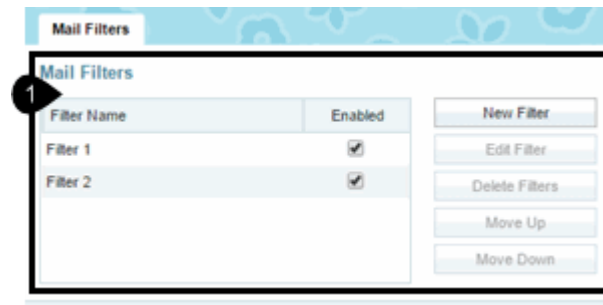


Figure 3-53 shows the location of the following widgets:

1. The mail.option.Filter widget (`mail.option.Filter.js`)

Figure 3-53 Convergence Mail Option (New) Filter Widget

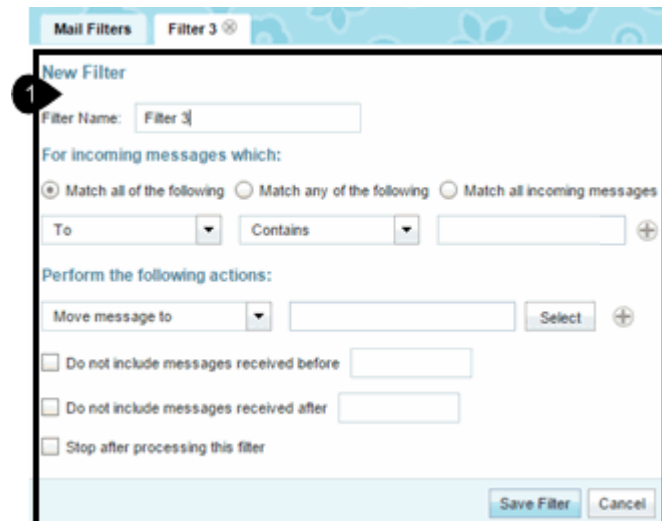
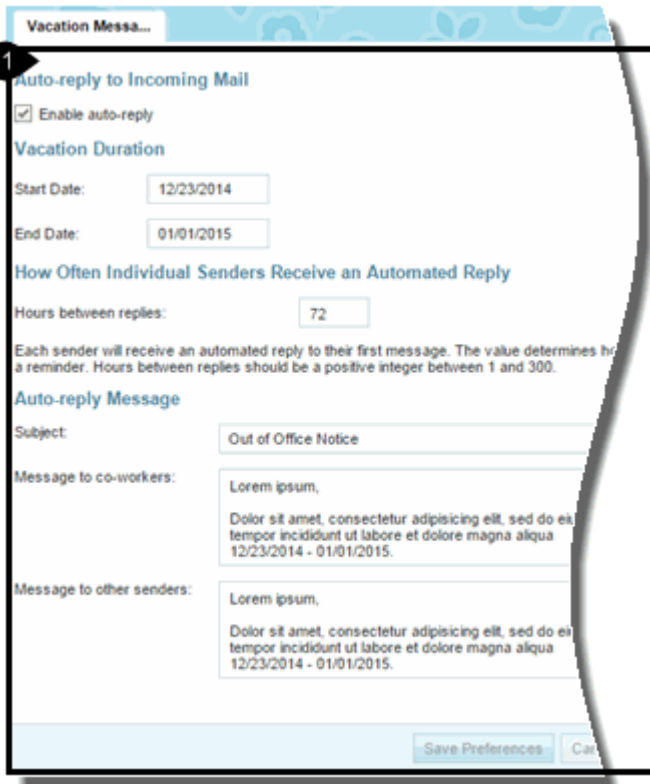


Figure 3-54 shows the location of the following widgets:

1. The mail.option.VacationMessage widget (**mail.option.VacationMessage.js**)

Figure 3-54 Convergence Mail Option Vacation Message Widget



The screenshot shows a configuration window titled "Vacation Message...". It contains the following sections and controls:

- Auto-reply to Incoming Mail**
 - Enable auto-reply
- Vacation Duration**
 - Start Date: 12/23/2014
 - End Date: 01/01/2015
- How Often Individual Senders Receive an Automated Reply**
 - Hours between replies: 72
 - Each sender will receive an automated reply to their first message. The value determines how often a reminder. Hours between replies should be a positive integer between 1 and 300.
- Auto-reply Message**
 - Subject: Out of Office Notice
 - Message to co-workers: Lorem ipsum, Dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua 12/23/2014 - 01/01/2015.
 - Message to other senders: Lorem ipsum, Dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua 12/23/2014 - 01/01/2015.

At the bottom right, there are "Save Preferences" and "Cancel" buttons.

Figure 3-55 shows the location of the following widgets:

1. The mail.option.Identity widget (**mail.option.Identity.js**)

Figure 3-55 Convergence Mail Option (Local Account) Identity Widget

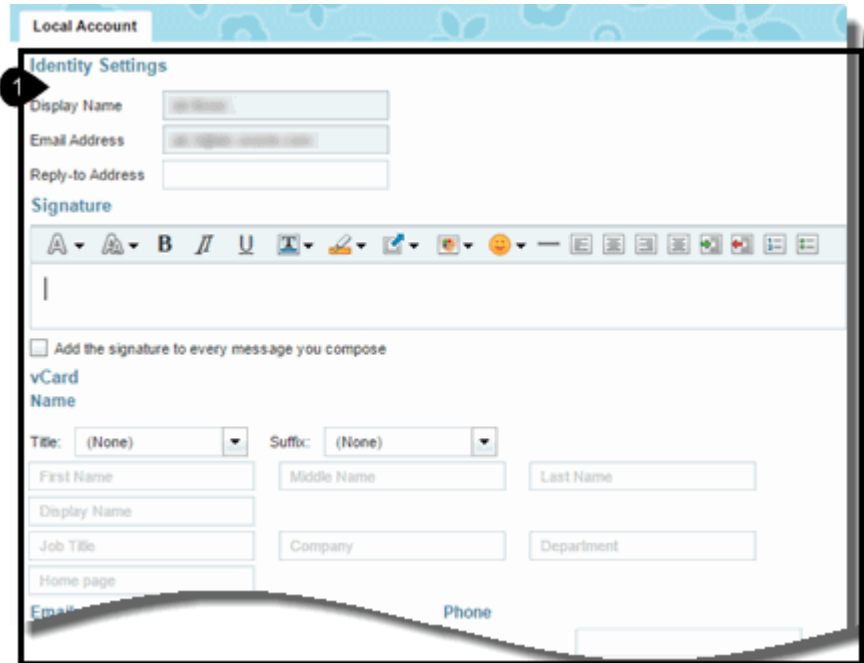


Figure 3-56 shows the location of the following widgets:

1. The calendar.option.General widget (**calendar.option.General.js**)

Figure 3-56 Convergence Calendar Option General Widget

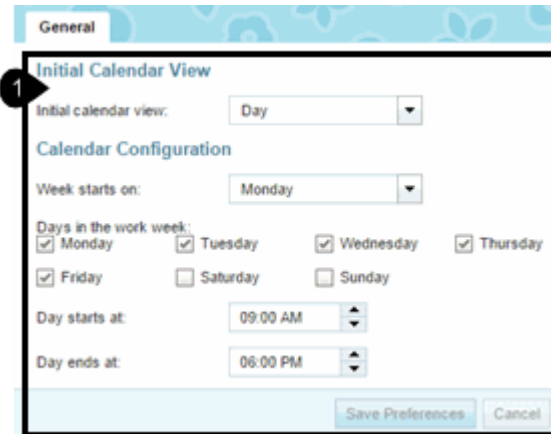


Figure 3-57 shows the location of the following widgets:

1. The calendar.option.Event widget (**calendar.option.Event.js**)

Figure 3-57 Convergence Calendar Option Event Widget

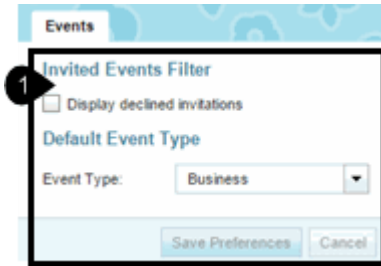
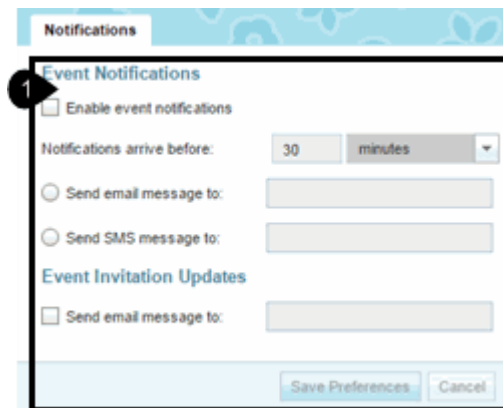


Figure 3-58 shows the location of the following widgets:

1. The calendar.option.Notification widget (`calendar.option.Notification.js`)

Figure 3-58 Convergence Calendar Option Notification Widget



4

Working with the Convergence UI

This chapter describes how to work with some elements of the Oracle Communications Convergence UI.

Customization Requirements

To perform any Convergence customization, you should have expert knowledge in dojo and knowledge of UI customization. See "[Technical Overview](#)" for more information on general Convergence customization.

Nearly all customization examples require the same setup and preparation.

- You start by verifying that the `c11n_Home` directory exists. If it does not exist, create it. See "[Customization Directory Structure](#)" for more information.
- Next, you verify that customization is enabled in Convergence. If customization is disabled, enable it. See "[Enabling Customization in the Convergence Server](#)" for more information.
- When you have completed your customization, you must restart the Oracle WebLogic Server for Convergence. See *Convergence System Administrator's Guide* for more information.

Theme Customization Features

Theme customization:

- allows for a single template to be used by multiple themes
- maintains a consistent way to define CSS declaration
- reduces the dependency on CSS selector changes by isolating those changes in the template file
- uses APIs instead of writing directly into the `themes.attr("store")`, creating streamlined updates and maintenance.
- maintains customization between patches
- reduces dependence on requiring all values to be specified in the `theme.json` for each theme
- reduces repetitive CSS values. By adding parent information when defining the theme in `customize.js` file, you do not 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.

Default Themes Included with Convergence

Themes included with Convergence are based on a template called `theme_basic`.

[Figure 4-1](#) illustrates the following themes, which are available with Convergence by default:

1. `theme_blue`
2. `theme_orange`
3. `theme_dark_blue`
4. `theme_light_blue`
5. `theme_grey`
6. `theme_yellow`
7. `theme_green`
8. `theme_teal`
9. `theme_pink`
10. `theme_butterfly`
11. `theme_teal_ocean`
12. `theme_pink_hearts`
13. `theme_blue_cherry`
14. `theme_starry`
15. `theme_altair`

Figure 4-1 Default Themes Included with Convergence



To customize a theme:

1. CSS declarations are housed in a template file (**/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: "#89AFD0",
mastheadColor: "#FFFFFF",
```

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

```
.Banner {
background: "#89AFD0";
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 "[About the Basic Theme](#)" 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 do not 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 style sheet.
9. Refresh layout.

About the Basic Theme

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 among 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 4-1 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.	<i>c11n_Home/themes/basic/theme.json</i>
theme_basic	parentTheme of all included visible themes. The theme_basic is made up of the theme.json file and the template.css file.	For Oracle WebLogic Server: <i>data_directory/web-src/client/iwc_static/layout/themes/basic/</i>
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.	For Oracle WebLogic Server: <i>data_directory/web-src/client/iwc_static/layout/themes/basic/</i>

The basic theme has the following definition:

```
{
  name: 'theme_basic',
  configPath: "themes/basic/theme.json",
  thumbnailColor: "#FFFFFF",
  visible: false
}
```

Basic Theme Properties

[Table 4-2](#) describes the variables and default values for **theme_basic**.

Table 4-2 Basic Theme Properties

Property Name	Value	Notes
warningBorderColor	#FEBE56	border color style for NotificationPanel-Container.
warningBackground	#FEEFDO	background style for NotificationPanel-Container.
warningTextColor	#282828	font color for NotificationPanel-Container.
invalidFieldBorderColor	#FEBE56	border color style for inputs in invalid state.
invalidFieldBackground	#FFFFFF	background style for inputs in invalid state.
invalidFieldTextColor	#282828	font color used for inputs in invalid state.
growlColor	#282828	color used for the notification growl.

Table 4-2 (Cont.) Basic Theme Properties

Property Name	Value	Notes
growlBorderColor	#FEBE56	border color used for the notification growl.
growlContentBackground	#FEEFD0	background style for the notification growl.
layoutBackground	transparent	Background for the entire application. Example usage exists in new themes provided.
tallTitlebarBackground	#CCCCCC	For title bars which span multiple lines background style
tallTitlebarColor	#FFFFFF	For title bars which span multiple lines font color style
focusedItemBackground	#E8E8E8	background color to be used when an item is in focus
focusedItemColor	#000000	font color to be used when an item is in focus.
calendarDateBorderColor	#CCCCCC	Calendar View
calendarDateTodayBorderColor	#CCCCCC	Calendar View
calendarDateTodayHeaderBackground	#E8E8E8	Calendar View
calendarDateTodayHeaderColor	#333333	Calendar View
calendarDateTodayBackground	#E8E8E8	Calendar View
calendarDateTodayColor	#333333	Calendar View
calendarDateSelectedHeaderBackground	#A6A6A6	Calendar View
calendarDateSelectedHeaderColor	#FFFFFF	Calendar View
calendarDateSelectedBackground	#FFFFFF	Calendar View
calendarDateSelectedColor	#333333	Calendar View
miniCalendarDateTodayBorderColor	#CCCCCC	Mini Calendar View
miniCalendarDateTodayBackground	#E8E8E8	Mini Calendar View
miniCalendarDateTodayColor	#FFFFFF	Mini Calendar View
miniCalendarDateSelectedBackground	#A6A6A6	Mini Calendar View
miniCalendarDateSelectedColor	#FFFFFF	Mini Calendar View
serviceNavigatorUnreadCountColor	#333333	Service Menu Mail Unread Count font color
taskbarButtonHighlightedBackground	#FEEFD0	Taskbar IM

Table 4-2 (Cont.) Basic Theme Properties

Property Name	Value	Notes
taskbarButtonHighlightedBorderColor	#FEBE56	Taskbar IM
taskbarButtonHighlightedColor	#282828	Taskbar IM
contactNameColor	#7EAF73	Currently, not used
userNameColor	#68B4CE	Currently, not used
messageTextColor	#434343	Currently, not used
messageTimeStampColor	#A9A9A9	Currently, not used
tileBackground	transparent url ('themes/basic/images/Tile_base.png') repeat-x top left	Used for SMS Feature
tileHoverBackground	transparent url ('themes/basic/images/Tile_selected.png') repeat-x top left	Used for SMS Feature
tileSelectedBackground	transparent url ('themes/basic/images/Tile_selected.png') repeat-x top left	Used for SMS Feature
tileColor	#424242	Used for SMS Feature
tileHoverColor	#FFF	Used for SMS Feature
tileSelectedColor	#FFF	Used for SMS Feature
smsSynopsisColor	#A9A9A9	SMS Feature
smsTextContrastColor	#666666	SMS Feature
smsThreadHeadersColor	#666	SMS Feature
smsThreadHeadersSortedColor	#FFFFFF	SMS Feature
smsMessageHeaderColor	#FFF	SMS Feature
smsThreadHeadersBackground	#CCCCCC	SMS Feature
smsThreadHeadersSortedBackground	#CCCCCC	SMS Feature
smsMessageHeaderBackground	#666666	SMS Feature
smsMessageColor	#434343	SMS Feature
smsMessageHoverColor	#000	SMS Feature
smsNewMessageBackground	#CCCCCC	SMS Feature

JSON Reference for Customizing Themes

Theme.json supports the following values:

- 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 Theme.json File

```
{
  // required
  // A path to a css file where these values will be inserted 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 online
  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"
}

```

Customizing Layout HTML Pages

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

- Because the customization 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 therefore be recorded so they can be re-applied if required.

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

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

- [Creating and Customizing login.html](#)
- [Creating and Customizing login.html in a Hosted Domain](#)
- [Modifying the Login Page Welcome Message](#)
- [Creating and Customizing main.html](#)
- [Configuring the Per-Domain Main Page](#)
- [Creating and Customizing calendar.html](#)

Creating and Customizing login.html

To create and customize the login.html page for all domains:

1. Ensure that *c11n_Home* exists. If it does not, create it.

2. Copy the **allDomain/layout** directory from **c11n_sample** into **c11n_Home**.
3. Run the **iwcadmin** command 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"
```

Creating and Customizing 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 **c11n_Home/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"
```

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.

Modifying the Login Page Welcome Message

To modify the login page welcome message:

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_Home** directory. For example, copy the following code in the Oracle WebLogic Server location:

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

For Oracle WebLogic Server, copy the code in the **data_directory/web-src/client/iwc_static/js/iwc/c11n_sample/allDomain/layout/login.html** to **c11n_Home/allDomain/layout/login.html**

Note:

The function **loadC11nResources()** loads **resources.js** from **c11n_Home/allDomain/nls** and uses the new **login_welcome_msg** string.

2. Copy **resources.js** from **c11n_sample/allDomain/nls/** to **c11n_Home/allDomain/nls/**.
 - In **Language/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_HomeI/config.js` to set `i18nEnabled`.
3. To hide the **login_welcome_msg** string, add the following style to `c11n_HomeI/allDomain/layout/login.html`:


```
<style type="text/css">
  #copyright, #welcomeMsg {
    display: none;
  }
</style>
```
 4. Restart the Oracle WebLogic Server and clear the browser cache to see the change.

Creating and Customizing main.html

To create and customize the `main.html` page for all domains:

1. Copy the sample from `c11n_sample/allDomain/layout` to `c11n_HomeI/allDomain/layout` directory.
2. Run the **iwadmin** command to set the new layout file location. For example, to set `allDomain` to use the new **main.html**:

```
iwadmin -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 `c11n_HomeI/allDomain/layout` and restart the Oracle WebLogic Server.

Configuring the Per-Domain Main Page

To configure a per-domain main page for each of your domains:

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=<example_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: set the **sunUCExtendedClientPrefs** attribute of **sunUCPreferences** as in the following example:

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

- Restart the Oracle WebLogic Server.

When a user logs into Convergence, 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.

Creating and Customizing calendar.html

You can customize an anonymous **calendar.html** page for a specific domain or for all domains. The following steps create an anonymous **calendar.html** page for allDomain, set allDomain to use the calendar page, and then customize **calendar.html** to display a new, user-defined theme.

- Copy **calendar.html** from **c11n_sample/allDomain/layout** to the **c11n_Home/allDomain/layout** directory.
- Use the **iwcadmin** command to set the anonymous calendar for allDomain to use the new **calendar.html** file. Enter:

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

- Open **calendar.html** for editing and enter the name of the domain that is providing Convergence:
 - Open **calendar.html**, and locate the line:


```
var userPrefsGeneralUserDomain = "in.example_com.com";
```
 - Replace **in.example_com.com** with the domain for Convergence, for example:


```
var userPrefsGeneralUserDomain = "Cv3example_domain.com";
```
- Do any of the following:
 - Set the default theme in an anonymous calendar.
See "[Setting a Theme in an Anonymous Calendar](#)" for more information.
 - Set the default date format in an anonymous calendar.
See "[Customizing Anonymous Calendar Date and Time Format](#)" for more information.
- Refresh your browser.
- Open an anonymous calendar to see your changes.

Setting a Theme in an Anonymous Calendar

To set a theme in an anonymous calendar:

- In **c11n_Home/allDomain/layout**, open **calendar.html** for editing.
- Set all unwanted themes to be hidden.
 - Locate the following line:


```
iwc.themes.loadSelectedItem();
```
 - Add a line above **iwc.themes.loadSelectedItem()** for each default theme you want to hide, for example:

```
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");
iwc.api.hideTheme("theme_teal");
iwc.api.hideTheme("theme_pink");
iwc.api.hideTheme("theme_butterfly");
iwc.api.hideTheme("theme_teal_ocean");
iwc.api.hideTheme("theme_pink_hearts");
iwc.api.hideTheme("theme_blue_cheery");
iwc.api.hideTheme("theme_starry");

```

In the above example, every default theme is hidden.

3. Below the entries for the hidden themes, add the following code for a new custom theme. For example, to add the new theme **theme_dark_gamboge**, enter:

```

iwc.api.addTheme({
  name: "theme_dark_gamboge", // The name must be unique
  parentTheme: "theme_basic", // Must be one of the available basic themes
  configPath: "../c11n/allDomain/themes/purple/theme.json",
  thumbnailColor: "#C290D6",
  visible: true
});

```

Because all other themes are hidden, this custom theme is the only available theme and appears by default.

4. Save **calendar.html**.

To set one of the default themes for an anonymous calendar, omit the desired theme from the list of hidden theme and do not include the code to add a new theme.

If the banner is smaller than before, enlarge it by adding a CSS style to the head section of **calendar.html**. Open **calendar.html** for editing and add 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>

```

Customizing Anonymous Calendar Date and Time Format

You can customize the date format for an anonymous calendar:

1. In **c11n_Home/allDomain/layout**, open **calendar.html** for editing.
2. Specify the date format:
 - a. Uncomment the following line:

```
iwc.api.setGeneralUserPreference('dateformat', 'D/M/Y');
```

- b. From the line you uncommented, replace **'D/M/Y'** with one of the following available values:
 - **'M/D/Y'**
 - **'D/M/Y'**
 - **'Y/M/D'**

Where **Y** is the year, **M** is the month, and **D** is the day. The year, month, and day are displayed in two-digit format in Convergence.

3. Specify the delimiter for the date format:

a. Uncomment the following line:

```
iwc.api.setGeneralUserPreference('datedelimiter', '-');
```

b. From the line you uncommented, replace '-' with one of the following available values:

- '-'
- '.'
- '/'

4. Specify the time format:

a. Uncomment the following line:

```
iwc.api.setGeneralUserPreference('timeformat', '24');
```

b. From the line you uncommented, replace '24' with one of the following available values:

- '12'
- '24'

5. Save **calendar.html**.

Integrating Third-Party Applications

Convergence provides access to the following back-end services: mail, address book, and calendar. 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.

The customization sample directory includes a reference implementation for a third-party service called HelloConvergence. See "[Integrating HelloConvergence into Convergence](#)" for more information.

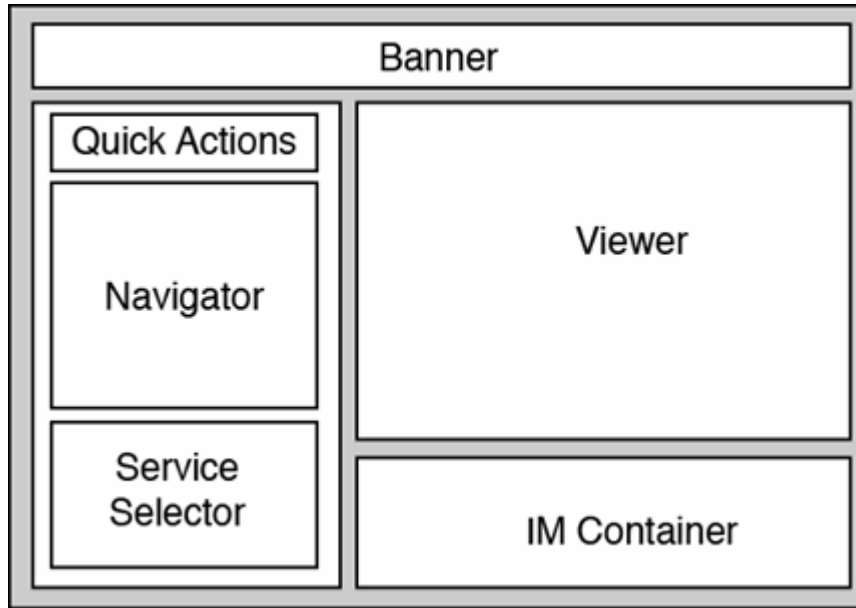
To add a third-party application (new service), you normally

- Create a menu button to select the service
- Create a new service navigator widget
- Create a new service viewer widget

You must have experience with dojo to create and customize widgets in Convergence. See "[Technical Overview](#)" for general information about customizing widgets.

[Figure 4-2](#) shows the Convergence UI layout, including the regions of the UI that can be used by an integrated third-party application.

Figure 4-2 Convergence UI Layout



Integrating HelloConvergence into Convergence

The customization sample directory includes a reference implementation for a third-party service called HelloConvergence. Use this reference implementation as a starting point for adding your own service to Convergence.

To integrate the third-party application called HelloConvergence into Convergence:

1. Verify that the `c11n_Home` directory exists. If `c11n_Home` does not exist, create it.
2. Verify that customization is enabled in Convergence. If customization is disabled, enable it.
3. In `c11n_Home`, verify that the following directories exist. If they do not exist, create them:

```
/c11n_Home/allDomain/js/service/  
/c11n_Home/allDomain/js/widget/helloConvergence/
```

4. In `c11n_Home`, verify that `config.js` exists. If it does not exist, create it.
5. Modify `config.js` so that it enables JavaScript customization (`jsEnabled: true`) and i18n customization (`i18nEnabled: true`) across all domains (`module: "allDomain"`).

```
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  
  
    }  
}
```

6. Create the following files:

- In `c11n_Home/allDomain/js/service`, create **HelloConvergence.js**
- In `c11n_Home/allDomain/js/widget/helloConvergence`, create **Navigator.js**
- In `c11n_Home/allDomain/js/widget/helloConvergence`, create **ViewerContainer.js**

7. In `c11n_Home/allDomain/js`, create or modify **customize.js** to include the following code:

```
// 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);
```

8. Modify **HelloConvergence.js** to include the following code:

```
dojo.provide("c11n.allDomain.js.service.HelloConvergence");

dojo.require("iwc.api");
dojo.require("iwc.service.ServiceBase");

dojo.require("c11n.allDomain.js.widget.helloConvergence.Navigator");
dojo.require("c11n.allDomain.js.widget.helloConvergence.ViewerContainer");

dojo.declare("c11n.allDomain.js.service.HelloConvergence",
    [iwc.service.ServiceBase],
    {
        // new properties
    }
);
```

9. Modify **Navigator.js** to include the following code:

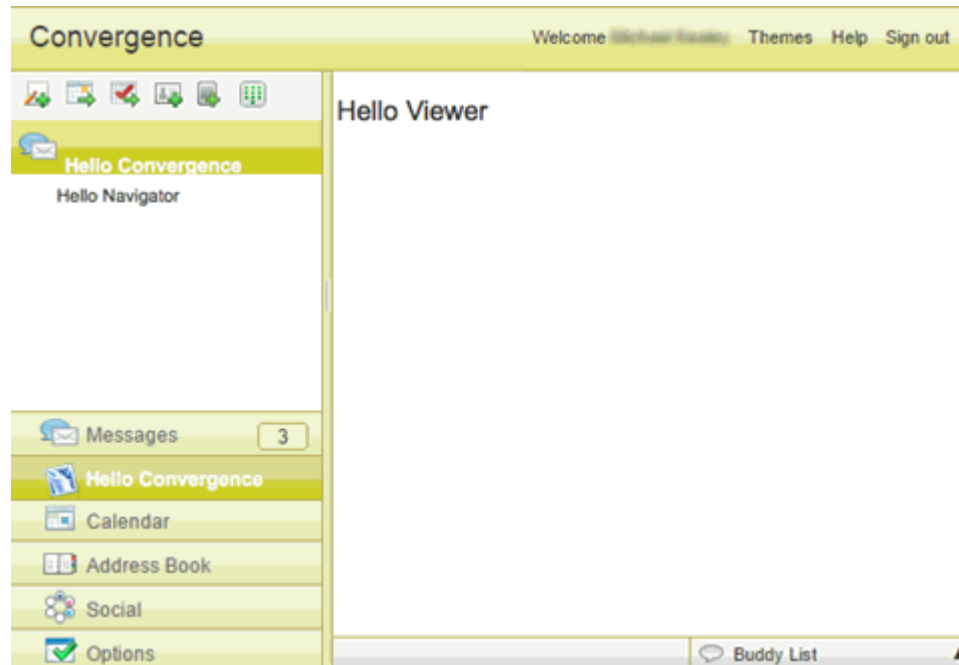
```
dojo.declare(
    "c11n.allDomain.js.widget.helloConvergence.Navigator",
    [ dijit.layout._LayoutWidget, dijit._Templated ],
    {
        // new properties
    }
);
```

10. Modify **ViewerContainer.js** to include the following code"

```
dojo.declare(
    "c11n.allDomain.js.widget.helloConvergence.ViewerContainer",
    [ dijit.layout._LayoutWidget, dijit._Templated ],
    {
        // new properties
    }
);
```

Figure 4-3 shows the HelloConvergence service in Convergence.

Figure 4-3 HelloConvergence Service Integrated into Convergence



About Adding a New Language

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.

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 `c11n_HomeIDomainInls` subdirectory. *Domain* is the domain where the customized languages will be available. Use the **allDomain** directory to apply to all domains in your deployment.

Adding a New Language in Convergence

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

1. Verify that the `c11n_Home` directory exists. If `c11n_Home` does not exist, create it.
2. Verify that customization is enabled in Convergence. If customization is disabled, enable it.
3. In `c11n_Home`, verify that the configuration file **config.js** exists. If it does not exist, create it. Edit **config.js** with the following:
 - Set the **i18nEnabled** flag to **true**
4. Create the following directory structure: `c11n_HomeIDomainInlsILanguage`.

where *Domain* is the name of the domain in which the new language will be available, and where *Language* is the subdirectory in which the new language's resource file is located. For example, to add Slovak as a new language to all domains: *c11n_Home/allDomain/nls/sk*.

5. In *c11n_Home/Domain/nls/Language*, create a default resource file named **resources.js**.

Since a new language requires the entire resource file translation, you can copy the Convergence English resource file (**iw_c_static/iw_c/i18n/nls/resources.js**) to begin with. 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.

The directory structure should be as shown in the following example. In this example, the language "sk" is added to the **allDomain** directory. Thus, it applies to all domains.

```
c11n_Home/allDomain/nls/resources.js (Default resources.js file for all customized
languages in this domain.)
c11n_Home/allDomain/nls/sk/resources.js (This resources.js file contains the
localization required for the newly added language, "sk".)
```

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

Adding 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 list of supported dojo languages is in the following directory: **iw_c_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 "[About Adding a New Language](#)" but with the following additions.

To customize the localized calendar:

1. Copy the English language version from **iw_c_static/js/dojotoolkit/dojo/cldr/nls/en/gregorian.js** to **nls/vi/gregorian.js**.
2. Translate the **nls/vi/gregorian.js** to the localized language (Vietnamese in this example).
3. Make sure you also have a **resources.js** file in the **nls** directory. For example:

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

4. Make sure the above file is called from *c11n_Home/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");
```

5. Restart the Oracle WebLogic server and clear the browser cache to view the change.

Sample Custom l10n Resource File

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

```
{
  compose_tab: "Soan thu",
  last: ""
}
```

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

Adding a Label for the New Language to the Global Options Language Menu

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

The Options Global **General** tab 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.

To add a label for a new language in the Options Global **General** tab:

1. Verify that the *c11n_Home* directory exists. If *c11n_Home* does not exist, create it.
2. Verify that customization is enabled in Convergence. If customization is disabled, enable it.
3. In *c11n_Home*, edit **config.js** and set the **jsEnabled** flag to **true**.
4. Create the following directory structure:

```
c11n_Home/Domain/js/widget/option
```

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

5. In *c11n_Home/Domain/js*, create or modify **customize.js** and uncomment the following line:

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

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

At run time, this file is responsible for loading the javascript customizations (the custom widget files) in this subdirectory.

6. 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

```
c11n_Home/Domain/js/widget/option/General.js
```

7. 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 I10n 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."
 - a. Edit "*vi*" and "*Vietnamese*", replacing them with your language directory and name. Use UTF-8 for double-byte characters.
 - b. Add additional **this.language.addOption** lines to add additional languages.

Adding a Label for the New Language to the Convergence Login Page

Figure 4-4 shows the default Convergence login page. The supported languages are listed in the drop-down menu at the bottom of the page.

Figure 4-4 Convergence Login Page with List of Supported Languages



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.

See "[Customizing Layout HTML Pages](#)" for information about adding a new label to the login page.

Setting 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 the **customize.js** file in *c11n_Home/allDomain/js* directory.

```
// 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");
*/
```

5

Convergence UI Customization Examples

This chapter provides several examples for customizing the UI in Oracle Communications Convergence.

Customization Requirements

To perform any Convergence customization, you should have expert knowledge in dojo and knowledge of UI customization. For more information on general Convergence customization, see "[Technical Overview](#)".

Nearly all customization examples require the same setup and preparation.

- You start by verifying that the `c11n_Home` directory exists. If it does not exist, create it. See "[Customization Directory Structure](#)" for more information.
- Next, you verify that customization is enabled in Convergence. If customization is disabled, enable it. See "[Enabling Customization in the Convergence Server](#)" for more information.
- When you have completed your customization, you must restart the Oracle WebLogic Server for Convergence. See *Convergence System Administrator's Guide* for more information.

Modifying 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. Verify that the `c11n_Home` directory exists. If `c11n_Home` does not exist, create it.
2. Verify that customization is enabled in Convergence. If customization is disabled, enable it.
3. In `c11n_Home`, verify that **config.js** exists. If it does not exist, create it.
4. Modify **config.js** so that it enables theme customization (**themeEnabled: true**), i18n customization (**i18nEnabled: true**), and Javascript customization (**jsEnabled: true**) across all domains (module: "**allDomain**").

The following example shows a **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
    },
}

```

5. In `c11n_Home/allDomain/themes`, create or modify **customize.js** to include the following code:

```
iwc.api.modifyThemeValues("theme_blue", "../c11n/allDomain/themes/blue/theme.json");
```

6. In `c11n_Home/allDomain/themes/blue`, edit **theme.json** with the location of the logo, the logo's width, and height:

```

{
  logoBackground: 'transparent url("../c11n/allDomain/themes/blue/images/logo.png") no-repeat center center',
  logoWidth: "180px"
  mastheadHeight: "40px"
}

```

7. In `/themes/blue/images/`, add a new **logo.png** file with black background
8. Restart the Oracle WebLogic Server and clear the browser cache to see the changes.

The blue theme now contains a logo with a black background.

Hiding a Single Theme

To hide the blue theme, edit the `/themes/customize.js` file to include the following code:

```
iwc.api.hideTheme("theme_blue");
```

When you remove the blue theme, the theme selector displays all the included themes except for the blue theme:

Creating a New Theme

The Convergence UI banner uses the `iwc.api.addTheme` to add themes to the theme selector. You can add themes to the theme selector.

The following example adds a purple theme to the theme selector:

1. Verify that the `c11n_Home` directory exists. If `c11n_Home` does not exist, create it.
2. Verify that customization is enabled in Convergence. If customization is disabled, enable it.
3. In `c11n_Home`, verify that the following directories exist. If they do not exist, create them:


```

/c11n_Home/allDomain/nls/
/c11n_Home/allDomain/themes/purple/

```
4. In `c11n_Home`, verify that **config.js** exists. If it does not exist, create it.
5. Modify **config.js** so that it enables theme customization (**themeEnabled: true**), i18n customization (**i18nEnabled: true**), and Javascript customization (**jsEnabled: true**) across all domains (module: **"allDomain"**).

The following example shows a **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
    },
}

```

6. In `c11n_Home/allDomain/themes`, create or modify **customize.js** to include the following code:

```

// Adding a new theme
iwc.api.addTheme({
    name: 'theme_purple', // name must be unique
    parentTheme: "theme_basic", // must use a theme_basic theme
    configPath: "../c11n/allDomain/themes/purple/theme.json",
    thumbnailColor: "#C290D6"
    visible: true
});

```

You can also add a background image, you do so by adding a link to that image in the **thumbnailColor** parameter. For example:

```

// Adding a new theme
iwc.api.addTheme({
    name: 'theme_purple', // name must be unique
    parentTheme: "theme_basic", // must use a theme_basic theme
    configPath: "../c11n/allDomain/themes/purple/theme.json",
    thumbnailColor: '#C290D6 url("../c11n/allDomain/themes/purple/images/logo.png")'
    visible: true
});

```

7. In `c11n_Home/allDomain/nls`, add the code for the customized theme to **resources.js** above `last: ""`.

```

{
    ...
    theme_purple : "Purple",
    last: ""
}

```

8. Restart the Oracle WebLogic Server and clear the browser cache to see the changes.
The new Purple theme appears as the last theme in the theme selector.

Making a Newly Created Theme the Default

This example assumes you have already created a new theme called Purple. See "[Creating a New Theme](#)" for more information.

To make the Purple theme the default theme in Convergence:

1. Use the **iwcadmin** command to set the Purple theme (`theme_purple`) as the default:

```
iwcadmin -o user.common.theme -v theme_purple
```

For more information about the user preferences or the Convergence command-line utility, see *Convergence System Administrator's Guide*.

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

The default theme appears by default for all users who have not selected a specific theme.

Adding 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. Verify that the *c11n_Home* directory exists. If *c11n_Home* does not exist, create it.
2. Verify that customization is enabled in Convergence. If customization is disabled, enable it.
3. In *c11n_Home*, verify that the following directories exist. If they do not exist, create them:

```
/c11n_Home/allDomain/themes/basic/images/
```

4. In *c11n_Home*, verify that **config.js** exists. If it does not exist, create it.
5. Modify **config.js** so that it enables theme customization (**themeEnabled: true**), i18n customization (**i18nEnabled: true**), and Javascript customization (**jsEnabled: true**) across all domains (module: "**allDomain**").

The following example shows a **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
    },
}
```

6. In *c11n_Home/allDomain/themes*, create or modify **customize.js** 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.

```
iwc.api.modifyThemeValues("theme_basic", "../c11n/allDomain/themes/basic/
theme.json");
```

7. In *c11n_Home/allDomain/themes/basic*, edit **theme.json** with the location of the logo, the logo's width, and height:

```
{
  logoBackground: 'transparent url("../c11n/allDomain/themes/basic/images/logo.png")
  no-repeat center center',
  logoWidth: "180px",
  mastheadHeight: "40px"
}
```

8. Replace the **logo.png** file in the **/themes/basic/images/** directory with your logo.
9. Restart the Oracle WebLogic Server and clear the browser cache to see the changes.

Adding a Logo to the Right Side of the Banner

This example explains how to add a logo to the right side of the banner for all domains in the deployment.

1. Verify that the `c11n_Home` directory exists. If `c11n_Home` does not exist, create it.
2. Verify that customization is enabled in Convergence. If customization is disabled, enable it.
3. In `c11n_Home`, verify that the following directories exist. If they do not exist, create them:

```
/c11n_Home/allDomain/js/widget/
```

4. In `c11n_Home`, verify that **config.js** exists. If it does not exist, create it.
5. Modify **config.js** so that it enables JavaScript customization (**jsEnabled: true**) across all domains (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

    }
}
```

6. In `c11n_Home/allDomain/js`, create or modify **customize.js** (the domain specific customization file) to include the following code:

```
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");
```

7. In `c11n_Home/allDomain/js/widget`, create **Banner.js** (the file that adds the logo to the right side of the banner). In **Banner.js**, do the following:

- The `float:right` parameter aligns the dom node to the right side of the banner.
- Change the style "height" from "40px" to the height of the logo space.
- change the style "width" from "138px" to the width of the logo space.
- change the style "background" URL from 'url("../c11n/allDomain/layout/images/logo.gif")', to the logo location of the logo you want to add to the banner. The URL should be relative to **main.html**.

```
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");
    }
    });

```

- Restart the Oracle WebLogic Server and clear the browser cache to see the change.

Figure 5-1 shows the logo on the right side of an example banner.

Figure 5-1 Customized Banner with Logo on Right Side



Making the Banner Logo a Clickable Link

This example assumes you have already added a logo to all themes. See ["Adding a Logo to All Themes"](#) for more information.

This example explains how to make a logo in the banner a clickable link in all domains in the deployment.

- Verify that the `c11n_Home` directory exists. If `c11n_Home` does not exist, create it.
- Verify that customization is enabled in Convergence. If customization is disabled, enable it.
- In `c11n_Home`, verify that the following directories exist. If they do not exist, create them:

```
/c11n_Home/allDomain/js/widget/
```

- In `c11n_Home`, verify that `config.js` exists. If it does not exist, create it.
- Modify `config.js` so that it enables JavaScript customization (`jsEnabled: true`) across all domains (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
    }
}

```

- In `c11n_Home/allDomain/js`, create or modify `customize.js` (the domain specific customization) to include the following code:

```
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. In `c11n_Home/allDomain/js/widget`, create the JavaScript file (**Banner.js**) to make the logo in a clickable link in the banner. 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 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 to link the logo to a URL. Comment out the first **logoOnClick** function.

```
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 the Oracle WebLogic Server and clear the browser cache to see the change.

Handling Large Logos in Gradient Themes

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 banner. However, themes with gradient images require adding a different image to fill the entire height of the banner when a larger logo is inserted. The theme needs to be customized to use the larger background image for the banner.

Figure 5-2 shows an 80px logo on top of a 40px gradient theme.

Figure 5-2 Large Logo on a Gradient Theme Not Displaying Properly

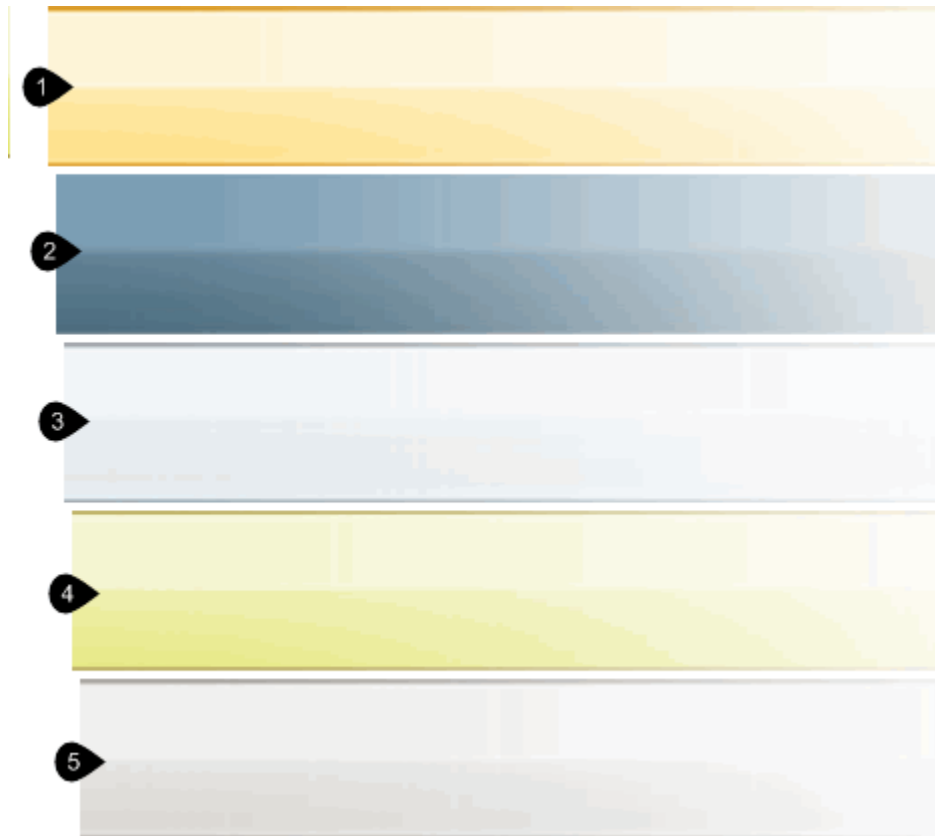


Re-Sized Gradient Banner Samples

Use the following banner samples when manipulating large logos (>40px) in gradient themes. The banner image files are available with the Convergence software on the Oracle software delivery site.

1. Yellow theme sample (yellow_masthead.png)
2. Dark blue theme sample (dark_blue_masthead.png)
3. Light blue theme sample (light_bue_masthead.png)
4. Green theme sample (green_masthead.png)
5. Grey them sample (grey_masthead.png)

Figure 5-3 Large Gradient Theme Samples



Customizing the Dark Blue Theme

To handle large logos in gradient themes, you must create a different 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 **masthead.png** into the *c11n_Home/allDomain/themes/dark_blue/images/* directory.
2. Follow the example in the section "[Modifying a Specific Theme](#)".
3. Set the **mastheadBackground** property to use the new image in the dark blue theme configuration (**theme.json**) file in the *c11n_Home/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 as to create the right look and feel.

Adding and Removing Fonts from the Editor Menu

To add and remove fonts from the editor menu:

1. Verify that the *c11n_Home* directory exists. If *c11n_Home* does not exist, create it.

2. Verify that customization is enabled in Convergence. If customization is disabled, enable it.
3. In *c11n_Home*, verify that the following directories exist. If they do not exist, create them:

```
/c11n_Home/allDomain/js/
```

4. In *c11n_Home*, verify that **config.js** exists. If it does not exist, create it.
5. Modify **config.js** so that it enables JavaScript customization (**jsEnabled: true**) across all domains (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
    }
}
```

6. In *c11n_Home/allDomain/js*, create or modify **customize.js** (the domain specific customization) to include the following code:

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

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

7. In *c11n_Home/allDomain/js/editor/plugins*, create the file (**FontChoice.js**) that adds the new font to the editor menu.

In the following example, the new font is a Japanese font named Ms_pgothic:

```
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: ""
    });
```

To translate the font name "ms gothic" in the editor font choice menu, see: Translating the Font Name "ms gothic" in the Editor Font Choice menu.

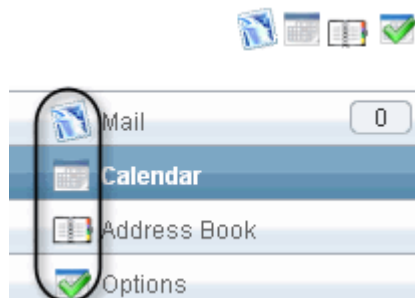
- Restart the Oracle WebLogic Server and clear the browser cache to see the change.

Changing an Icon in the Service Selector

This example describes how to change the Mail, Address Book, or Options icon in the service selector for all domains in your deployment.

Figure 5-4 shows the appearance of the default icon sprite above the default service selector icons.

Figure 5-4 Original Icon Sprite and Service Selector Icons



- Verify that the *c11n_Home* directory exists. If *c11n_Home* does not exist, create it.
- Verify that customization is enabled in Convergence. If customization is disabled, enable it.
- In *c11n_Home*, verify that the following directories exist. If they do not exist, create them:

```
/c11n_Home/allDomain/js/  
/c11n_Home/allDomain/layout/css
```

- In *c11n_Home*, verify that **config.js** exists. If it does not exist, create it.
- Modify **config.js** so that it enables JavaScript customization (**jsEnabled: true**) across all domains (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  
  
    }  
}
```

- In *c11n_Home/allDomain/js*, create or modify **customize.js** (the domain specific customization) to include the following code:

```

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");

```

7. In `c11n_Home/allDomain/layout/images`, add new icons or icon sprite (a single image that contains multiple icons).
8. In `c11n_Home/allDomain/layout/css`, create `c11n_icons.css` to point to the new images.

In the following example, the referenced image sprite is called **new_services_icons_sprite.png**:

```

/* 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;
}
```

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:

```
/* 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;
}
```

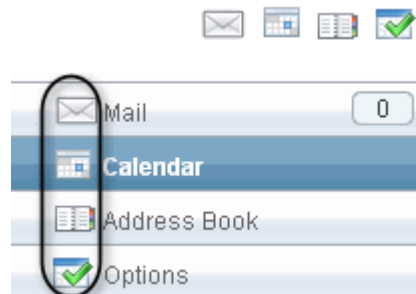
To use an individual image instead of an image sprite, point the background-image to the icon image of the individual service. For example, in the Mail service:

```
/* 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;
}
```

- Restart the Oracle WebLogic Server and clear the browser cache to see the change.

Figure 5-5 shows a new icon sprite and new service selector icons.

Figure 5-5 Example Updated Icon Sprite and Service Selector Icons



Displaying and Printing the Japanese Yen Symbol

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

To display the Yen symbol in an email subject, calendar title, and calendar description:

- Verify that the *c11n_Home* directory exists. If *c11n_Home* does not exist, create it.
- Verify that customization is enabled in Convergence. If customization is disabled, enable it.
- In *c11n_Home*, verify that the following directories exist. If they do not exist, create them:

```
/c11n_Home/allDomain/js/
/c11n_Home/allDomain/layout/css
```

4. In `c11n_Home/allDomain/js`, create or modify `customize.js` (the domain specific customization) to include the following code:

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

5. In `c11n_Home/allDomain/js`, modify `customize.js` and add or uncomment the following code:

```
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")
```

6. In `c11n_Home/allDomain/layout/css`, edit `c11n.css` and remove or comment the following code:

```
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;
}
```

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

To display the Yen symbol in the email preview pane:

1. Verify that the `c11n_Home` directory exists. If `c11n_Home` does not exist, create it.
2. Verify that customization is enabled in Convergence. If customization is disabled, enable it.
3. In `c11n_Home`, verify that the following directories exist. If they do not exist, create them.

```
c11n_Home/allDomain/js/editor
c11n_Home/allDomain/js/widget/mail
c11n_Home/allDomain/layout/css/editor
```

4. In `c11n_Home`, verify that `config.js` exists. If it does not exist, create it.
5. Modify `config.js` so that it enables JavaScript customization (`jsEnabled: true`) across all domains (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

}
}

```

6. In `c11n_Home/allDomain/js/widget/mail`, create `IframeMessagePane.js` and add the following code:

```

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";

```

7. In `c11n_Home/allDomain/layout/css`, create or modify `MessagePane.css` and add the following style:

```

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

```

8. In `c11n_Home/allDomain/js/editor`, create `Editor.js` and add the following code:

```

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";

```

9. In `c11n_Home/allDomain/layout/css/Editor`, create `Editor.css` and add the following style:

```

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

```

10. In `c11n_Home/allDomain/js`, create or modify `customize.js` (the domain specific customization) to include the following code:

```

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

```

11. Restart the Oracle WebLogic Server and clear the browser cache to see the change.

To add a new font to the editor font menu:

1. Verify that the `c11n_Home` directory exists. If `c11n_Home` does not exist, create it.
2. Verify that customization is enabled in Convergence. If customization is disabled, enable it.
3. In `c11n_Home`, verify that the following directories exist. If they do not exist, create them:


```

/c11n_Home/allDomain/js/editor/editor/plugins

```
4. In `c11n_Home`, verify that `config.js` exists. If it does not exist, create it.
5. Modify `config.js` so that it enables JavaScript customization (`jsEnabled: true`) across all domains (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
    }
}

```

6. In *c11n_Home/allDomain/js*, create or modify **customize.js** (the domain specific customization) to include the following code:

```

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

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

```

7. In *c11n_Home/allDomain/js/editor/plugins*, create or modify **FontChoice.js** with the following sample:

```

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
        }
    });

```

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

To translate the ms pgothic font name in the editor font menu:

1. Verify that the *c11n_Home* directory exists. If *c11n_Home* does not exist, create it.
2. Verify that customization is enabled in Convergence. If customization is disabled, enable it.
3. In *c11n_Home*, verify that the following directories exist. If they do not exist, create them:

```

/c11n_Home/allDomain/nls/

```

4. In `c11n_Home`, verify that **config.js** exists. If it does not exist, create it.
5. Modify **config.js** so that it enables i18n customization (**i18nEnabled: true**) across all domains (module: **"allDomain"**).

```
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
    },
}
```

6. In `c11n_Home/allDomain/nls`, edit **resources.js** and add the following code:

```
{
    "ms pgothic": "localized_font_name",
    last: ""
}
```

Where `localized_font_name` is the name you give the font.

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

To print the Yen Symbol:

1. Perform the following:

- For Oracle WebLogic Server, in `data_directory/web-src/client/iwc_static/layout/css`, edit **Default.css** by adding the following lines at the end of the file:

```
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;
}
```

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.

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.

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), see ["Working with the Convergence UI"](#).

Figure 5-6 shows the document title and the Convergence text in the banner before the customization is applied.

Figure 5-6 Document Title and Convergence Text Before Customization



1. Verify that the *c11n_Home* directory exists. If *c11n_Home* does not exist, create it.
2. Verify that customization is enabled in Convergence. If customization is disabled, enable it.
3. In *c11n_Home*, verify that the following directories exist. If they do not exist, create them:

```
/c11n_Home/mydomain_org/js/widget/  
/c11n_Home/mydomain_org/nls/
```

In this example, *mydomain_org* is the single domain that's being modified.

4. In *c11n_Home*, verify that **config.js** exists. If it does not exist, create it.
5. Modify **config.js** so that it enables i18n customization (**i18nEnabled: true**) for the *mydomain_org* domain (module: "**mydomain_org**").

```
dojo.provide("c11n.config");  
c11n.config = {  
  
    // allDomain configuration  
    allDomain: {  
        ...  
    },  
  
    // 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  
  
    }  
}
```

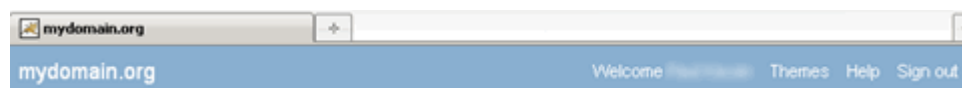
6. In *c11n_Home/mydomain_org/nls*, create **resources.js** that specifies the document title text:

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

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

Figure 5-7 shows the document title and the Convergence text in the banner after the customization is applied.

Figure 5-7 Document Title and Convergence Text After Customization



Changing Names and Labels in the Convergence UI

You can change the labels for icons, tabs, menus, and other elements in the Convergence 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.

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

1. Verify that the *c11n_Home* directory exists. If *c11n_Home* does not exist, create it.
2. Verify that customization is enabled in Convergence. If customization is disabled, enable it.
3. In *c11n_Home*, verify that the following directories exist. If they do not exist, create them:

```
/c11n_Home/allDomain/nls/
```

4. In *c11n_Home*, verify that **config.js** exists. If it does not exist, create it.
5. Modify **config.js** so that it enables i18n customization (**i18nEnabled: true**) across all domains (module: "**allDomain**").

```
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
    }
}
```

6. In *c11n_Home/allDomain/nls*, create a default resource file named **resources.js**.

This customized resources file extends the default label values with the new, custom values.

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

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

In this example, the text New Mail Message replaces the default text [No Subject] when the user composes a new email.

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

At run time, the default **resources.js** file in the default directory is loaded first, followed by the customized **resources.js** file. 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.

- 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:

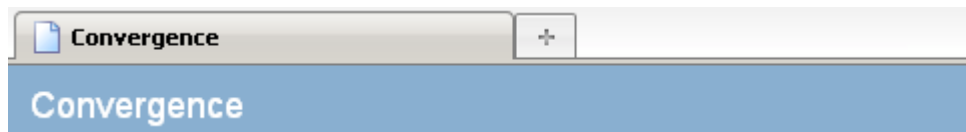
```
c11n_Home/Domain/nls/fr/resources.js
```

Removing or Changing the Product Name on the Mail HTML Page

This example outlines how to remove or change the product name from Convergence in the Banner. 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.

Figure 5-8 shows the default product name in the banner.

Figure 5-8 Default Product Name in Banner



- Verify that the `c11n_Home` directory exists. If `c11n_Home` does not exist, create it.
- Verify that customization is enabled in Convergence. If customization is disabled, enable it.
- In `c11n_Home`, verify that the following directories exist. If they do not exist, create them:

```
/c11n_Home/allDomain/nls/
```

- In `c11n_Home`, verify that **config.js** exists. If it does not exist, create it.
- Modify **config.js** so that it enables i18n customization (**i18nEnabled: true**) across all domains (module: "**allDomain**").

```
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
    }
}
```

- In `c11n_Home/allDomain/nls`, create or modify **resources.js** and add the following code:

```
{
  ...
  login_product_name: "Customized Convergence",
}
```

```
last: ""
}
```

Figure 5-9 shows the updated product name.

Figure 5-9 Customized Product Name in Banner and Web Browser



You can remove the product name from the banner completely if you leave an empty string for the `login_product_name` attribute in `resources.js`.

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

However, without a product name, your web browser may display the host name or IP address in the address bar, as shown in Figure 5-10.

Figure 5-10 Product Name Removed from Banner



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

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).

To display your password policy in Convergence:

1. Verify that the `c11n_Home` directory exists. If `c11n_Home` does not exist, create it.
2. Verify that customization is enabled in Convergence. If customization is disabled, enable it.
3. In `c11n_Home`, verify that the following directories exist. If they do not exist, create them:

```
/c11n_Home/allDomain/js/option
```

4. In `c11n_Home`, verify that `config.js` exists. If it does not exist, create it.
5. Modify `config.js` so that it enables JavaScript customization (**jsEnabled: true**) across all domains (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

    }
}

```

6. In `c11n_Home/allDomain/js`, create or modify **customize.js** (the domain specific customization) to include the following code:

```

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

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

```

7. In `c11n_Home/allDomain/js/option`, create or modify **Password.js** in which you add the password policy information on the Change Password page:

```

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: ""
    }
);

```

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

Hiding the Quick Actions Menu

To hide the Quick Action menu in the Convergence UI from all domains:

1. Verify that the `c11n_Home` directory exists. If `c11n_Home` does not exist, create it.
2. Verify that customization is enabled in Convergence. If customization is disabled, enable it.
3. In `c11n_Home`, verify that the following directories exist. If they do not exist, create them:

```

/c11n_Home/allDomain/js/widget
/c11n_Home/allDomain/layout/css

```

4. In `c11n_Home`, verify that **config.js** exists. If it does not exist, create it.
5. Modify **config.js** so that it enables JavaScript customization (**jsEnabled: true**) across all domains (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
}
}
```

6. In `c11n_Home/allDomain/js`, create or modify **customize.js** and uncomment the following code:

```
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")
```

7. In `c11n_Home/allDomain/layout/css`, create or modify **c11n.css** to include the following code:

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

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

6

Convergence Messaging Customization Examples

This chapter provides several examples for customizing the messaging service in Oracle Communications Convergence.

Customization Requirements

To perform any Convergence customization, you should have expert knowledge in dojo and knowledge of UI customization. For more information on general Convergence customization, see "[Technical Overview](#)".

Nearly all customization examples require the same setup and preparation.

- You start by verifying that the `c11n_Home` directory exists. If it does not exist, create it. See "[Customization Directory Structure](#)" for more information.
- Next, you verify that customization is enabled in Convergence. If customization is disabled, enable it. See "[Enabling Customization in the Convergence Server](#)" for more information.
- When you have completed your customization, you must restart the Oracle WebLogic Server for Convergence. See *Convergence System Administrator's Guide* for more information.

Changing the Mail Forward Default from As Attachment to Inline

When you click the Forward button to forward an email message, 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. Verify that the `c11n_Home` directory exists. If `c11n_Home` does not exist, create it.
2. Verify that customization is enabled in Convergence. If customization is disabled, enable it.
3. In `c11n_Home`, verify that the following directories exist. If they do not exist, create them:
 - `/c11n_Home/allDomain/js/widget/mail`
 - `/c11n_Home/allDomain/nls`
4. In `c11n_Home`, verify that **config.js** exists. If it does not exist, create it.
5. Modify **config.js** so that it enables JavaScript customization and i18n customization across all domains (module: "**allDomain**").

```
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

    }
}

```

6. In `c11n_Home/allDomain/js`, create or modify **customize.js** (the domain specific customization) to include the following code:

```

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

dojo.require("c11n.allDomain.js.widget.mail.OpenFolder");
dojo.require("c11n.allDomain.js.widget.mail.OpenMessage");

```

7. In `c11n_Home/allDomain/js/widget/mail`, create a JavaScript file (**OpenFolder.js**) that changes the mail forward default from As Attachment to Inline in the mail.OpenFolder widget.

```

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: ""

});

```

8. In `c11n_Home/allDomain/js/widget/mail`, create a JavaScript file (**OpenMessage.js**) that changes the mail forward default from As Attachment to Inline in the mail.OpenMessage widget.

```

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: ""

});

```

9. In *c11n_Home/allDomain/nls*, create a default resource file named **resources.js** to contain the new labels in the UI.

10. Modify **resources.js** to include the following code:

```

{
    forward_attach_item: "Inline",
    forward_inline_item: "As Attachment",
}

```

11. Restart the Oracle WebLogic Server and clear the browser cache to see the change.

Changing Default Folder Mappings for Sent and Deleted Messages

By default, copies of sent messages are mapped to the **Sent** folder and deleted messages are mapped to the **Trash** folder. You can change the default folder names for the **Sent** folder and **Trash** folder. Users can then map sent and deleted messages to the folder names in the Mail Options **General** tab.

1. Verify that the *c11n_Home* directory exists. If *c11n_Home* does not exist, create it.
2. Verify that customization is enabled in Convergence. If customization is disabled, enable it.
3. In *c11n_Home*, verify that the following directories exist. If they do not exist, create them:

```
/c11n_Home/allDomain/js/widget/
```

4. In *c11n_Home*, verify that **config.js** exists. If it does not exist, create it.
5. Modify **config.js** so that it enables i18n customization (**i18nEnabled: true**) across all domains (module: "**allDomain**").

```

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
    }
}

```

6. In *c11n_Home/allDomain/js*, create or modify **customize.js** (the domain specific customization) to include the following code:

```
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";
```

7. In *c11n_Home/allDomain/nls*, edit **resources.js** by deleting the sample content and adding the following code:

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

    last: ""
}
```

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

In the Messaging section of the UI, the **Sent Messages** and **Deleted Messages** folder appear below the **Inbox** folder.

On the Mail Options **General** tab, the user can select to copy sent messages to the **Sent Messages** folder, and can select to move deleted messages to the **Deleted Messages** folder.

Changing From: Address to Only Include Email Address

This customization example describes how to change the From: address to only include the email address (not cn):

1. Verify that the *c11n_Home* directory exists. If *c11n_Home* does not exist, create it.
2. Verify that customization is enabled in Convergence. If customization is disabled, enable it.
3. In *c11n_Home*, verify that the following directories exist. If they do not exist, create them:

```
/c11n_Home/allDomain/js/mail/
```

4. In *c11n_Home*, verify that **config.js** exists. If it does not exist, create it.
5. Modify **config.js** so that it enables JavaScript customization (**jsEnabled: true**) across all domains (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
    }
}
```

6. In *c11n_Home/allDomain/js*, create or modify **customize.js** (the domain specific customization) to include the following code:

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

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

7. In `c11n_Home/allDomain/js/widget/mail`, create a JavaScript file (**CreateMessage.js**) that changes the From: address to only include the email address.

```
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: ""

});
```

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

Changing or Removing the Signature Separator

When a user composes an email using Convergence, Convergence auto-inserts a signature separator "-- " on a separate line before the user's email signature (if the user has configured an email signature).

You can change or remove the signature separator.

1. Verify that the `c11n_Home` directory exists. If `c11n_Home` does not exist, create it.
2. Verify that customization is enabled in Convergence. If customization is disabled, enable it.
3. In `c11n_Home`, verify that the following directories exist. If they do not exist, create them:

```
/c11n_Home/allDomain/js/widget/mail
```

4. In `c11n_Home`, verify that **config.js** exists. If it does not exist, create it.
5. Modify **config.js** so that it enables JavaScript customization (**jsEnabled: true**) across all domains (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
    }
}
```

6. In `c11n_Home/allDomain/js`, create or modify **customize.js** (the domain specific customization) to include the following code:

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

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

7. In `c11n_Home/allDomain/js/widget/mail`, create `CreateMessage.js` to specify the desired "signatureSeparator".

The default signature separator is "-- ". To remove the signature separator, set it to "".

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

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

dojo.require("iwc.api");

dojo.declare("iwc.widget.mail.CreateMessage", iwc.widget.mail.CreateMessage, {
  signatureSeparator: "",
  last: ""

});
```

8. Restart the Oracle WebLogic Server and clear the browser cache to see the change.
9. Log into Convergence, create an email signature, and create a new email message. The signature separator is changed.

Modifying Mail Folder Icons in the Service Navigator

This example describes how to customize the mail folder icons in the service navigator.

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. Verify that the `c11n_Home` directory exists. If `c11n_Home` does not exist, create it.
2. Verify that customization is enabled in Convergence. If customization is disabled, enable it.
3. In `c11n_Home`, verify that the following directories exist. If they do not exist, create them:

```
/c11n_Home/allDomain/js/
/c11n_Home/allDomain/layout/css/
```

4. In `c11n_Home`, verify that `config.js` exists. If it does not exist, create it.
5. Modify `config.js` so that it enables JavaScript customization (`jsEnabled: true`) across all domains (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

  }
}
```

6. In `c11n_Home/allDomain/js`, create or modify `customize.js` (the domain specific customization) to include the following code:

```
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");
    
```

7. Add new icons or an icon sprite (a single image that contains multiple icons) to the *c11n_Home/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:

Table 6-1 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

Table 6-1 (Cont.) Default and New Icons in Service Navigator Example

Icon	Default Image	New Image	New Image File Name
Drafts Folder			treeMailDrafts_new_red.png

8. In `c11n_Home/allDomain/layout/css`, create `c11n_icons.css` to point to the new image icons or sprite. In this example, the **background-image** points to the new images in `c11n_Home/allDomain/layout/images/`:

```

/* Personal Mail Icon */
.FolderIcons {
    width: 16px;
    height: 16px;
    background-repeat:no-repeat;
    background-position:top left;
    background-image:url("../images/treeMailFolderPersonal_new_red.png");
}

/* Root Folder Icon */
.FolderIcons_Root {
    background-image: url("../images/optionsIconMail_new_red.png");
    background-repeat: no-repeat;
    background-position: center center;
    background-color: transparent;
}

/* Inbox Folder Icon */
.FolderIcons_Inbox {
    background-image: url("../images/treeMailInbox_new_red.png");
    background-repeat: no-repeat;
    background-position: center center;
    background-color: transparent;
}

/* Trash Folder Icon */
.FolderIcons_Shared {
    background-image: url("../images/MailFolders_new_red.png");
    background-repeat: no-repeat;
    background-position: 0 0px;
    background-color: transparent;
}

/* Trash Folder Icon */
.FolderIcons_Trash {
    background-image: url("../images/MailFolders_new_red.png");
    background-repeat: no-repeat;
    background-position: 0 -102px;
    background-color: transparent;
}

/* Sent Folder Icon */
.FolderIcons_Sent {
    background-image: url("../images/treeMailSent_new_red.png");
    background-repeat: no-repeat;
    background-position: center center;
    background-color: transparent;
}
    
```



```

/* Drafts Folder Icon */
.FolderIcons_Drafts {
    background-image: url("../images/treeMailDrafts_new_red.png");
    background-repeat: no-repeat;
    background-position: center center;
    background-color: transparent;
}

/* 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;
}

```

- Restart the Oracle WebLogic Server to clear the browser cache and view the change.

Removing Folder Sharing and Subscribing Menu Options

While you can disable folder sharing and folder subscriptions with the following commands:

```
iwcadm -o mail.restrictanyone -v true
```

And

```
configutil -o store.privatesharedfolders.restrictanyone -v 1
```

The **Share Folder** and **Subscribe To Folder** menu options still appear in the UI.

To remove these menu options from the UI:

- Verify that the *c11n_Home* directory exists. If *c11n_Home* does not exist, create it.
- Verify that customization is enabled in Convergence. If customization is disabled, enable it.
- In *c11n_Home*, verify that the following directories exist. If they do not exist, create them:

```
/c11n_Home/allDomain/js/widget/
```

- In *c11n_Home*, verify that **config.js** exists. If it does not exist, create it.
- Modify **config.js** so that it enables JavaScript customization (**jsEnabled: true**) across all domains (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
    }
}

```

- In *c11n_Home/allDomain/js*, create or modify **customize.js** (the domain specific customization) to include the following code:

```
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");
```

7. In `c11n_Home/allDomain/js/widget/mail`, create the JavaScript customization file (**Navigator.js**) to remove folder sharing and subscribing menu options from the drop-down menus.

```
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: ""
});
```

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

Removing the Local Account Mail Forwarding Option

By default, users can set up their local account to automatically forward all incoming email messages.

You can configure Convergence to remove the Mail Local Account Forwarding option.

1. Verify that the *c11n_Home* directory exists. If *c11n_Home* does not exist, create it.
2. Verify that customization is enabled in Convergence. If customization is disabled, enable it.
3. In *c11n_Home*, verify that the following directories exist. If they do not exist, create them:

```
/c11n_Home/allDomain/js/widget/option/
```

4. In *c11n_Home*, verify that **config.js** exists. If it does not exist, create it.
5. Modify **config.js** so that it enables JavaScript customization (**jsEnabled: true**) across all domains (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
    }
}
```

6. In *c11n_Home/allDomain/js*, create or modify **customize.js** (the domain specific customization) to include the following code:

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

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

7. In *c11n_Home/allDomain/js/widget/option*, create **Navigator.js** to include the following code:

```
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 Vacation Message option
        //iwc.api.removeOption("/Mail/Local Account/Vacation Message");

        //remove the Mail Forwarding option
        iwc.api.removeOption("/Mail/Local Account/Forward");
    }
}
```

```

        //remove Mail Filters option
        //iwc.api.removeOption("/Mail/Local Account/Mail Filters");

    },

    last: ""
}
);

```

 **Note:**

The code to remove "Forwarding" is `iwc.api.removeOption("/Mail/Local Accounts/Forward");` not `Forwarding`.

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

Removing the Move Button in the Mail Open Folder

The following example deletes the **Move** button in Account Setting in the Mail Open Folder for users in all domains.

1. Verify that the `c11n_Home` directory exists. If `c11n_Home` does not exist, create it.
2. Verify that customization is enabled in Convergence. If customization is disabled, enable it.
3. In `c11n_Home`, verify that the following directories exist. If they do not exist, create them:

```
/c11n_Home/allDomain/js/widget/
```

4. In `c11n_Home`, verify that **config.js** exists. If it does not exist, create it.
5. Modify **config.js** so that it enables JavaScript customization (**jsEnabled: true**) across all domains (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
    }
}

```

6. In `c11n_Home/allDomain/js`, create or modify **customize.js** (the domain specific customization) to include the following code:

```

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

// Delete Move Button in Mail Option Folder
dojo.require("c11n.allDomain.js.widget.mail.OpenFolder");

```

7. In `c11n_Home/allDomain/js/widget/mail`, create the JavaScript file (`OpenFolder.js`) which deletes the **Move** button.

```

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: ""

});

```

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

Removing the Reply-To Address Option

The following example hides the **Reply-To: Account Setting** in the Mail option for users in all domains.

1. Verify that the *c11n_Home* directory exists. If *c11n_Home* does not exist, create it.
2. Verify that customization is enabled in Convergence. If customization is disabled, enable it.
3. In *c11n_Home*, verify that the following directories exist. If they do not exist, create them:
/c11n_Home/allDomain/js/widget/mail/option/
4. In *c11n_Home*, verify that **config.js** exists. If it does not exist, create it.
5. Modify **config.js** so that it enables JavaScript customization (**jsEnabled: true**) across all domains (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

    }
}

```

6. In *c11n_Home/allDomain/js*, create or modify **customize.js** (the domain specific customization) to include the following code:

```

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

// Hide replyTo Option from the Mail Account Settings
dojo.require("c11n.allDomain.js.widget.mail.option.Identity");

```

7. In `c11n_Home/allDomain/js/widget/mail/option`, create the JavaScript file (**Identity.js**) which hides the **Reply-To:** option. This example uses the `dojoAttachPoint="replyTo"` value in the `Convergence_Domain/docroot/iwc_static/js/debug/iwc/widget/templates/mail/option/Identity.html` file to determine the appropriate location in the DOM-tree to make the modification.

```

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: ""

});

```

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

Restricting Outgoing Mail with Local Account Identity Parameters

If you do not want to disable external POP account access, but you want to control what appears in the **From** menu for addressing, the following customization example only allows outgoing mail to use the parameters in the local account identity:

To fully disable POP account functionality and to hide External Accounts in the Options panel in Mail, see ["Disabling External POP Account Access"](#) for more information.

1. Verify that the `c11n_Home` directory exists. If `c11n_Home` does not exist, create it.
2. Verify that customization is enabled in Convergence. If customization is disabled, enable it.
3. In `c11n_Home`, verify that the following directories exist. If they do not exist, create them:

```
/c11n_Home/allDomain/js/widget/option/
```

4. In `c11n_Home`, verify that **config.js** exists. If it does not exist, create it.
5. Modify **config.js** so that it enables JavaScript customization (**jsEnabled: true**) across all domains (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

```

- ```

 }
}

```
- In *c11n\_Home/allDomain/js*, create or modify **customize.js** (the domain specific customization) to include the following code:

```

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

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

```
  - In *c11n\_Home/allDomain/js/widget/mail*, create or modify the Javascript file (**CreateMessage.js**) to only allow outgoing mail to use the parameters in the local account identity:

```

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");
 }
});

```
  - Restart the Oracle WebLogic Server and clear the browser cache to see the change.

## Hiding User-Created Folder Names

By default, sent messages are mapped to the **Sent** folder, deleted messages are mapped to the **Trash** folder, spam messages are mapped to the **Spam** folder, and drafts are mapped to the **Drafts** folder. Users can map their sent messages, drafts, deleted messages, and spam messages to custom folders.

You can customize Convergence such that the custom folder name is hidden and the default name is used.

- Verify that the *c11n\_Home* directory exists. If *c11n\_Home* does not exist, create it.
- Verify that customization is enabled in Convergence. If customization is disabled, enable it.
- In *c11n\_Home*, verify that the following directories exist. If they do not exist, create them:

```

/c11n_Home/allDomain/nls/

```
- In *c11n\_Home*, verify that **config.js** exists. If it does not exist, create it.
- Modify **config.js** so that it enables i18n customization (**i18nEnabled: true**) across all domains (module: "**allDomain**").

```

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
 }
};

```

```
 }
}
```

6. In the `/nls` directory, create the customization file (**resources.js**).
7. In **resources.js**, add the following text strings:

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

This customized resources file inherits the default values in the standard resources file and extends them with the new, custom values. At run time, 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.

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, create **resources.js** in `c11n_Home/allDomain/nls/fr`. In **resources.js**, add the following code:

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

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

## Adding Additional Spell Checker Dictionaries

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://www.lasr.cs.ucla.edu/geoff/ispell.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:

<https://www.cs.hmc.edu/~geoff/ispell-dictionaries.html#Italian-dicts>

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
```

For example:

```
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 do not 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
5. Verify that customization is enabled in Convergence. If customization is disabled, enable it.
6. Verify that the **c11n\_Home** directory exists. If **c11n\_Home** does not exist, create it.
7. In **c11n\_Home**, verify that the following directories exist. If they do not exist, create them:

```
/c11n_Home/allDomain/js/widget/
/c11n_Home/allDomain/nls/
```

8. In **c11n\_Home**, verify that **config.js** exists. If it does not exist, create it.
9. Modify **config.js** so that it enables JavaScript customization (**jsEnabled: true**) and i18n customization (**i18nEnabled: true**) across all domains (module: **"allDomain"**).

```
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
 }
}

```

10. In `c11n_Home/allDomain/nls`, create the Javascript file (**resources.js**) to define the localized label for the "Italian" dictionary.

```

{
 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: ""
}

```

11. (Optional) To make the label different for each locale, do the following:

- a. Add a localization directory for the locale. For example:

```
c11n_Home/allDomain/nls/it
```

- b. In `c11n_Home/allDomain/nls/it/`, create a Javascript file (**resources.js**) in the specified locale to display the label. For example:

```

{
 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: ""
}

```

12. Restart the Oracle WebLogic Server and clear the browser cache to see the change.

## Customizing the Attachment Blacklist and Whitelist

You can customize the Oracle Outside In Transformation Server blacklist to prevent certain types of attachments from being sent to the transformation server, such as ZIP files or EXE files.

Also, you can customize the Oracle Outside In Transformation Server whitelist to specify types of attachments that are sent to the transformation server, such as DOC files or XLS files.

Each time an attachment preview is requested by a user, the Outside In Proxy consults both the blacklist and the whitelist to determine whether to send the attachment to the transformation server.

See *Convergence System Administrator's Guide* for more information about Outside In Transformation Server.

To customize the transformation server blacklist or whitelist:

1. Verify that the `c11n_Home` directory exists. If `c11n_Home` does not exist, create it.
2. Verify that customization is enabled in Convergence. If customization is disabled, enable it.
3. In `c11n_Home`, verify that the following directories exist. If they do not exist, create them:

```
/c11n_Home/allDomain/js/
```

4. In `c11n_Home`, verify that **config.js** exists. If it does not exist, create it.

5. Modify **config.js** so that it enables JavaScript customization (**jsEnabled: true**) across all domains (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
 }
}
```

6. In `c11n_Home/allDomain/js`, create or modify **customize.js** (the domain specific customization) to include the following code:

```
iwc.config.oin
mimeTypes: {
 whiteList: [
 ["*", "*"]
],
 blacklist: [// ignore exe and zip files
 ["application", "octet-stream"],
 ["application", "x-msdownload"],
 ["application", "x-compressed"],
 ["application", "x-zip-compressed"],
 ["application", "zip"],
 ["application", "x-zip"]
]
}
```

7. To customize the whitelist:

In **customize.js**, add or remove file types after the line **whiteList: [**.

By default, the whitelist consists of the code `["*", "*"]`, which means that all attachment types (excluding those in the blacklist) are sent to the transformation server.

You can update the whitelist to enumerate a limited and specific list of attachment types. For example, to allow only DOC, XLS, and PPT attachment types to be sent to the transformation server:

```
whiteList: [
 ["application", "doc"]
 ["application", "xls"]
 ["application", "ppt"]
],
```

8. To customize the blacklist:

In **customize.js**, add or remove file types after the line **blackList: [**. For example, to prevent PDF files from being sent to the transformation server:

```
blackList: [// ignore exe and zip files
 ["application", "pdf"],
 ["application", "octet-stream"],
 ["application", "x-msdownload"],
 ["application", "x-compressed"],
 ["application", "x-zip-compressed"],
 ["application", "zip"],
```

```
 ["application", "x-zip"]
]
}
```

- Restart the Oracle WebLogic Server.

## Increasing Corporate Address Book Entries in Email Autocomplete

When composing an email and typing popular names in the 'To:' field, the Corporate Directory users are shown in the autocomplete results. The following example shows how to increase the number of entries listed in autocomplete.

- Verify that the `c11n_Home` directory exists. If `c11n_Home` does not exist, create it.
- Verify that customization is enabled in Convergence. If customization is disabled, enable it.
- In `c11n_Home`, verify that the following directories exist. If they do not exist, create them: `/c11n_Home/allDomain/js/widget/addressBook`
- In `c11n_Home`, verify that `config.js` exists. If it does not exist, create it.
- Modify `config.js` so that it enables JavaScript customization (`jsEnabled: true`) across all domains (`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 js is customized
 // the last entry must not end with comma
 }
}
```

- In `c11n_Home/allDomain/js`, create or modify `customize.js` (the domain specific customization) to include the following code:

```
dojo.provide("c11n.allDomain.js.customize");
dojo.require("c11n.allDomain.js.widget.addressBook.EmailComboTextarea");
```

- In `c11n_Home/allDomain/js/widget/addressBook`, create or modify `EmailComboTextarea.js` to include the following code:

```
dojo.provide("c11n.allDomain.js.widget.addressBook.EmailComboTextarea");
dojo["require"]("iwc.widget.addressBook.EmailComboTextarea");
dojo.declare("iwc.widget.addressBook.EmailComboTextarea",
iwc.widget.addressBook.EmailComboTextarea, {
 entryCount:20,
 additionalSearch Count:20,

 last: ''

});
```

 **Note:**

Replace 20 with whatever number you want to use. By default the number of search entries per page is 10. When entryCount is increased additionalSearchCount also needs to be increased.

8. Restart the application server and clear the browser cache to see the change.

# 7

## Convergence Calendar Customization Examples

This chapter provides several examples for customizing the calendar service in Oracle Communications Convergence.

### Customization Requirements

To perform any Convergence customization, you should have expert knowledge in dojo and knowledge of UI customization. For more information on general Convergence customization, see "[Technical Overview](#)".

Nearly all customization examples require the same setup and preparation.

- You start by verifying that the `c11n_Home` directory exists. If it does not exist, create it. See "[Customization Directory Structure](#)" for more information.
- Next, you verify that customization is enabled in Convergence. If customization is disabled, enable it. See "[Enabling Customization in the Convergence Server](#)" for more information.
- When you have completed your customization, you must restart the Oracle WebLogic Server for Convergence. See *Convergence System Administrator's Guide* for more information.

### Displaying a Complete Title in Calendar List Views

Currently, a long title on an event or task is truncated and appended with ellipses. This Convergence customization example describes how to display the complete title in the calendar list views (agenda/invitations/tasks) for events and tasks with long titles.

1. Verify that the `c11n_Home` directory exists. If `c11n_Home` does not exist, create it.
2. Verify that customization is enabled in Convergence. If customization is disabled, enable it.
3. In `c11n_Home`, verify that the following directories exist. If they do not exist, create them:  
`/c11n_Home/allDomain/layout/css/calendar/`
4. In `c11n_Home`, verify that `config.js` exists. If it does not exist, create it.
5. Modify `config.js` so that it enables JavaScript customization (`jsEnabled: true`) across all domains (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
 }
}

```

6. In the `c11n_Home/allDomain/layout/css/calendar`, create or modify `ListView.css` with the following code:

```

.CalendarListItemTask-Title h3 {
white-space: normal;
word-break: break-all;
height: auto;
}

.CalendarListItemEvent h3 {
height: auto;
}

.CalendarListItemEvent-Title h3 {
white-space: normal;
word-break: break-all;
}

```

7. In `c11n_Home/allDomain/js`, create or modify `customize.js` (the domain specific customization) to include the following code:

```

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);
}
loadCustomizedCssFile("../c11n/allDomain/layout/css/calendar/ListView.css")

```

8. Restart the Oracle WebLogic Server, clear the browser cache and reload the page to see the change.

## Adding or Modifying Calendar Time Zones

Convergence dynamically generates time-zone options based on time-zone data downloaded through a WCAP call to the Oracle Communications Calendar Server defined in the `cal.host` Convergence configuration option. In this way the time-zone settings are kept in-sync between Convergence and Calendar Server.

Calendar Server time zone data is stored in the Calendar Server `timezones.ics` configuration file. See your Calendar Server documentation for more information.

The following example demonstrates how to modify calendar time zones when Convergence is integrated with Calendar Server 7:

- Adding a new calendar time zone for Saskatchewan in Canada
  - Modifying the existing "America Los Angeles" time zone to display as the more generic "America Pacific Time".
1. Verify that the *c11n\_Home* directory exists. If *c11n\_Home* does not exist, create it.
  2. Verify that customization is enabled in Convergence. If customization is disabled, enable it.
  3. In *c11n\_Home*, verify that the following directories exist. If they do not exist, create them:

```
/c11n_Home/allDomain/nls/
```

4. In *c11n\_Home*, verify that **config.js** exists. If it does not exist, create it.
5. Modify **config.js** so that it enables i18n customization (**i18nEnabled: true**) across all domains (module: "**allDomain**"). This allows for Calendar Server TZID string to City Name translations.

```
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
 }
}
```

6. For instructions on adding and modifying new WCAP time zones in Calendar Server 7, see the Calendar Server documentation.
7. In *c11n\_Home/allDomain/nls*, create the Javascript file (**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**:

```
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, to change the name of a TZID (for example Los Angeles to Pacific Time), and you do not specify the change in **resources.js**, the original name (in this case, Los Angeles) displays in the UI.

8. Restart the Oracle WebLogic Server, clear the browser cache and reload the page to see the change.



## Categorizing Calendar Events with Text or Background Colors

This 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. Verify that the `c11n_Home` directory exists. If `c11n_Home` does not exist, create it.
2. Verify that customization is enabled in Convergence. If customization is disabled, enable it.
3. In `c11n_Home`, verify that the following directories exist. If they do not exist, create them:

```
./c11n_Home/allDomain/js/widget/calendar/
```

4. In `c11n_Home`, verify that `config.js` exists. If it does not exist, create it.
5. Modify `config.js` so that it enables JavaScript customization (`jsEnabled: true`) across all domains (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
 }
}
```

6. In `c11n_Home/allDomain/js`, create or modify `customize.js` (the domain specific customization) to include the following code:

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

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

7. Modify `customize.js` and uncomment the following:

```
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")
```

8. In `c11n_Home/allDomain/js/widget/calendar`, create the Javascript file (**Event.js**) that specifies the color values for types of events. 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:

```

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: ""
});

```

9. In `c11n_Home/allDomain/layout/css`, modify **c11n.css** by adding the following code to define each category with text color.

```

.redCategory {
 color: red;
}

```

The given example illustrates the changes for red category, similar changes can be done for blue category and purple category as well.

10. Modify **c11n.css** by adding the following code to define each category with background color.

```

.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");
}

```

A list of Convergence background colors exists in *Convergence\_DomainIdocroot/iwc\_static/layout/images/calendar*. The given example illustrates the changes for red category, similar changes can be done for blue category and purple category as well.

11. Restart the Oracle WebLogic Server, clear the browser cache and reload the page to see the change.

## Disabling Event Balloon User Input Saving as 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. Verify that the *c11n\_Home* directory exists. If *c11n\_Home* does not exist, create it.
2. Verify that customization is enabled in Convergence. If customization is disabled, enable it.
3. In *c11n\_Home*, verify that the following directories exist. If they do not exist, create them:  
*/c11n\_Home/allDomain/js/widget/calendar/*
4. In *c11n\_Home*, verify that **config.js** exists. If it does not exist, create it.
5. Modify **config.js** so that it enables JavaScript customization (**jsEnabled: true**) across all domains (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
 }
}

```

- In `c11n_Home/allDomain/js`, create or modify `customize.js` (the domain specific customization) to include the following code:

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

// Disable event balloon user input from being saved as event description
dojo.require("c11n.allDomain.js.widget.calendar.QuickEventBalloon");
```

- In `c11n_Home/allDomain/js/widget/calendar`, create `QuickEventBalloon.js`, which disables event balloon user input from being saved as event description:

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

dojo.require("iwc.widget.calendar.QuickEventBalloon");

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: ""
});
```

- Restart the Oracle WebLogic Server, clear the browser cache and reload the page to see the change.

## Disabling Quick Parsing Calendar Capabilities

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 quick parsing.

- Verify that the `c11n_Home` directory exists. If `c11n_Home` does not exist, create it.
- Verify that customization is enabled in Convergence. If customization is disabled, enable it.
- In `c11n_Home`, verify that the following directories exist. If they do not exist, create them:

```
/c11n_Home/allDomain/js/widget/calendar/
```

- In `c11n_Home`, verify that `config.js` exists. If it does not exist, create it.
- Modify `config.js` so that it enables JavaScript customization (`jsEnabled: true`) across all domains (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
 }
}

```

6. In `c11n_Home/allDomain/js`, create or modify `customize.js` (the domain specific customization) to include the following code:

```

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

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

```

7. In `c11n_Home/allDomain/js/widget/calendar`, create the JavaScript file (`QuickEventBalloon.js`) that disables quick parsing feature in calendar:

```

dojo.provide("c11n.allDomain.js.calendar.QuickEventBalloon");

dojo.require("iwc.widget.calendar.QuickEventBalloon");

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

```

8. Restart the Oracle WebLogic Server, clear the browser cache and reload the page to see the change.

## Removing the Attachment Button in the New Task Tab

To remove the **Attachment** button in the **New Task** tab:

1. Verify that the `c11n_Home` directory exists. If `c11n_Home` does not exist, create it.
2. Verify that customization is enabled in Convergence. If customization is disabled, enable it.
3. In `c11n_Home`, verify that the following directories exist. If they do not exist, create them:

```

/c11n_Home/allDomain/js/widget/

```

4. In `c11n_Home`, verify that `config.js` exists. If it does not exist, create it.
5. Modify `config.js` so that it enables JavaScript customization (`jsEnabled: true`) across all domains (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
 }
}

```

6. In `c11n_Home/allDomain/js`, create or modify `customize.js` (the domain specific customization) to include the following code:

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

// Remove the Attachment Button in a New Task Tab
dojo.require("c11n.allDomain.js.widget.calendar.TaskDetail");
```

7. In *c11n\_Home/allDomain/js/widget/calendar*, create the JavaScript file (**TaskDetail.js**) that removes the **Attachment** button from the **New Task** tab

```
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: ""

});
```

8. Restart the Oracle WebLogic Server, clear the browser cache and reload the page to see the change.

## Removing Reservations from the New Event Tab

To remove the reservation option in the **New Event** tab:

1. Verify that the *c11n\_Home* directory exists. If *c11n\_Home* does not exist, create it.
2. Verify that customization is enabled in Convergence. If customization is disabled, enable it.
3. In *c11n\_Home*, verify that the following directories exist. If they do not exist, create them:

```
/c11n_Home/allDomain/js/widget/
```

4. In *c11n\_Home*, verify that **config.js** exists. If it does not exist, create it.
5. Modify **config.js** so that it enables JavaScript customization (**jsEnabled: true**) across all domains (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
 }
}
```

6. In *c11n\_Home/allDomain/js*, create or modify **customize.js** (the domain specific customization) to include the following code:

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

```
// Remove Reservations from the New Event Tab
dojo.require("c11n.allDomain.js.widget.calendar.CreateEvent");
```

7. In `c11n_Home/allDomain/js/widget/calendar`, create the JavaScript file (**CreateEvent.js**) that removes the **Reservations** button and label from the **New Event** tab.

```
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");

 },

 last: ""

});
```

8. Restart the Oracle WebLogic server, clear the browser cache and reload the page to see the change.

## Disabling Calendar Event Notification by SMS

The following example removes the SMS Option in the Calendar Notifications Options tab and from the New Event Reminder dialog.

1. Verify that the `c11n_Home` directory exists. If `c11n_Home` does not exist, create it.
2. Verify that customization is enabled in Convergence. If customization is disabled, enable it.
3. In `c11n_Home`, verify that the following directories exist. If they do not exist, create them:

```
/c11n_Home/allDomain/js/widget/
```

4. In `c11n_Home`, verify that **config.js** exists. If it does not exist, create it.
5. Modify **config.js** so that it enables JavaScript customization (**jsEnabled: true**) across all domains (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

 }
}
```

6. In `c11n_Home/allDomain/js`, create or modify **customize.js** (the domain specific customization) to include the following code:

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

// Remove the SMS Option from the possible Calendar Notifications
```

```
dojo.require("c11n.allDomain.js.widget.calendar.option.Notification");
dojo.require("c11n.allDomain.js.widget.calendar.NotificationDialog");
```

7. In `c11n_Home/allDomain/js/widget/calendar/option`, create the JavaScript file (**Notification.js**) that removes SMS notification in Calendar Options.

```
dojo.provide("c11n.allDomain.js.widget.calendar.option.Notification");
dojo.require("iwc.widget.calendar.option.Notification");
dojo.declare("iwc.widget.calendar.option.Notification",
iwc.widget.calendar.option.Notification, {

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

 dojo.style(this.notificationSMS.domNode.parentNode, "display",
"none");

 },

 last: ""

});
```

8. In `c11n_Home/allDomain/js/widget/calendar`, create the JavaScript file (**NotificationDialog.js**) that removes SMS notification from the Notification dialog box.

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

dojo.require("iwc.widget.calendar.NotificationDialog");

dojo.declare("iwc.widget.calendar.NotificationDialog",
iwc.widget.calendar.NotificationDialog, {

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

 this.methodSelector.removeOption("sms");
 this.methodSelectorAbs.removeOption("sms");
 },

 last: ""

});
```

9. Restart the Oracle WebLogic Server, clear the browser cache and reload the page to see the change.



# 8

## Convergence Address Book Customization Examples

This chapter provides several examples for customizing the address book service in Oracle Communications Convergence.

### Customization Requirements

To perform any Convergence customization, you should have expert knowledge in dojo and knowledge of UI customization. For more information on general Convergence customization, see "[Technical Overview](#)".

Nearly all customization examples require the same setup and preparation.

- You start by verifying that the `c11n_Home` directory exists. If it does not exist, create it. See "[Customization Directory Structure](#)" for more information.
- Next, you verify that customization is enabled in Convergence. If customization is disabled, enable it. See "[Enabling Customization in the Convergence Server](#)" for more information.
- When you have completed your customization, you must restart the Oracle WebLogic Server for Convergence. See *Convergence System Administrator's Guide* for more information.

### Changing the Corporate Directory Name

By default, when a user selects the corporate address book, the **Corporate Directory** tab appears in the Convergence UI.

If the address book service is provided by Convergence, you can customize the tab name when the user selects the corporate address book.

If the address book service is provided by Oracle Communications Contacts Server, see the Contacts Server documentation for information about changing the name of the corporate address book. Because the display name of the corporate address book is being provisioned by Contacts Server, it is not possible to localize the name in Convergence.

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.

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 and new directory entries without having to change the *displayName* entries in *o=PiServerDb* for every user.

To change the name Corporate Directory:

1. Verify that the *c11n\_Home* directory exists. If *c11n\_Home* does not exist, create it.
2. Verify that customization is enabled in Convergence. If customization is disabled, enable it.
3. In *c11n\_Home*, verify that the following directories exist. If they do not exist, create them:

```
/c11n_Home/allDomain/nls/
```

4. In *c11n\_Home*, verify that **config.js** exists. If it does not exist, create it.
5. Modify **config.js** so that it enables i18n customization (**i18nEnabled: true**) across all domains (module: "**allDomain**").

```
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
 }
}
```

6. In *c11n\_Home/allDomain/nls*, create a default resource file named **resources.js**.

This customized resources file inherits the default values in the standard resources file and extends them with the new, custom values. At run time, for each domain that has *i18nEnabled* enabled, Convergence loads that domain's i10n 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.

7. In **resources.js**, remove any references to corporate directory (for example, *corp\_dir*, *the\_corporate\_directory*, *corporate\_lookup*, or *corp\_dir\_lookup*).
8. In **resources.js**, add the following:

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

 last: ""
}
```

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

## Displaying Additional Address Book Attributes When Adding Contacts to an Invitation

By default, when inviting contacts to an event from the corporate directory, contacts are listed by Display Name (LDAP attribute "cn") and Email (LDAP attribute "mail").

You can customize the Add Contacts dialog box to display additional columns of information.

This example shows how to add Job Title (LDAP attribute "title") and Department (LDAP attribute "ou") to the list of attributes when inviting contacts from the corporate directory to an event.

 **Note:**

The LDAP attribute to Convergence attribute mapping for Corporate Directory is in this file: `/var/opt/sun/comms/iwc/config/templates/ab/corp-dir/xlate-inetorgperson.xml`

The LDAP attribute to Convergence attribute mapping for Personal Address Book is in this file: `/var/opt/sun/comms/iwc/config/templates/ab/ldappstore/xlate-piTypePerson.xml`

1. Verify that the `c11n_Home` directory exists. If `c11n_Home` does not exist, create it.
2. Verify that customization is enabled in Convergence. If customization is disabled, enable it.
3. In `c11n_Home`, verify that the following directories exist. If they do not exist, create them:

```
/c11n_Home/allDomain/js/widget/addressBook
```

4. In `c11n_Home`, verify that `config.js` exists. If it does not exist, create it.
5. Modify `config.js` so that it enables JavaScript customization (`jsEnabled: true`) across all domains (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
 }
}
```

6. In `c11n_Home/allDomain/js`, create or modify `customize.js` (the domain specific customization) to include the following code:

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

dojo.require("c11n.allDomain.js.widget.addressBook._FilteringQuerierMixin");
dojo.require("c11n.allDomain.js.widget.addressBook.BookStoreItemSelector");
```

7. In `c11n_Home/allDomain/js/widget/addressBook`, create or modify `_FilteringQuerierMixin.js` to include the following code:

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

dojo.require("iwc.widget.addressBook._FilteringQuerierMixin");

dojo.setObject("iwc.widget.addressBook._FilteringQueryCellTypes", {
 "entry/displayname": {
```

```

 l10nKey: "display_name",
 field: "entry/displayname",
 formatter: function(value){
 // displayname could be an array in corporate.
 return dojo.isArray(value) ?
iwc.util.encodeXMLEntities(value[0]) : value;
 },
 width: "25%",
 noresize: true
 },
 "organization/title": {
 l10nKey: "job_title",
 field: "organization/title",
 formatter: function(value){
 return dojo.isArray(value) ?
iwc.util.encodeXMLEntities(value[0]) : value;
 },
 width: "25%",
 noresize: true
 },
 "organization/organizationalunit": {
 l10nKey: "department",
 field: "organization/organizationalunit",
 formatter: function(value){
 return dojo.isArray(value) ?
iwc.util.encodeXMLEntities(value[0]) : value;
 },
 width: "25%",
 noresize: true
 },
 "email": {
 l10nKey: "email",
 field: 'email',
 formatter: function(value){
 // email is multivalued... will return an array
 return value === undefined ? "" :
iwc.util.encodeXMLEntities(value[0].content);
 },
 width: "25%",
 noresize: true
 }
});

```

8. In `c11n_Home/allDomain/js/widget/addressBook/BookStoreItemSelector.js`, create or modify `BookStoreItemSelector.js` to include the following code:

```

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

dojo.declare("iwc.widget.addressBook.BookStoreItemSelector",
iwc.widget.addressBook.BookStoreItemSelector, {
 displayColumns: ["entry/displayname", "email", "organization/title",
"organization/organizationalunit"],
 last: ""
});

```

9. Restart the Oracle WebLogic Server and clear the browser cache.

## Displaying Additional Address Book Attributes When Adding Resources to an Invitation

By default, when inviting resources to an event from the corporate directory, resources are listed by Display Name (LDAP attribute "cn") and Email (LDAP attribute "mail").

You can customize the Add Resources dialog box to display additional columns of information.

This example builds on the example "[Displaying Additional Address Book Attributes When Adding Contacts to an Invitation](#)".

This example shows how to add Job Title (LDAP attribute "title") and Department (LDAP attribute "ou") to the list of attributes when inviting resources from the corporate directory to an event.

1. In `c11n_Home/allDomain/js`, create or modify **customize.js** (the domain specific customization) to include the following code in bold:

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

dojo.require("c11n.allDomain.js.widget.addressBook._FilteringQuerierMixin");
dojo.require("c11n.allDomain.js.widget.addressBook.BookStoreItemSelector");
dojo.require("c11n.allDomain.js.widget.addressBook.ResourceStoreItemSelector");
```

2. In `c11n_Home/allDomain/js/widget/addressBook`, create or modify **ResourceStoreItemSelector.js** to include the following code:

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

dojo.require("iwc.widget.addressBook.ResourceStoreItemSelector");

dojo.declare("iwc.widget.addressBook.ResourceStoreItemSelector",
iwc.widget.addressBook.ResourceStoreItemSelector, {
 displayColumns: ["entry/displayname", "email", "organization/title",
"organization/organizationalunit"],
 last: ""
});
```

3. Restart the Oracle WebLogic Server and clear the browser cache.

## Removing the Copy To Button

You can limit a user's ability to copy corporate address book contacts to their personal address book by removing the Copy To button from the Address Book UI.

1. Verify that the `c11n_Home` directory exists. If `c11n_Home` does not exist, create it.
2. Verify that customization is enabled in Convergence. If customization is disabled, enable it.
3. In `c11n_Home`, verify that the following directories exist. If they do not exist, create them:

```
/c11n_Home/allDomain/js/widget/addressBook/
```

4. In `c11n_Home`, verify that **config.js** exists. If it does not exist, create it.
5. Modify **config.js** so that it enables JavaScript customization (**jsEnabled: true**) across all domains (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

}
}
```

6. In `c11n_Home/allDomain/js`, create or modify **customize.js** (the domain specific customization) to include the following code:

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

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

7. In `c11n_Home/allDomain/js/widget/addressBook`, create or modify **\_BookBrowserToolbar.js** to include the following code:

```
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: ""

});
```

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

## Removing the Google Maps Link

By default, the Convergence personal address book automatically creates a link to Google Maps when a contact profile includes an address.

To remove the link to Google Maps:

1. Verify that the `c11n_Home` directory exists. If `c11n_Home` does not exist, create it.
2. Verify that customization is enabled in Convergence. If customization is disabled, enable it.
3. In `c11n_Home`, verify that the following directories exist. If they do not exist, create them:

```
/c11n_Home/allDomain/js/service/
/c11n_Home/allDomain/js/widget/addressBook/
```

4. In `c11n_Home`, verify that **config.js** exists. If it does not exist, create it.

5. Modify **config.js** so that it enables JavaScript customization (**jsEnabled: true**) across all domains (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
 }
}
```

6. In **c11n\_Home/allDomain/js**, create or modify **customize.js** (the domain specific customization) to include the following code:

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

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

7. In **c11n\_Home/allDomain/js/widget/addressBook**, create or modify **\_DisplayContactBody.js**. Find the **\_DisplayContactBody** method and replace it with the following:

```
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: ""
});
```

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

## Importing or Exporting Address Book Information in a Custom Language

Unless the Convergence address book is provided by Contacts Server, you can customize Convergence so that users can export and import address book information in a custom language.

This example assumes you have completed the example ["Adding a New Language in Convergence"](#).

1. Configure Convergence to export address book data in the custom language:

- a. In `/var/opt/sun/comms/iwc/config/templates/ab/export`, create a file named **export-csv-language.xls**, where *language* is language code for your custom language. For example, if your custom language is Slovak, create **export-csv-sk.xls**.

You can create this file by copying and renaming another CSV file with the following command:

```
diff export-csv-us.xls export-csv-language.xls
```

- b. Modify **export-csv-language.xls** with column labels in the custom language.

Convergence can now export address book data in CSV format in the custom language.

2. Configure Convergence to import address book data in the custom language:

- a. In `/var/opt/sun/comms/iwc/config/templates/ab/import`, modify **headrXlations.orig** to add mappings and translations for the custom language:

```
diff headrXlations.orig headrXlations
```

For example, if the custom language is Slovak:

```
> [sk]
> sk-First Name=First Name
> sk-Last Name=Last Name
> sk-Middle Name=Middle Name
> sk-Name=Name
> sk-Nick Name=Nick Name
> sk-Email=E-mail Address
```

- b. In `/var/opt/sun/comms/iwc/config/templates/ab/import`, create **xlate-csvlanguage.xml**, where *language* is the custom language. For example, **xlate-csvsk.xml**:

You can create this file by copying and renaming another XML file with the following command:

```
diff xlate-csvus.xml xlate-csvlanguage.xml
```

- c. In `/var/opt/sun/comms/iwc/config/templates/ab/import`, modify **import.properties** to include the following code:

```
import.csvlanguage.class = com.sun.comms.client.ab.importexport.ImportedCsvParser
import.csvlanguage.xlatePath = csvlanguage
import.csvlanguage.encoding = ISO-8859-1
language.charset=ISO-8859-1
```

For example, if the custom language is Slovak:

```
import.csvsk.class = com.sun.comms.client.ab.importexport.ImportedCsvParser
import.csvsk.xlatePath = csvsk
import.csvsk.encoding = ISO-8859-1
sk.charset=ISO-8859-1
```

Convergence can now import address book data in CSV format in the custom language.



# 9

## Convergence Options Customization Examples

This chapter provides several examples for customizing the options in Oracle Communications Convergence.

### Customization Requirements

To perform any Convergence customization, you should have expert knowledge in dojo and knowledge of UI customization. For more information on general Convergence customization, see "[Technical Overview](#)".

Nearly all customization examples require the same setup and preparation.

- You start by verifying that the `c11n_Home` directory exists. If it does not exist, create it. See "[Customization Directory Structure](#)" for more information.
- Next, you verify that customization is enabled in Convergence. If customization is disabled, enable it. See "[Enabling Customization in the Convergence Server](#)" for more information.
- When you have completed your customization, you must restart the Oracle WebLogic Server for Convergence. See *Convergence System Administrator's Guide* for more information.

### Disabling External POP Account Access

The following example removes **External Accounts** from the Options section of Convergence.

This is a useful example when there is a need to restrict users from modifying specific account settings in Convergence.

1. Verify that the `c11n_Home` directory exists. If `c11n_Home` does not exist, create it.
2. Verify that customization is enabled in Convergence. If customization is disabled, enable it.
3. In `c11n_Home`, verify that the following directories exist. If they do not exist, create them:  
`/c11n_Home/allDomain/js/service/`
4. In `c11n_Home`, verify that **config.js** exists. If it does not exist, create it.
5. Modify **config.js** so that it enables JavaScript customization (**jsEnabled: true**) across all domains (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
 }
}

```

6. In *c11n\_Home/allDomain/js*, create or modify **customize.js** (the domain specific customization) to include the following code:

```

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

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

```

7. In *c11n\_Home/allDomain/js/widget/option*, create the JavaScript file (**Navigator.js**) that removes External Accounts.

```

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 External Accounts
 iwc.api.removeOption("/Mail/External Accounts");
 },
 last: ""
 }
);

```

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

## Enabling or Disabling the Modification of Identity Settings

The following example prevents or allows users from editing the values in the **Display Name** field, **Email Address** field, or **Reply-to Address** field on the Options **Local Account** tab. These fields still appear in Convergence, but cannot be edited.

1. Verify that the *c11n\_Home* directory exists. If *c11n\_Home* does not exist, create it.
2. Verify that customization is enabled in Convergence. If customization is disabled, enable it.
3. In *c11n\_Home*, verify that the following directories exist. If they do not exist, create them:

```
/c11n_Home/allDomain/js/widget/
```

4. In *c11n\_Home*, verify that **config.js** exists. If it does not exist, create it.
5. Modify **config.js** so that it enables JavaScript customization (**jsEnabled: true**) across all domains (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

 }
}

```

6. In `c11n_Home/allDomain/js`, create or modify **customize.js** (the domain specific customization) to include the following code:

```

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

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

```

7. Do one of the following:

- To disable the modification of identity settings, in `c11n_Home/allDomain/js/widget/mail/option`, create or modify the JavaScript file (**Identity.js**) that controls Identity settings.

```

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.replyTo.setDisabled(true);

 },

 last: ""

});

```

- To enable the modification of identity settings, in `c11n_Home/allDomain/js/widget/mail/option`, create or modify the JavaScript file (**Identity.js**) that controls Identity settings.

```

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.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: ""
});

```

- Restart the Oracle WebLogic Server and clear the browser cache to see the change.

## Redirecting Users to Another Page to Change Password

This example demonstrates how you can redirect users to another page to their password for Convergence.

- Verify that the `c11n_Home` directory exists. If `c11n_Home` does not exist, create it.
- Verify that customization is enabled in Convergence. If customization is disabled, enable it.
- In `c11n_Home`, verify that the following directories exist. If they do not exist, create them:

```
/c11n_Home/allDomain/js/option/
```

- In `c11n_Home`, verify that `config.js` exists. If it does not exist, create it.
- Modify `config.js` so that it enables JavaScript customization (`jsEnabled: true`) across all domains (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
 }
}

```

- In `c11n_Home/allDomain/js`, create or modify `customize.js` (the domain specific customization) to include the following code:

```

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

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

```

- In `c11n_Home/allDomain/js/widget`, create a JavaScript file (`option/Password.js`) that redirects the change password option to `www.example.com`.

```

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.ViewerTabPage");

dojo.declare("iwc.widget.option.Password", [iwc.widget.option.ViewerTabPage], {
 templateString: '<div><form dojoType="iwc.widget.form.BorderContainerForm"
dojoAttachPoint="form"><iframe src="graphics/www.example.com" width=100%
height=1000px></iframe></form></div>',

 postCreate: function() {

```

```

 dojo.addClass(this.form.buttonNode.domNode, "dijitHidden");
 },

 last: ""

});

```

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

## Removing Change Password, Vacation Message, and Calendar Notification Options

The following example removes **Change Password**, **Vacation Message**, and **(Calendar) Notifications** from the Options section of Convergence.

This is a useful example when there is a need to restrict users from modifying specific account settings in Convergence.

1. Verify that the *c11n\_Home* directory exists. If *c11n\_Home* does not exist, create it.
2. Verify that customization is enabled in Convergence. If customization is disabled, enable it.
3. In *c11n\_Home*, verify that the following directories exist. If they do not exist, create them:

```
/c11n_Home/allDomain/js/service/
```

4. In *c11n\_Home*, verify that **config.js** exists. If it does not exist, create it.
5. Modify **config.js** so that it enables JavaScript customization (**jsEnabled: true**) across all domains (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
 }
}

```

6. In *c11n\_Home/allDomain/js*, create or modify **customize.js** (the domain specific customization) to include the following code:

```

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

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

```

7. In *c11n\_Home/allDomain/js/widget/option*, create the JavaScript file (**Navigator.js**) that removes Change Password, Mail Vacation Message, and Calendar Notifications.

```

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

 // call the superclass postCreate
 this.inherited(arguments);
 },

 last: ""
}
);

```

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

## Removing Default Language List in General Options

On the Options **General** tab, you can remove the **Language** list from the Convergence UI. By removing the **Language** list, users can view Convergence only in the default language configured by the application administrator.

This is a useful example when there is a need to restrict users from modifying specific account settings in Convergence.

1. Verify that the *c11n\_Home* directory exists. If *c11n\_Home* does not exist, create it.
2. Verify that customization is enabled in Convergence. If customization is disabled, enable it.
3. In *c11n\_Home*, verify that the following directories exist. If they do not exist, create them:

```

/c11n_Home/allDomain/js/service/
/c11n_Home/allDomain/js/widget/option

```

4. In *c11n\_Home*, verify that **config.js** exists. If it does not exist, create it.
5. Modify **config.js** so that it enables JavaScript customization (**jsEnabled: true**) across all domains (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
 }
}

```

6. In *c11n\_Home/allDomain/js*, create or modify **customize.js** (the domain specific customization) to include the following code:

```

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

```

7. In `c11n_Home/allDomain/js/widget/option`, overwrite the contents of the global options JavaScript file **General.js** with the following code.

```
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: ""
 }
);
```

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

## Removing Languages from Language List in General Options

On the Options **General** tab, you can remove languages from the **Language** list. By removing languages from the **Language** list, users cannot select them.

This is a useful example when there is a need to restrict users from modifying specific account settings in Convergence.

1. Verify that the `c11n_Home` directory exists. If `c11n_Home` does not exist, create it.
2. Verify that customization is enabled in Convergence. If customization is disabled, enable it.
3. In `c11n_Home`, verify that the following directories exist. If they do not exist, create them:

```
/c11n_Home/allDomain/js/service/
/c11n_Home/allDomain/js/widget/option/
```

4. In `c11n_Home`, verify that **config.js** exists. If it does not exist, create it.
5. Modify **config.js** so that it enables JavaScript customization (**jsEnabled: true**) across all domains (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
 }
}
```

6. In `c11n_Home/allDomain/js`, create or modify `customize.js` (the domain specific customization) to include the following code:

```
dojo.provide("c11n.allDomain.js.customize");
// Hide a few default languages
dojo.require("c11n.allDomain.js.widget.option.General");
```

7. In `c11n_Home/allDomain/js/widget/option`, extend the JavaScript file `option/General.js` with the following code. In this example, the languages `ja`, `ko`, `zh-CN`, and `zh-TW` are removed from the list of available languages in the **Languages** list.

```
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: ""
 }
);
```

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

## Customizing the Default Alert Sounds

Each user can configure their Convergence to play a sound alert when they receive a new email message.

You can customize the sound files that are played to the user.

1. Verify that the `c11n_Home` directory exists. If `c11n_Home` does not exist, create it.
2. Verify that customization is enabled in Convergence. If customization is disabled, enable it.
3. In `c11n_Home`, verify that the following directories exist. If they do not exist, create them:

```
/c11n_Home/allDomain/js/service/
/c11n_Home/allDomain/layout/media
```

4. In `c11n_Home`, verify that `config.js` exists. If it does not exist, create it.
5. Modify `config.js` so that it enables JavaScript customization (`jsEnabled: true`) across all domains (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
 }
};
```



```
 }
}
```

6. In `c11n_Home/allDomain/js`, create or modify `customize.js` (the domain specific customization) to include the following code:

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

dojo.require("c11n.allDomain.js.service.AudioNotification");
```

7. In `c11n_Home/allDomain/layout/media`, place the two custom MP3 files, one for new email and instant messages, and one for incoming and outgoing calls.
8. In `c11n_Home/allDomain/js/service`, create the JavaScript file (`AudioNotification.js`) that specifies the path to the new audio files.

```
dojo.provide("c11n.allDomain.js.service.AudioNotification");

dojo.require("iwc.service.AudioNotification");

dojo.declare("iwc.service.AudioNotification", iwc.service.AudioNotification, {
 mediaPath: "media_path_directory",
 audioAlertFileName: "custom_audio_message.mp3",
 callAlertFileName: "custom_audio_call.mp3"
});
```

Where:

- `media_path_directory` is the customized **media** directory relative to the location of the **main.html** page used for the domain. For example, if the default **main.html** page is used for all domains (`iwc_static/layout/main.html`), `media_path_directory` is `c11n_Home/allDomain/layout/media`.
- `custom_audio_message` is the custom audio MP3 file for new email and instant messages.
- `custom_audio_call` is the custom audio MP3 file for incoming and outgoing voice or video calls.

#### Note:

One of the following steps are also required in WebLogic Server to enable audio alerts:

- Create a file called **mimemappings.properties** under the `<Convergence_Domain>/config` folder and add **mp3=audio/mpeg**.
- Add the following fields in the `<Convergence_Domain>/servers/ManagedServer/tmp/_WL_user/<Convergence>/<value>/war/WEB-INF/web.xml` file:

```
<mime-mapping>
 <extension>mp3</extension>
 <mime-type>audio/mpeg</mime-type>
</mime-mapping>
```

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

# 10

## Custom Convergence Modules

To enhance security in Oracle Communications Convergence, you can write your own custom modules for authentication or single sign-on. See the discussion about security considerations for developers in *Convergence Security Guide* for more information.