Oracle® Cloud

Building Sites with Oracle Content Management

F24096-68
January 2024

Oracle Cloud Building Sites with Oracle Content Management,

F24096-68

Primary Author: Bruce Silver

Contributing Authors: Sarah Bernau, Kalpana N, Ron van de Crommert

Contents

6 Use Site Templates and Themes in Sites

7 Manage Custom Components and Layouts

8 Work with Site Pages

9 Arrange Page Content

10 Use Built-In Components

Part IV Developing for Sites

18 JavaScript Technologies

19 Customize Designs and Styles

20 Understand Background Use

21 Set Triggers and Actions

22 Develop Site Templates

23 Develop Themes

24 Develop Layouts

25 Develop Components

26 Customize the Controller File

Part V Developing for Sites with Other Tools

27 Develop with Oracle Content Management Toolkit

Part VI Appendixes

28 Tutorial: Develop Components with Knockout

29 Troubleshoot

Preface

This document describes how to create and manage experiences, including sites, assets, and all their associated structures and policies with Oracle Content Management.

Audience

This publication is intended for content providers and developers who want to build content-rich, secure websites with Oracle Content Management using a number of intuitive, business-friendly tools, including a visual Site Builder.

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.

Related Resources

For more information, see these Oracle resources:

- Oracle Public Cloud:

  http://cloud.oracle.com

- *What's New for Oracle Content Management*
- *Collaborating on Documents with Oracle Content Management*
- *Administering Oracle Content Management*
- *Developing with Oracle Content Management As a Headless CMS*
- *Integrating and Extending Oracle Content Management*
- *Known Issues for Oracle Content Management*
- *Getting Started with Oracle Cloud*

Conventions

The following text conventions are used in this document.

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Part I
Introduction

This part provides conceptual information about building sites with Oracle Content Management and includes an easy-to-use tutorial for creating your first site. It includes the following chapters:

- Overview of Oracle Content Management
- Get Started with Building Sites
- Create Your First Website

1

Overview of Oracle Content Management

Whether you need to manage digital assets, publishing to multiple channels in various languages, or oversee business documents gathered from a variety of sources, Oracle Content Management helps you throughout the entire content lifecycle. Create, capture, organize, review, and protect all your content as it flows through your organization with integrated processes and data. Oracle Content Management is a cloud-based content hub, offering scalability, security, and governance, so you can eliminate the typical inefficiencies in content management—including organizing and tagging new content and locating existing documents—and do more with fewer resources.

Using Oracle Content Management for digital asset management, you can rapidly collaborate internally and externally on any device to approve content and create contextualized experiences. Built-in business-friendly tools make building new web experiences with stunning content a breeze. You can drive digital engagement with all your stakeholders using the same content platform and the same processes. Technical and organizational bottlenecks are gone, so you no longer have barriers to create engaging experiences, improving customer and employee engagement.

Using Oracle Content Management for business document management, you have the same collaboration capabilities internally and externally on any device to manage your content. Integrated tools such as content connectors enable you to upload content from third-part cloud storage, and Content Capture makes it easy to *automate* document discovery and capture.

> **Note:**
>
> Oracle Content Management Starter Edition has a limited feature set. To take advantage of the full feature set, upgrade to the Premium Edition.

Access Oracle Content Management

After you've been granted access to Oracle Content Management, you receive a welcome email with details about the instance URL and your user name. You'll need this information to log in to the service, so it's a good idea to keep it for future reference.

There are different ways to interact with Oracle Content Management:

- The web interface provides easy access from your favorite web browser. You can manage your content in the cloud, share files and folders with others, start and participate in conversations, create websites (if allowed), and more.
- The desktop app lets you keep your files and folders synchronized between the cloud and your computer. You can sync your own files and those shared with you, making sure you always have access to the latest versions.
- A Microsoft Office add-on gives you access to Oracle Content Management features directly from Microsoft Word, Excel, PowerPoint, and Outlook.

- Mobile apps for Android and iOS provide easy access on your phone or other mobile devices. The mobile apps are instantly familiar, because they look and act just like the service in your web browser. You can access your cloud content, search and sort your files and folders, share content, and work with conversations.
- REST APIs and SDKs provide developers with powerful tools to programmatically incorporate Oracle Content Management functionality into web applications and mobile apps.

Understand Roles

The Oracle Content Management features that you can access depend on the role you've been assigned. You'll see different options depending on your application role. Standard users can work with documents, conversations, and sites. Enterprise users can also access assets. Developers see options to build and customize website pieces such as templates, themes, components, and layouts. Administrators see options to configure the service, integrate the service with other business applications, and set up asset repositories.

There are different types of roles in Oracle Content Management:

- **Organization roles** — Your role within your organization determines what tasks you need to perform and how you use features.
- **Application roles** — Application roles control what features you see in Oracle Content Management.
- **Resource roles** (permissions) — What you can see and do with a resource, such as a document, content item, site, or template, depends on the role you're assigned when the resource is shared with you.

Learn more...

Manage Assets

Oracle Content Management offers enterprise users powerful capabilities to manage all your assets whether you need to manage digital assets, publishing to multiple channels in various languages, or oversee business documents gathered from a variety of sources. It provides a central content hub for all your assets, where you can organize them into repositories and collections, and create rules to define how they can be used and where.

There are also extensive management and workflow features to guide assets through their creation and approval process and to ensure that only authorized versions are available for use.

It's easy to tag and filter assets so you can quickly find the assets you need. And smart content features will tag and suggest assets automatically as you use them!

Create asset types to define what information you need to collect when users create assets. *Digital asset types* define the custom attributes required for your digital assets (files, images, and videos) and business documents. *Content types* group different pieces of content into reusable units. Users can then create digital assets, business documents, and content items based on these asset types for consistent use.

Learn more...

Collaborate on Documents

With Oracle Content Management, you can manage your content in the cloud, all in one place and accessible from anywhere.

You can group your files in folders and perform common file management operations (copy, move, delete, and so on) in much the same way as on your local computer. And since all your files reside in the cloud, you have access to them wherever you go, also on your mobile devices. If you install the desktop app, all your content can be automatically synchronized to your local computer, so you always have the most recent versions at your fingertips.

After you get all your content in the cloud, it's easy to share your files or folders to collaborate with others inside or outside your organization. Everyone you share your content with has access to the latest information—wherever they are, whenever they need it. You can grant access to entire folders or provide links to specific items. All access to shared items is recorded, so you can monitor how and when each shared item was accessed.

Conversations in Oracle Content Management allow you to collaborate with other people by discussing topics and posting comments in real time. You can start a stand-alone conversation on any topic, adding files as needed. Or you can start a conversation about a specific file, folder, asset, or site for quick and easy feedback.

All messages, files, and annotations associated with a conversation are retained, so it's easy to track and review the discussion. And your conversations live in the cloud, so you can also view them and participate on the go from your mobile devices.

Learn more...

Build Sites

With Oracle Content Management, you can rapidly build and publish marketing and community websites—from concept to launch—to provide engaging online experiences. The process is completely integrated: content, collaboration, and creativity are combined in a single authoring and publishing environment.

To get started quickly, use an out-of-the-box template, drag-and-drop components, sample page layouts, and site themes to assemble a site from predefined building blocks. Or developers can create custom templates, custom themes, or custom components to create unique online experiences.

Add YouTube videos, streaming videos, images, headlines, paragraphs, social media links, and other site objects simply by dragging and dropping components into designated slots on a page. Switch themes and rebrand a site at the touch of a button to provide an optimized, consistent look and feel across your organization.

You can work on one or more updates, preview an update in the site, and then, when you're ready, publish the update with a single click.

In addition to creating and publishing sites in Site Builder, Oracle Content Management also supports 'headless' site development using REST APIs, React JS, Node JS, and other web technologies.

Learn more...

Integrate and Extend Oracle Content Management

As an Oracle Platform-as-a-Service (PaaS) offering, Oracle Content Management works seamlessly with other Oracle Cloud services.

You can embed the web UI into your web applications so users can interact with content directly. Use the Application Integration Framework (AIF) to integrate third-party services and applications into the Oracle Content Management interface through custom actions. Or develop content connectors to bring content that you have already created elsewhere into Oracle Content Management, manage it centrally, and use it in new experiences across multiple channels.

With a rich set of REST APIs and SDKs for content and site management, delivery, and collaboration, you can incorporate Oracle Content Management functionality into your web applications.

Create client applications that interact with your content SDKs and assets in the cloud. Develop custom integrations with collaboration objects or retrieve assets for use wherever you need them. You can access and deliver all your content and assets optimized for each channel, whether it’s through a website, content delivery network (CDN), or mobile apps.

Learn more...

Get Started

To help you get started with Oracle Content Management, visit the Oracle Help Center, which has lots of resources, including documentation, videos, guided tours, and developer information.

And if you need it, there’s support and a community to help.

Migrate to Oracle Cloud Infrastructure

If your Oracle Content Management subscription isn't already running on Oracle Cloud Infrastructure (OCI) with the Infrastructure Console, then Oracle recommends that you migrate to that native OCI environment. This will ensure that you'll enjoy the benefits and advances of Oracle's cloud platform in the future.

Migration is not automatic; you’ll need to submit a service request to initiate the process.

Learn more...

2

Get Started with Building Sites

Oracle Content Management is a cloud-based content hub to drive omni-channel content management and accelerate experience delivery. With Oracle Content Management, you can rapidly collaborate internally and externally on any device to approve content, manage digital assets, and create contextualized experiences using built-in business-friendly tools.

- Overview of Oracle Content Management
- Get Started with Sites
- Understand the Site Creation Process
- Learn About Site Development
- Understand Site Governance
- Create Your First Website

 Video

Get Started with Sites

Anyone with the proper permissions is capable of building a website with Oracle Content Management. You don't need to use any proprietary tools or code or software. The user interface is graphical, intuitive, and friendly.

> **Note:**
>
> With Oracle Content Management Starter Edition, you are limited to 1 site and no site governance. For a full feature set, upgrade to Oracle Content Management Premium Edition.

Who is able to build a site depends on several factors:

- Whether your service administrator has enabled site creation.
- Whether your service administrator has enabled site governance. See Understand Site Governance
- Whether site creation is limited to site administrators (available when site governance is disabled).
- Whether your site administrator has made templates available.

When you create a site, you begin with a template. A template has everything you need to get started with your site, including the site code framework, a default site with sample pages and content, a theme with styling, resources such as images, and even custom components. See Understand the Site Creation Process.

If you're creating an enterprise site, it will be associated with a repository and must have a localization policy defined specifying a default language. Both the repository and the

localization policy must be created before you create a site. You store the assets and documents you need for the site in the repository, and the repository policies dictate what can be done with the assets. See Understand Asset Repositories for details about how to use repositories.

When you edit a site, you create a new update or use an existing update. Within an update, you can edit and add content, adjust the style settings, add and delete pages, change page layouts, and organize pages. See Get to Know the Site Builder Page to find out what you can do with the editor.

Updates don't need to be ready all at once. You and your team members can work on multiple updates concurrently and independently. For example, you might be working on an update with weekly news while another team member is adding pages for an upcoming sales conference. You can edit, review, and save changes to your updates as often as you need to and merge an update with the base site at any time.

When you're ready, launch your site. Just a single click publishes your site to the web. That's it in a nutshell—from concept to launch.

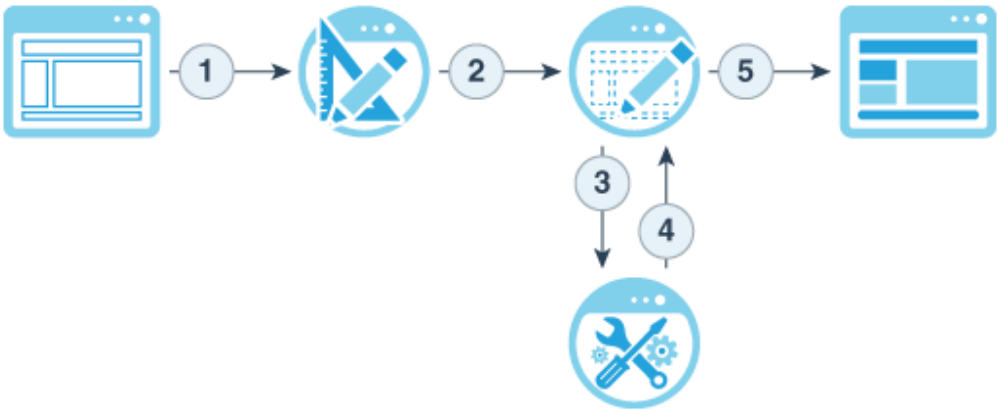
Understand the Site Creation Process

Let's look at the overall process to create and publish a website.

Before you get started, your service administrator should have enabled site creation, configured who can create sites, and installed and made available the templates you'll use to create sites. The service administrator must assign you at least Downloader permissions to the templates. See Learn About User Roles, Get Started with Sites and Understand Site Governance.

If you'll be creating an enterprise site, which enables the use of assets, multilingual sites, and site security taxonomies, it will be associated with a repository. You store the assets and documents you need for the site in the repository, and the repository policies dictate what can be done with the assets. The repository and any security taxonomies must be created before you create an enterprise site. See Understand Asset Repositories and Manage Sites in a Shared Repository.

Here's an overview of the site creation process, with links to more information about the details.



1. **Select a template and name the site**: If site governance is enabled and your site requires approval, you'll have to wait for it to be approved before you can complete the next step. The site remains offline until you're ready to publish it.
2. **Create a site**: This creates an empty framework that you can customize.
3. **Create a new update or use an existing update**: An update is a named collection of changes to the current base site. Each time you view or edit a site in the editor, you use an update. See Understand Site Updates.
4. **Open the site in the editor**: Use the drag-and-drop editor to add pages and content to your site. See Understand Site Updates.
    - **Add pages to the site**: Select a predefined layout to quickly define the type of page. See Add Pages.
    - **Add components to a page**: Drag-and-drop text, images, documents, and more onto the page. See Use Built-In Components.
    - **Change the content or properties of a component**: Fine-tune the spacing, alignment, and other properties for components. See Use Styles and Formatting.
    - **Optionally switch the layout used for a page**: Change the page design on the fly. See Change the Page Layout.
    - **Optionally switch the theme used for the site**: Quickly change the look and feel of the entire site. See Use Site Templates and Themes in Sites.
    - **Set search engine properties**: Help search engines find your site to increase traffic. See Set Search Engine Properties.
5. **Apply a site update**: Update the site with the changes in an update. See Publish Site Changes.
    - **Share the site with other team members**: Share the site with specified users and assign each of them a role to determine what they can do with the site. See Understand Site Security for information about sharing.
    - **Set site security**: Choose which users can access your published site. See Change Site Security.
    - **Publish the site:** Publish the changes to the site. See Publish Site Changes.
    - **Bring the site online**: Make the site available to users based on your site security. See Bring a Site Online or Take It Offline.

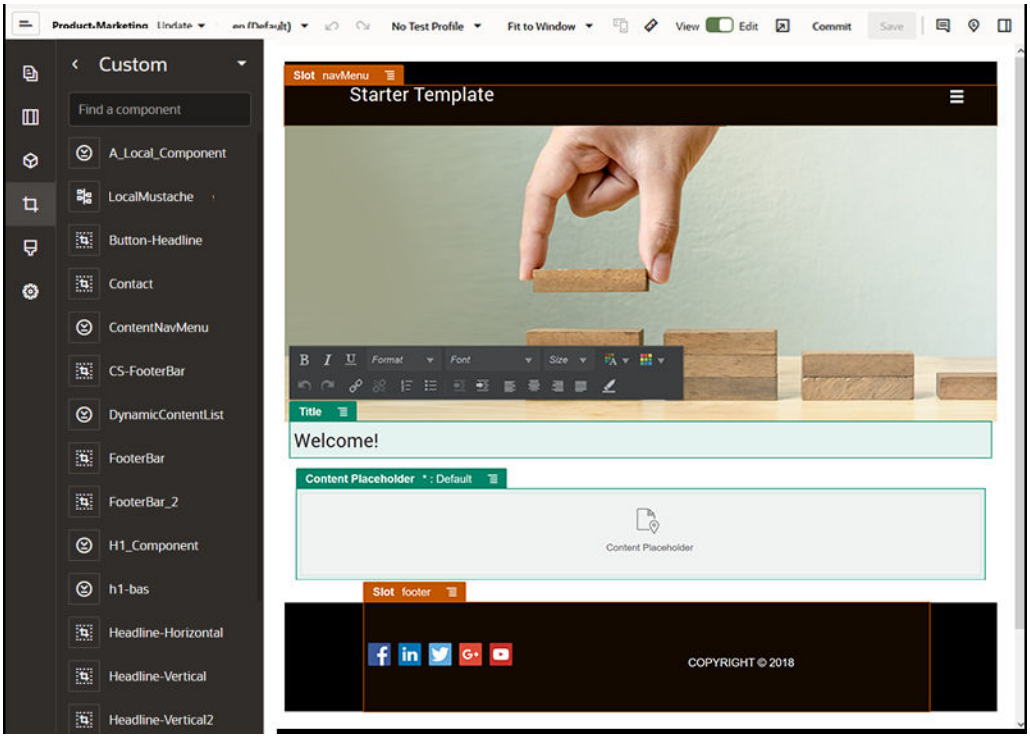
Learn About Site Development

As a site developer, you define the framework that site creators use to build sites, such as:

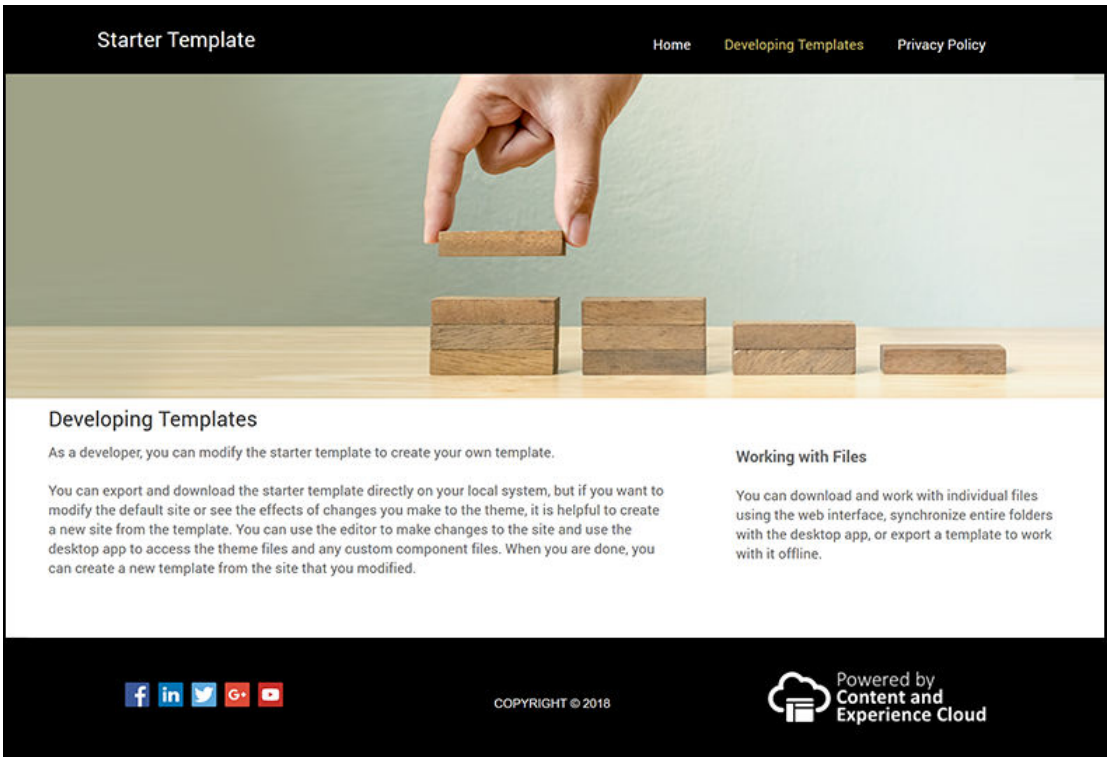
- standard templates, used as a framework for a site, based on a theme, with sample pages and content, custom components, and other resources you need to start building a site
- themes, used to define the overall look and feel of a site, consisting of logos, style sheets, configuration files and background code that defines site navigation
- custom components, used to add specific types of content to site pages, enabling you to develop compound elements that can be embedded within a site page, using any page technology of your choice

By separating site presentation from site content, you ensure that any site created maintains the standards and branding of your organization, and you free content creators to focus on content, making development and site creation more efficient and effective.

Here is an example of a simple template a content creator can use when adding content to a site. Note the generic placeholders layed out on the page:



Here is an example of what a site built using a simple template could look like when previewed after content is added. Note how the placeholders have been customized with specific content and navigation elements:



In addition, site developers create and maintain style sheets, build sites through the web interface or using the Content Toolkit, and configure integrations between Oracle Content Management and other services. And, like any other employee, they also collaborate with others by sharing content, starting or participating in conversations, or using the desktop or mobile apps.

Developers must be assigned the standard user or enterprise user role to be able to use Oracle Content Management. Developers with the standard user role can create components, themes, and standard templates. Developers with the enterprise user role can also create layouts and save a site as a standard or enterprise template.

Beyond site development, Oracle Content Management can also be used in a headless environment as a powerful and flexible back-end content management system (CMS) in the cloud.

Understand Site Governance

Site governance makes it easy for business users to create sites that conform to company policy and gives site administrators an easy way to control and track sites from a centralized location.

> **Note:**
>
> With Oracle Content Management Starter Edition, you are limited to 1 site and no site governance. For a full feature set, upgrade to Oracle Content Management Premium Edition.

The Problem

There's a problem in the industry today. Companies are suffering from experience explosion. These experiences need to be created and deployed quickly, with central visibility to keep with company branding and messaging. Enterprises often find it takes forever to develop and deliver new experiences. To be successful, your company needs powerful and adaptable technology so you can scale, with speed, to create and manage many experiences consistently—and propel product or brand initiatives. You need to quickly and efficiently make new experiences available, or you'll miss market opportunities, and not engage customers and prospects optimally.

Experience explosion is often the result of years of separate, siloed initiatives, during which each division, brand, and team acquired web content management (WCM) systems without central oversight. This creates a complicated environment that is slow to market, vulnerable to security threats, and inflicts spiraling and unpredictable costs.

A lack of consistent processes for building experiences along with the use of a disparate mix of WCM technologies exposes your organization to massive security risks. When multiple stakeholders each own a little piece of the website problem, core responsibilities, such as security, become fragmented, leaving your enterprise vulnerable. All these experiences delivered from different WCM systems with different web application and IT infrastructure oversight make it difficult, if not impossible, to ensure everything is properly protected.

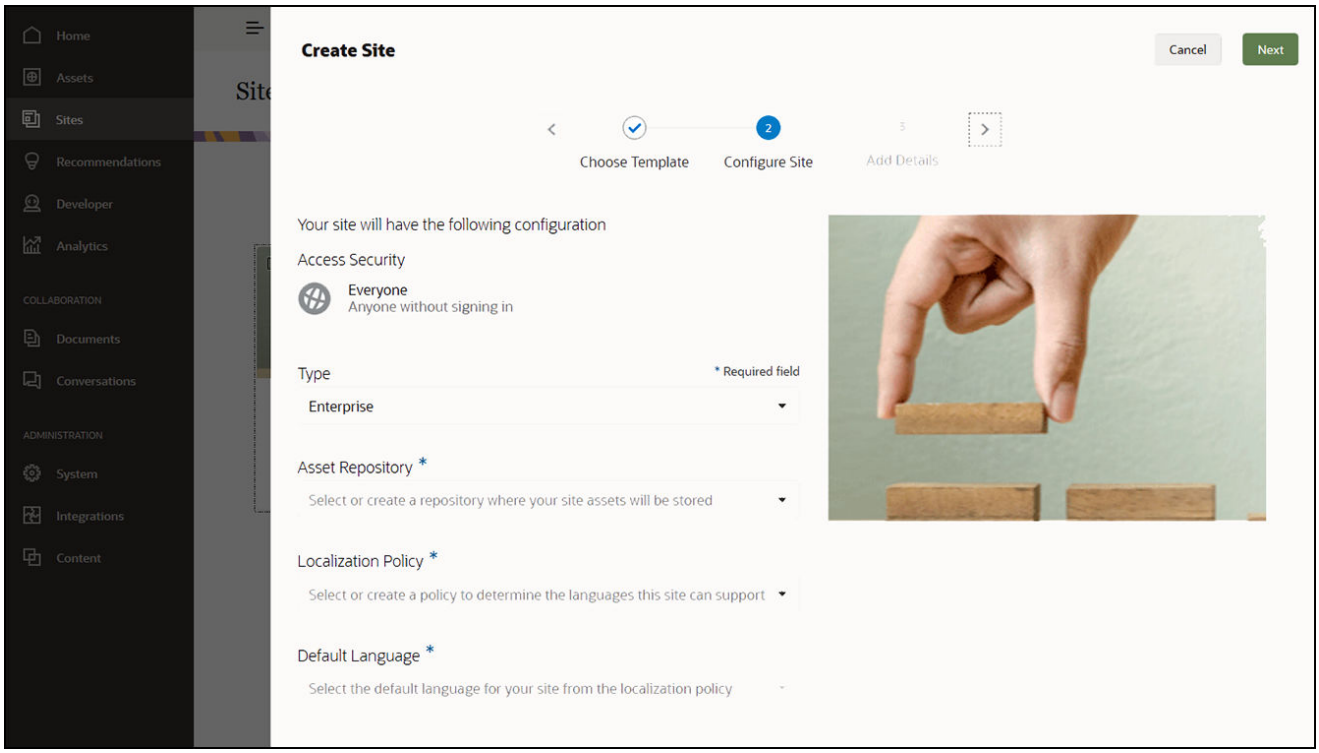
Plus, maintaining multiple high-priced commercial WCM systems results in duplicative costs—a gift that keeps on giving every year with license renewal fees and ongoing support costs from internal or external partners.

This leads to constraints on everyone:

- Constraints on the business:
  - No self-service; reliant on IT or expensive outsourcing
  - Inability to make updates without technical help
  - No business-friendly tools to manage the effort
  - No central visibility into all the experiences and activity
- Constraints on IT:
  - IT involvement needed for every experience
  - Lack of governance over experiences created by the business, including unmonitored outsourced experiences
  - Fragmented stacks used to build experiences
  - Must manage and deploy changes to content and layouts across hundreds of experiences
- Constraints on users:
  - Inconsistent messaging across channels
  - Out-of-date information
  - Poor performance, leading to channel abandonment

The Solution

Oracle Content Management's governance simplifies and accelerates experience delivery for business users while giving your IT departments an easy way to control and track experiences from a centralized location with the ability to fully manage the entire experience lifecycle, reducing the cost to create and maintain each new experience a company needs. Governance is built into the core of Oracle Content Management; it just needs to be enabled by your administrator. See Configure Sites and Assets Settings in *Administering Oracle Content Management*.



Features

- **Catalog of approved templates**—IT developers can populate a template catalog with a set of site templates for the needs of different lines of business. They can limit which templates are available and to whom they're available. They can apply template policies to specify the type of security new sites must adhere to and whether new sites require approval. This allows for a fast and simple way to request new sites while ensuring that business users follow brand and security guidelines. See Change Site Template Status or Audience and Change Site Template Policies.
- **Streamlined requests, approvals, and provisioning**—Give business users the ability to rapidly request new sites with required approvals and automated provisioning. See Create Sites and Manage Site Requests.
- **Site management**—IT departments can manage all sites from one place regardless of who created and deployed the site. IT users can monitor site status and change status for any deployed site. See Manage Sites and Site Settings.

Benefits

When you use governance, you'll see a reduction in cost to create and maintain each new experience.

- Benefits for the business:
  - Rapid provisioning without dependence on IT
  - Separation of content from design, enabling reuse
  - Manage experiences, users, and permissions globally in one console
  - Gain insights into experience operations via reports
- Benefits for IT:
  - No complex implementation (cloud-native solution)
  - Auto-scaling to deal with growth and seasonal peaks
  - Governance over experiences, ensuring they are secure and meet corporate branding and compliance standards
- Benefits for the user:
  - Consistent messages across channels
  - Up-to-date information
  - Optimal performance
  - Secure experience

The Process

Here are the steps involved with creating and managing sites using site governance:

1. Your administrator enables governance. See Configure Sites and Assets Settings in *Administering Oracle Content Management*.
2. A site administrator makes approved templates available to users for creating sites. See Change Site Template Status or Audience and Change Site Template Policies.
3. A user creates a site request from an approved template. See Create Sites.
4. The site administrator approves the site request and the site is automatically created. The site administrator can also deny the site request with a note explaining why the request was denied, and the site creator can correct the issue and submit the request again. See Manage Site Requests.
5. The site creator continues the site creation process, editing and publishing the site. See Understand the Site Creation Process.
6. Ongoing site management and updates can be done on the Sites page. Site administrators can see *all* sites on the Sites page; other users can see sites they created or are members of. See Manage Sites and Site Settings.

3

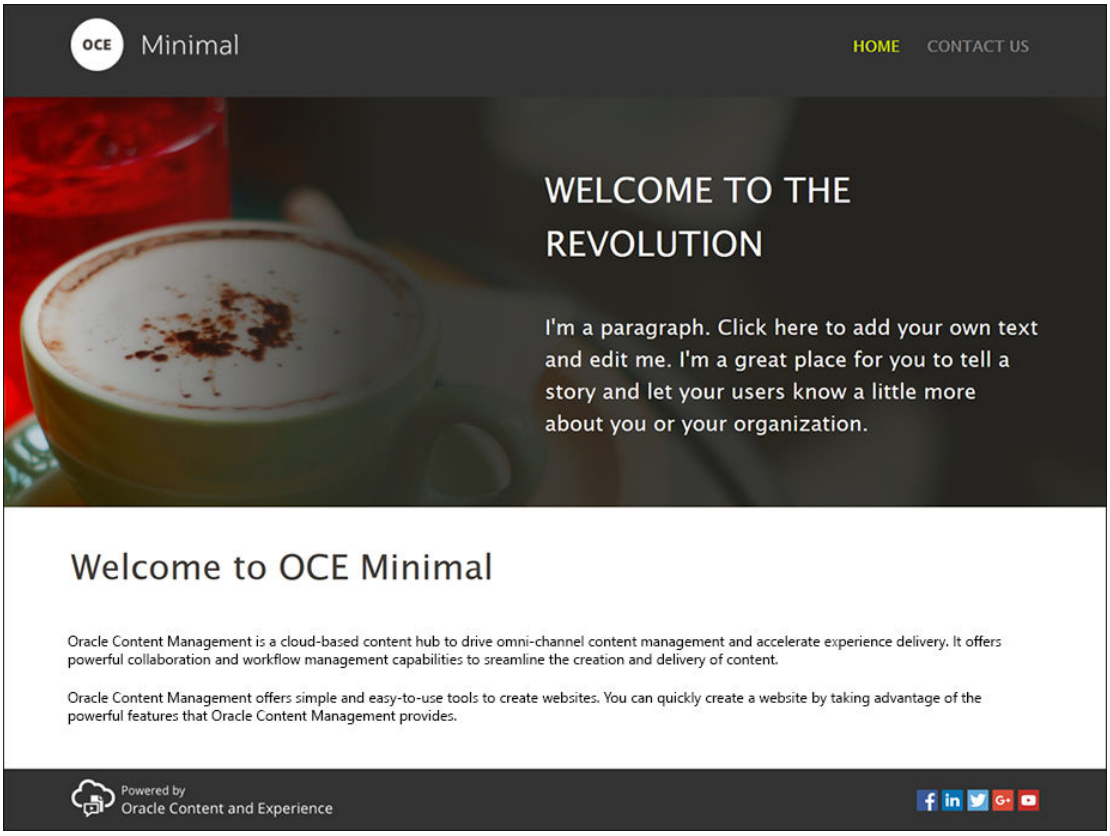
Create Your First Website

Build your first website easily and quickly using Oracle Content Management.

In this tutorial, you'll learn how to set up and publish a website using an out-of-the-box template available in Oracle Content Management called **Blank-Template**. You'll see how easy it is to build your first website using Site Builder, without the need for any additional coding.

This tutorial consists of three basic steps:

1. Setting up the environment
2. Setting up the website
3. Publishing the website



But before we get started, let's first check a few things.

Before You Begin

To get started, you'll need access to an Oracle Content Management instance with the following application roles:

- **CECContentAdministrator**
- **CECDeveloperUser**

If you don't have the above roles, ask a service administrator to assign them to you.

Once you have access to an Oracle Content Management instance, sign in as a content administrator.

Let's get started.

Step 1: Set Up the Environment

First, you need to set up the environment by importing the Minimal-NavMenu custom component and uploading the sample images.

Therefore, download the asset pack, **OCESamplesAssetPack.zip**, which is available at https://www.oracle.com/middleware/technologies/content-experience-downloads.html. Within the asset pack you downloaded, locate the **OCECreateYourFirstSite_data.zip** file, which contains the following two zip files:

- **Minimal-NavMenu.zip**, which contains the Minimal-NavMenu custom component, and
- **Minimal-Images.zip**, which contains the sample images used in the website.

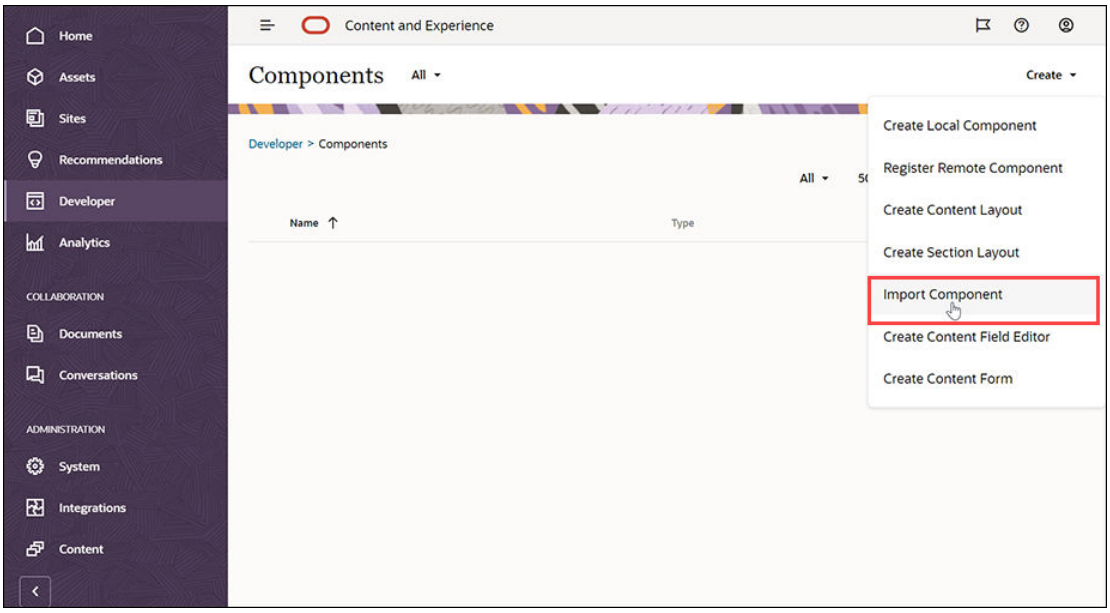
Let's get started with setting up the environment:

1. Import the Minimal-NavMenu Custom Component
2. Publish the Minimal-NavMenu Custom Component
3. Add Sample Images

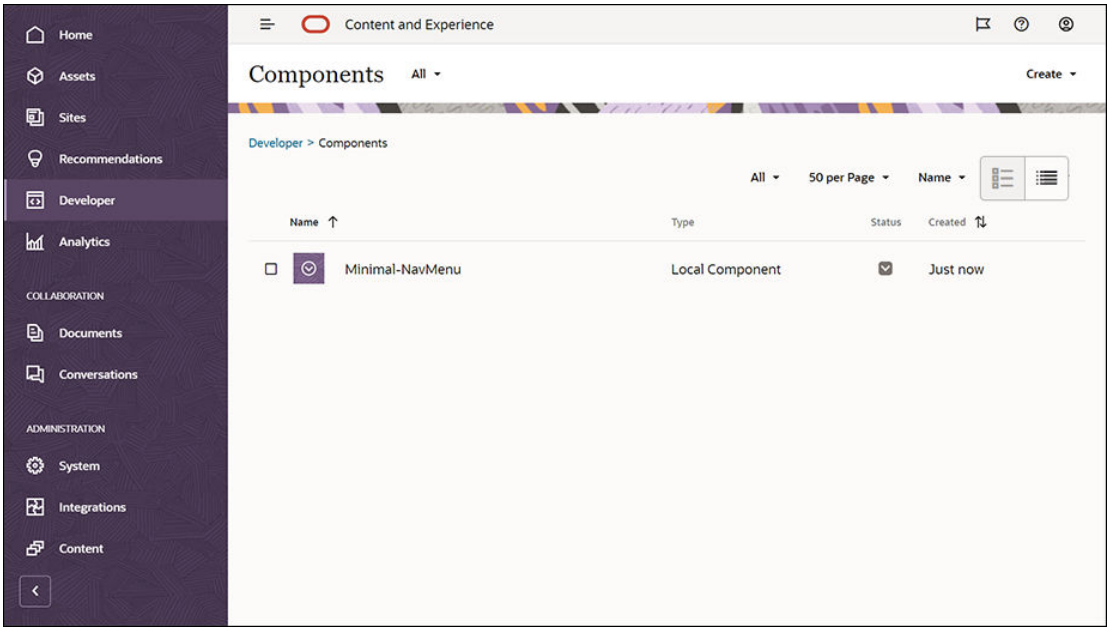
Import the Minimal-NavMenu Custom Component

After logging in to the Oracle Content Management web interface, click **Developer** in the left navigation menu, and then **View All Components**. If you don't see the **Developer** option, you don't have the required user roles.

On the Components page, click **Create**, and choose **Import Component**. Upload the **Minimal-NavMenu.zip** file (available within the **OCECreateYourFirstSite_data.zip** file) and select it.



The Minimal-NavMenu component is now listed on the Components page.

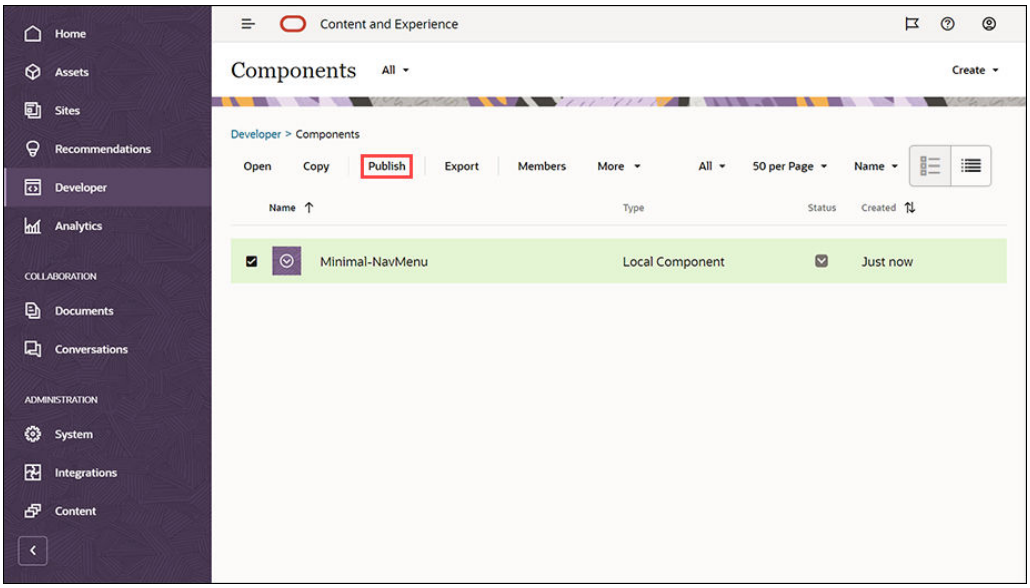


Next step: Publish the Minimal-NavMenu Custom Component

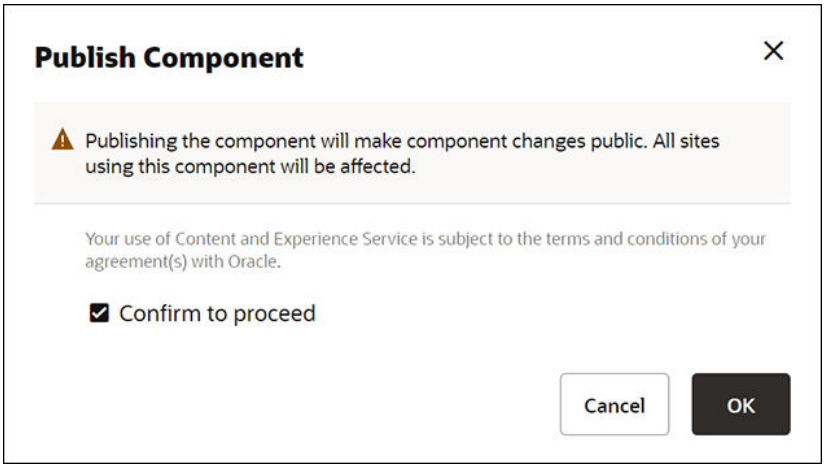
Publish the Minimal-NavMenu Custom Component

Now, you need to publish the Minimal-NavMenu component you imported.

On the Components page, select the Minimal-NavMenu component and click **Publish** from the menu bar or right-click menu.



In the Publish Component dialog, choose **Confirm to proceed** and click **OK**.



Once the Minimal-NavMenu component is published, a notification is displayed at the top of the page mentioning the component has been published.

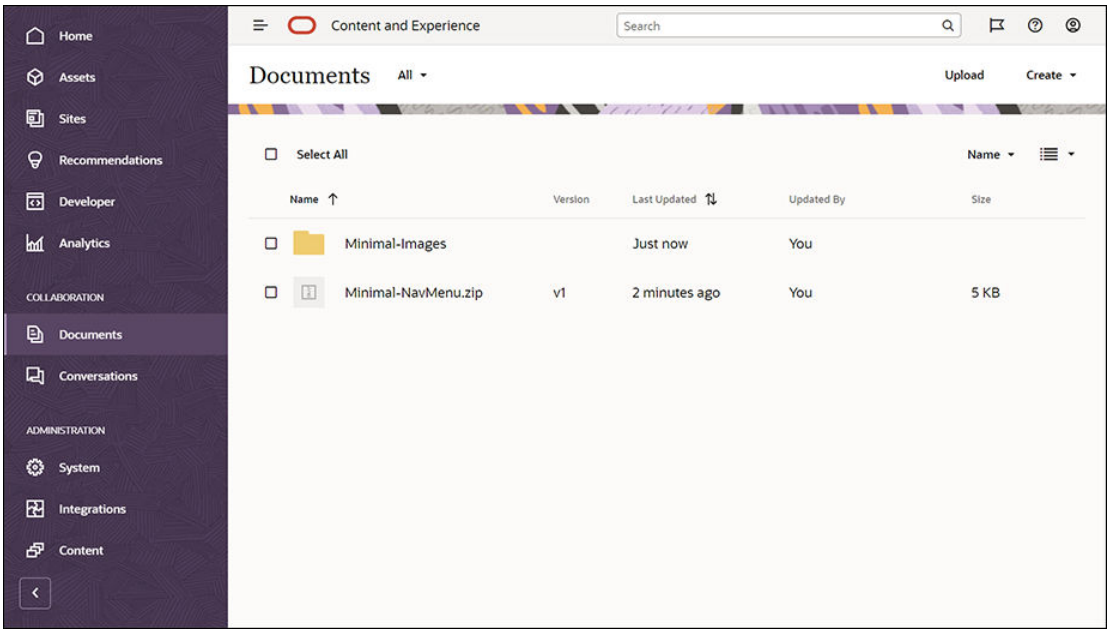
Next step: Add Sample Images

Add Sample Images

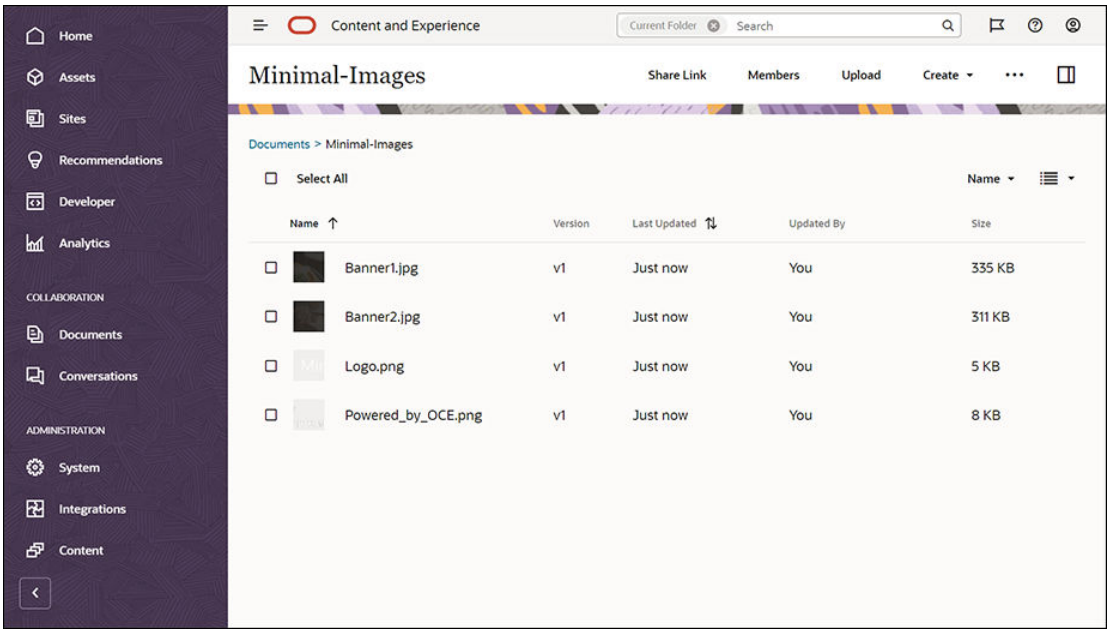
Now, you'll need to add the sample images (which you'll use in the website you create) to **Documents**.

Extract the contents of the **Minimal-Images.zip** file (available within the **OCECreateYourFirstSite_data.zip** file) to a folder named **Minimal-Images** on your local computer.

In the left navigation menu, click **Documents**, and then click **Create** to create a folder called **Minimal-Images**.



Upload all the images from your local computer's **Minimal-Images** folder into the newly created **Minimal-Images** folder in Oracle Content Management.



> **Note:**
>
> The images used in the template and the website are Shutterstock images. If you reuse these images, then they must be licensed through Shutterstock. You can also replace these images with your own images.

Next step: Set up the website

Step 2: Set Up the Website

Now that you have everything you need to create a website, you can begin setting up the website.

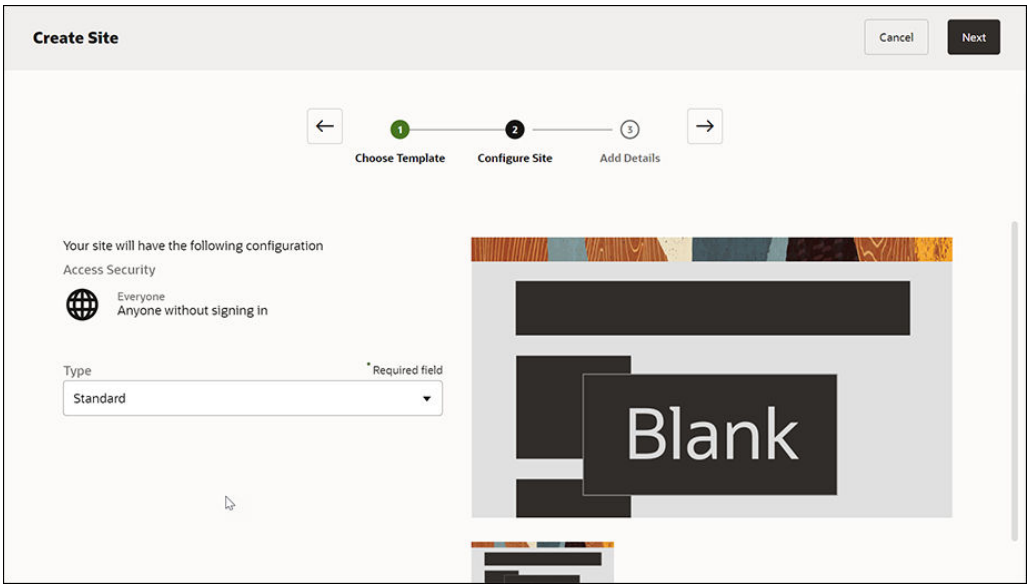
1. Create the Website
2. Edit the Website

Create the Website

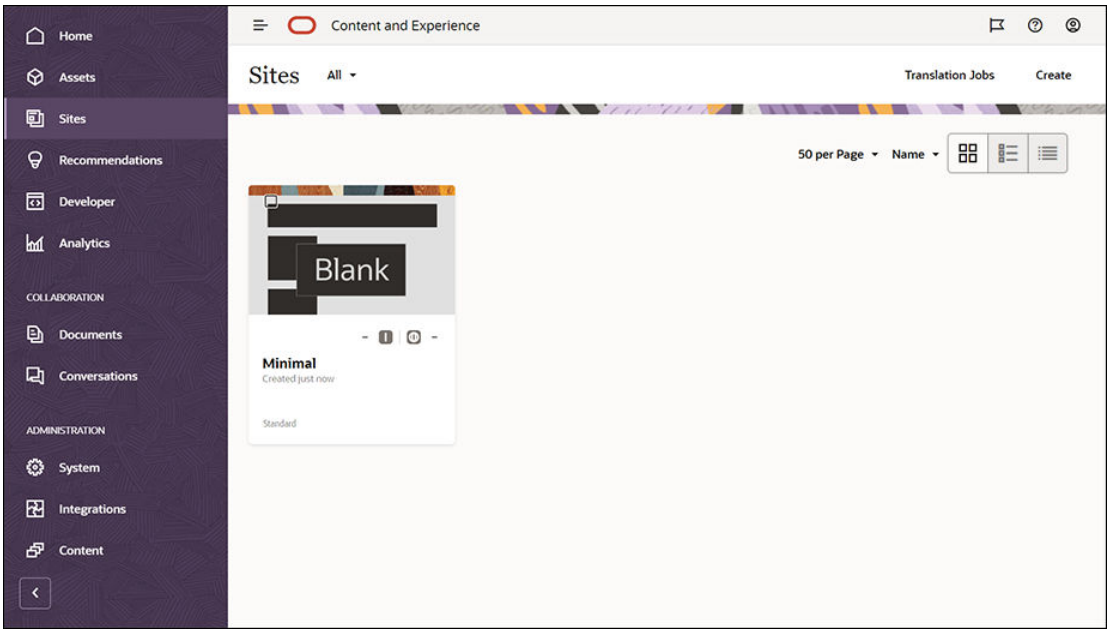
In the left navigation menu, click **Sites** and then click **Create**. Choose **Blank-Template** and then click **Next**.

If you don't see **Blank-Template**, contact your service administrator. The service administrator will need to make the out-of-the-box templates available for use. A service administrator typically installs the out-of-the-box templates that Oracle Content Management provides when the service is provisioned.

Make sure you choose to create a standard website and then click **Next**. Enter a name (for example, **Minimal**) for your website. Click **Finish**.



Your new website is created and listed on the Sites page.

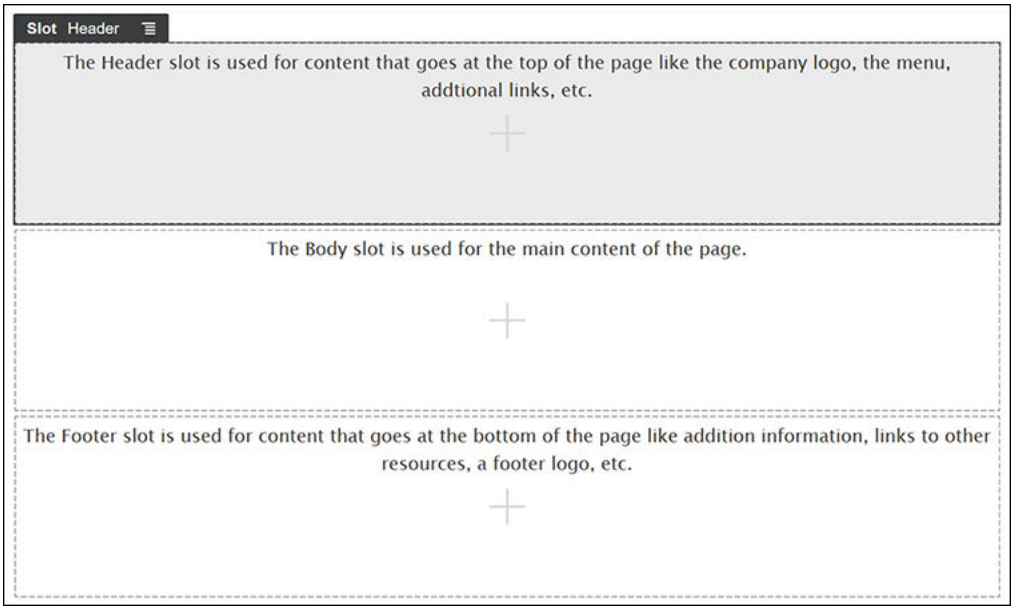


Next step: Edit the Website

Edit the Website

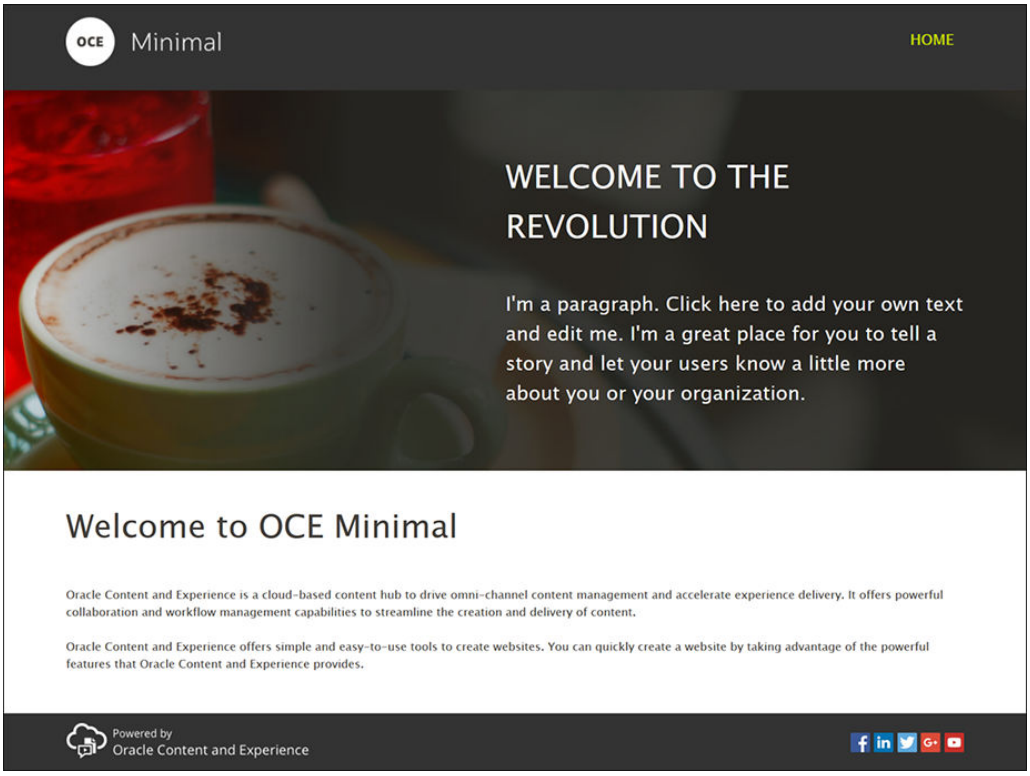
Open the newly created website in Site Builder by selecting it and choosing **Open** from the menu bar or right-click menu. In Site Builder, set the switch to **Edit** mode. Enter a name for the update and click **OK**.

In edit mode, you'll see that the website has three slots, which are areas available on the page (depending on the page layout). Hover over each + on the page to see slots such as Header, Body, and Footer.



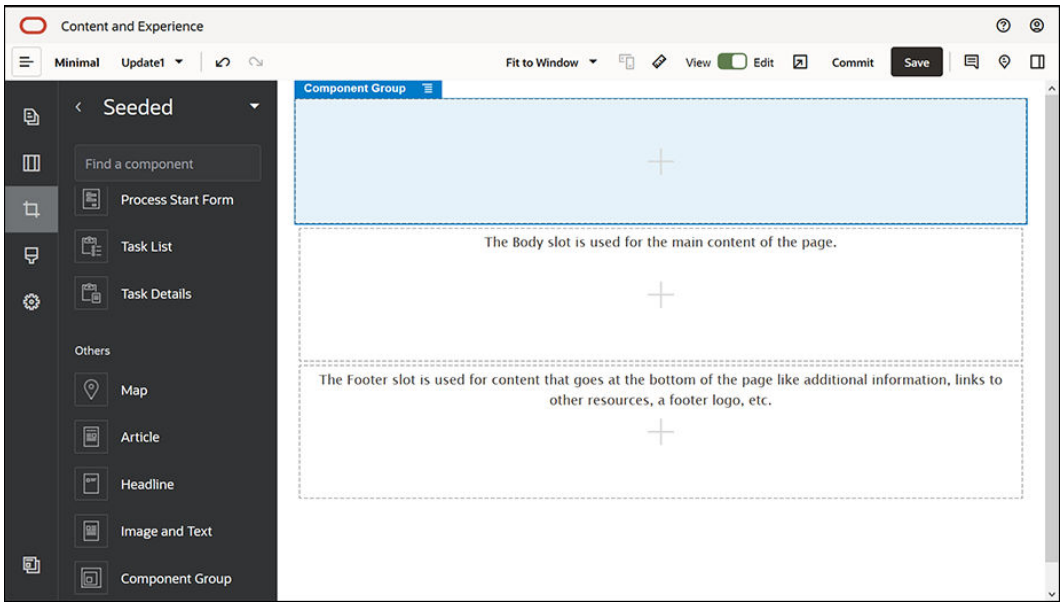
We generally use the Header slot to display the company's logo, the navigation menu, and so on. We use the Body slot for the main content of the page, and the Footer slot for copyright information, social media links, and any additional information.

First, let's build the home page. This is what the home page will look like once it's completed:

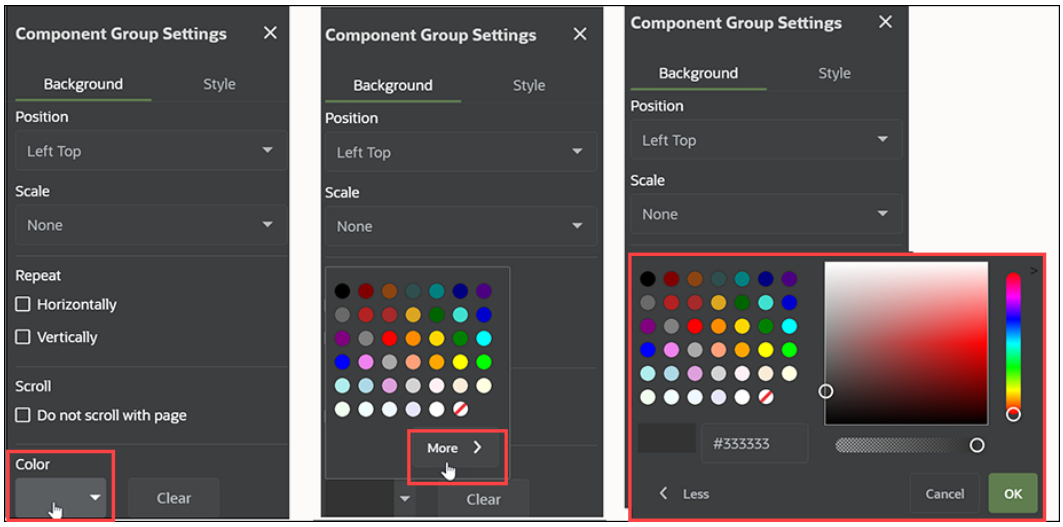


Let's use out-of-the-box components to fill in the Header slot:

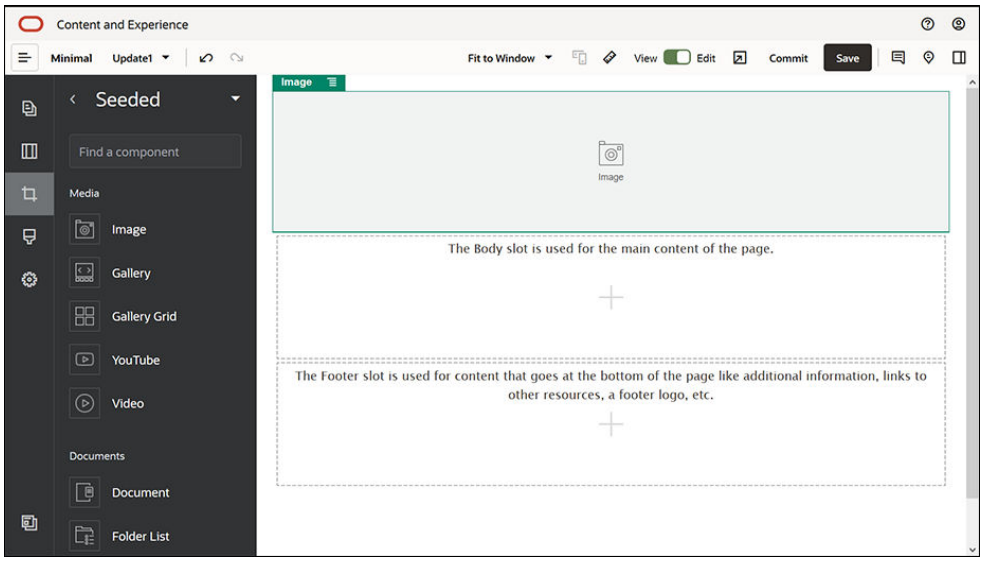
1. In the left sidebar, click and then click **Seeded** to show the list of out-of-the-box components available with Oracle Content Management.
2. In the left sidebar, look for an out-of-the-box component called Component Group. Drag and drop it into the Header slot.



3. Click the component group's menu icon and then click **Settings**. In the settings, click the **Color** drop-down list (available at the bottom of the settings list) and then click **More**. Enter **#333333** and click **OK**.

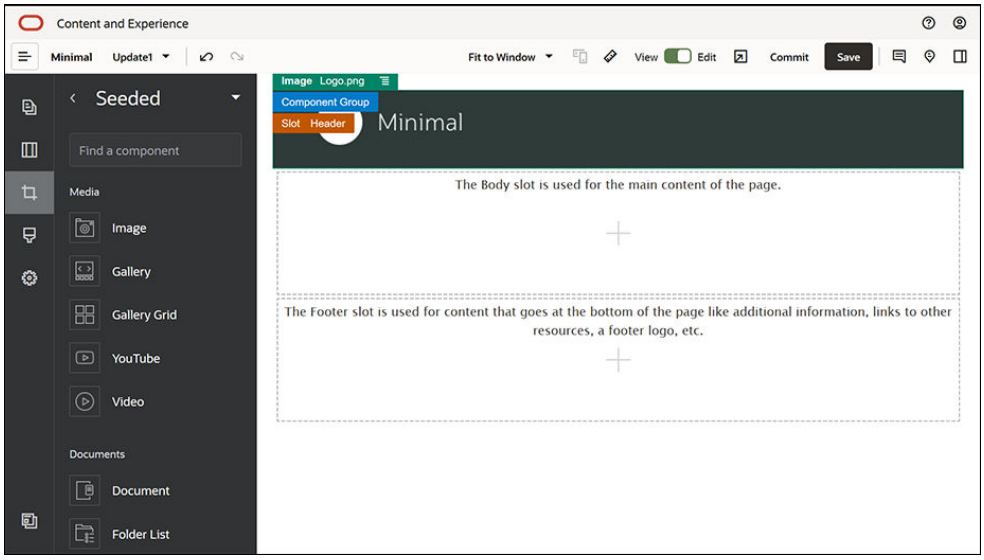


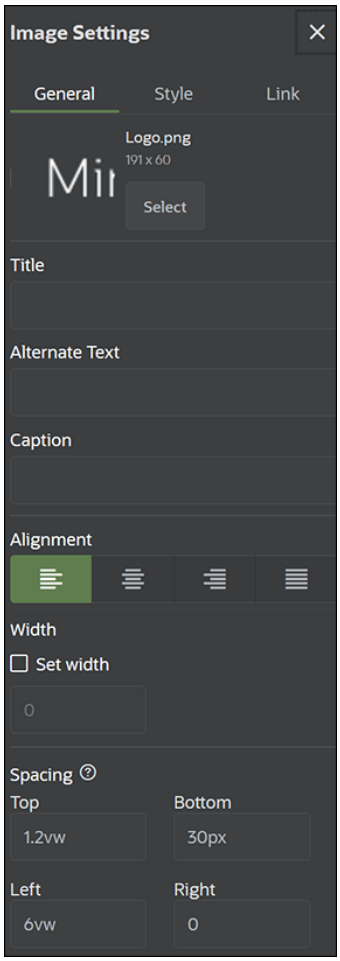
4. Drag and drop an Image component into the component group.



5. Click the Image component's menu icon and then click **Settings**. Complete the settings in the **General** tab.

Property	Value
Select	Logo.png from the Minimal-Images folder
Alignment	Left
Width	Deselect **Set Width**
Top	1.2vw
Bottom	30px
Left	6vw
Right	0

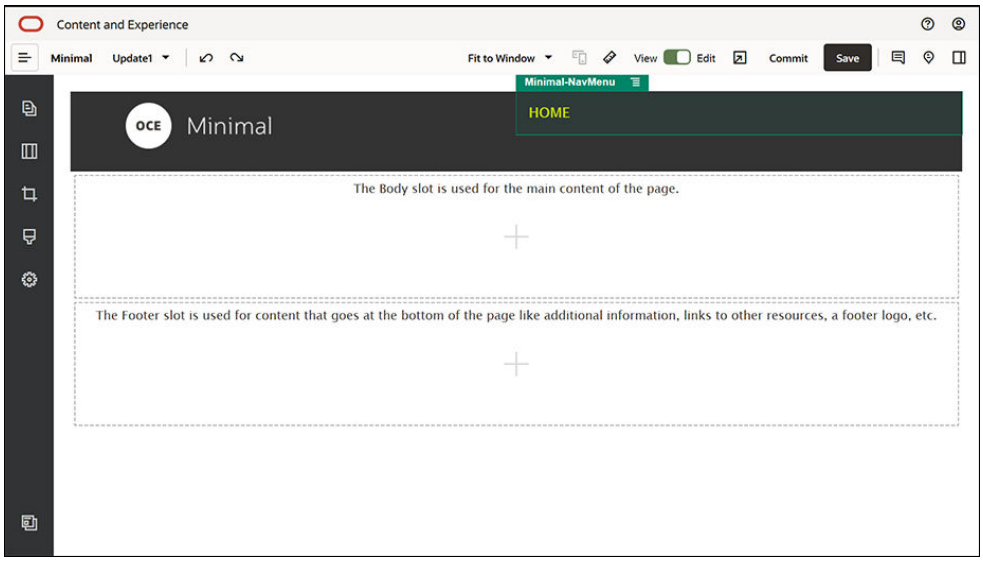




6. Let's link this logo image to the home page. Complete the Image component's settings in the **Link** tab.

Property	Value
Select Link Type	Site Page
Page	HOME
Target	Open in Same Window

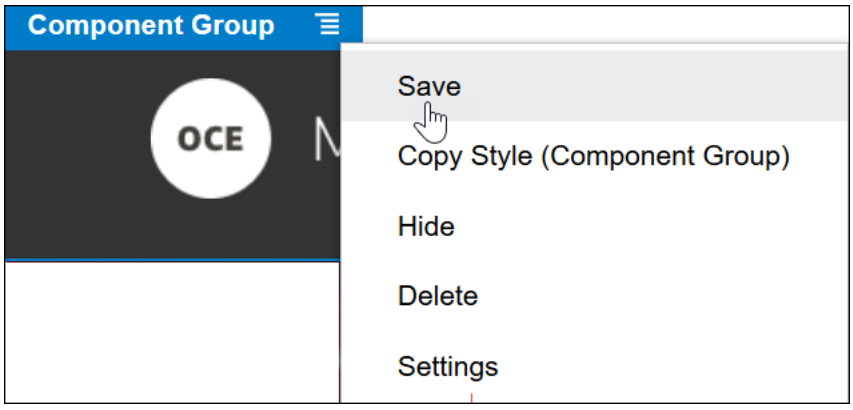
7. In the left sidebar, click  and click **Custom** to show the list of custom components.
8. Now, let's add a navigation menu to the home page using the Minimal-NavMenu custom component. Drag and drop a Minimal-NavMenu component into the component group, to the right side of the Image component. Click the title of the Minimal-NavMenu component to make sure its parent is the component group you added earlier. This is a useful way to see where any component resides within the structure of the web page.



Complete its settings in the **General** tab.

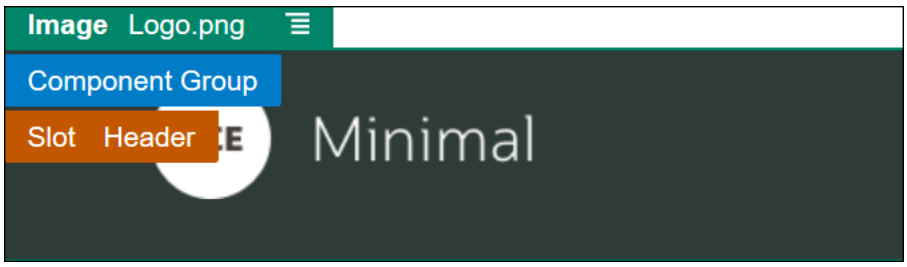
Property	Value
Alignment	Right
Top	1.2vw
Bottom	0
Left	0
Right	6vw

9. Now, the header is ready. Let's save this component group as a custom component group so that we can use it later in the other website pages. Click the title of the component group, then click its menu icon , and then click **Save**. In the Save Component Group dialog, in the **Name** field, enter "Minimal-Header" and then click **Save**.



> **Tip:**
>
> If you click the component group and you see the Image or Minimal-NavMenu component title rather than the component group title, click the Image or Minimal-NavMenu component title again and then you'll see the component group's title. Now, you can click the component group's title and perform the above step.

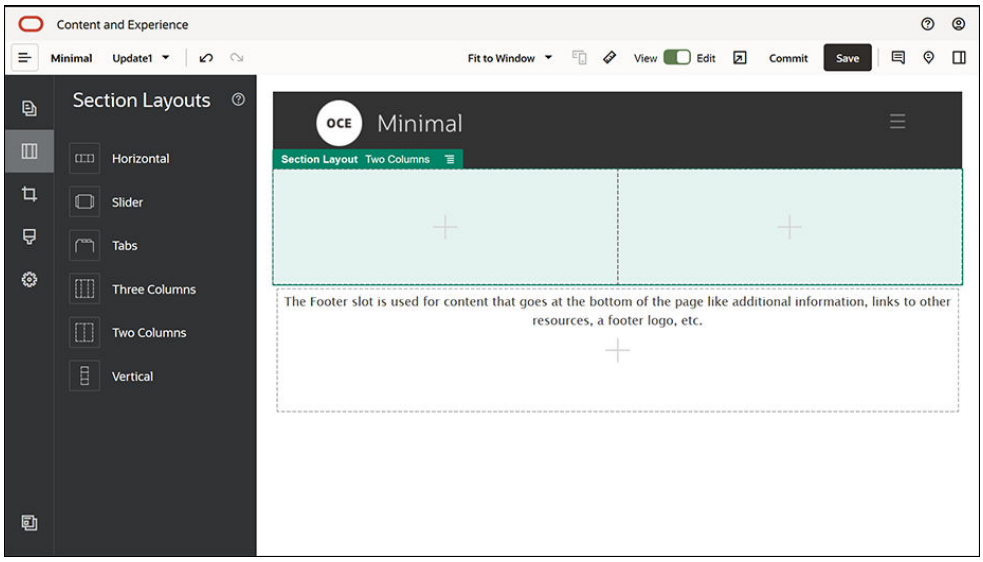
The following image shows the parent structure for the Image component in the Header slot:



10. Click **Save** in the upper right of Site Builder to save your changes. The Header slot should now look like the following image:

Let's move on to the Body slot:

1. In the left sidebar, click and then click **Seeded**.
2. From the left sidebar, drag and drop a Component Group into the Body slot. We'll create a banner using this component group and the components (which we'll be adding into it).
3. In the left sidebar, click .
4. Drag and drop a Two Columns section layout into the component group.



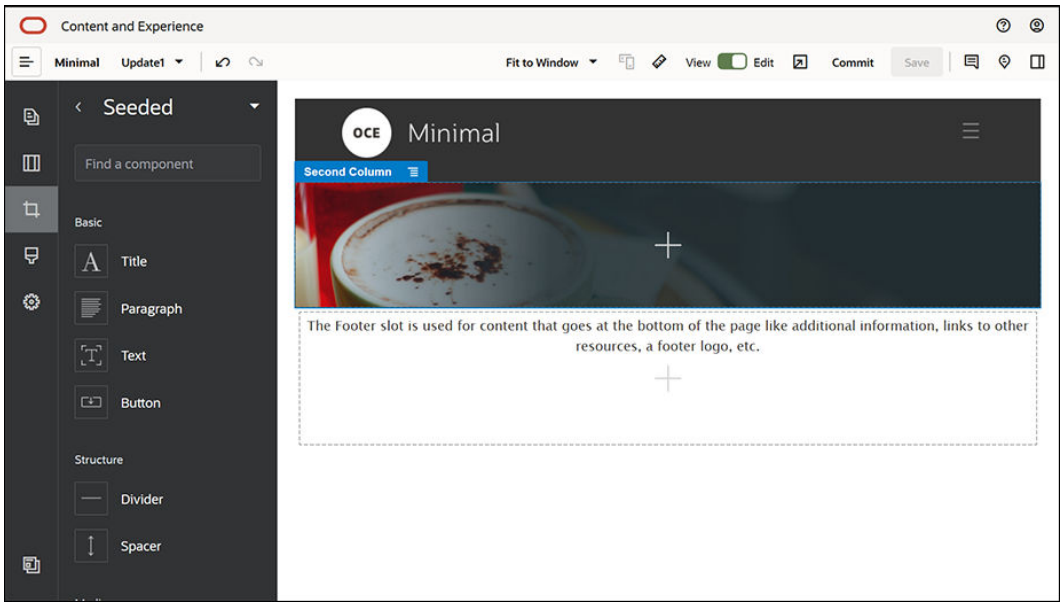
5. Complete the section layout's settings in the **General** tab. Click **Custom Settings** to specify the following settings.

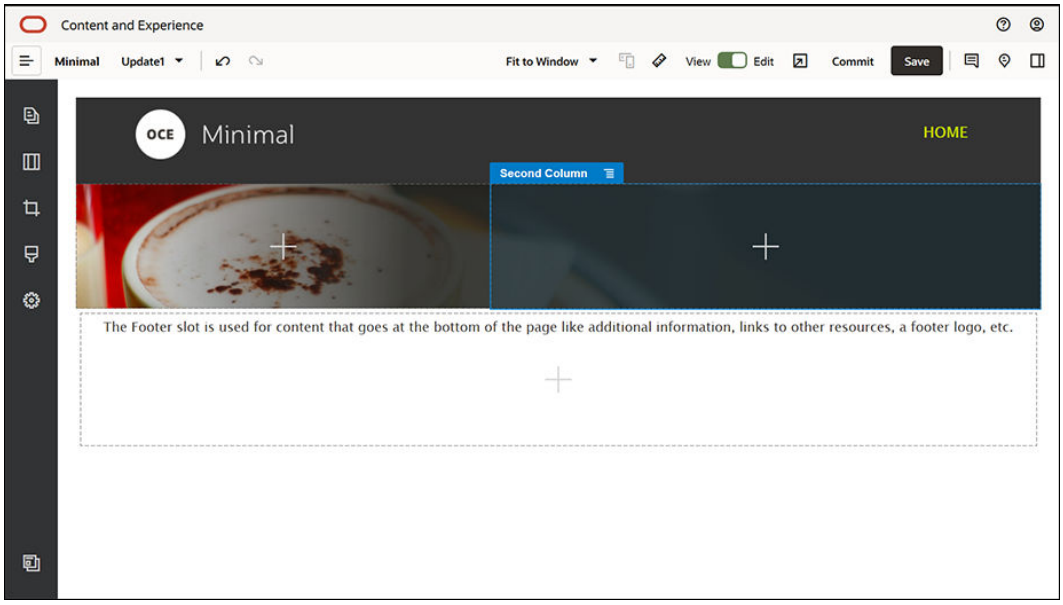
Property	Value
First Column Width (%)	43
Second Column Width (%)	57
Responsive Breakpoint (pixels)	1,023
Responsive Behavior	Hide the first column

Complete the settings in the **Background** tab:

Property	Value
Image	Banner1.jpg from the Minimal-Images folder
Position	Center Center
Scale	Stretch

6. In the left sidebar, click  to see the list of seeded components.
7. From the list of seeded components, drag and drop a Title component into the second column of the Two Columns layout.





8. Click in the Title component and enter "WELCOME TO THE REVOLUTION". Select the text and set its font color to **White** in the text editor. Complete the title component's settings in the **General** tab.

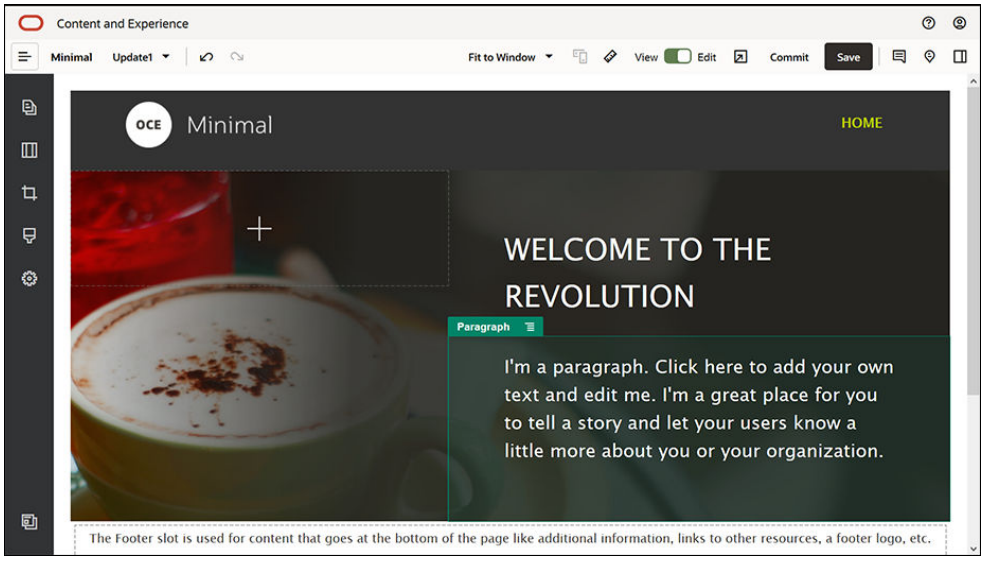
Property	Value
Top	6vw
Bottom	1.8vw
Left	6vw
Right	6vw

9. From the list of seeded components in the left sidebar, drag and drop a Paragraph component below the Title component, within the second column of the Two Columns layout. Complete its settings in the **General** tab.

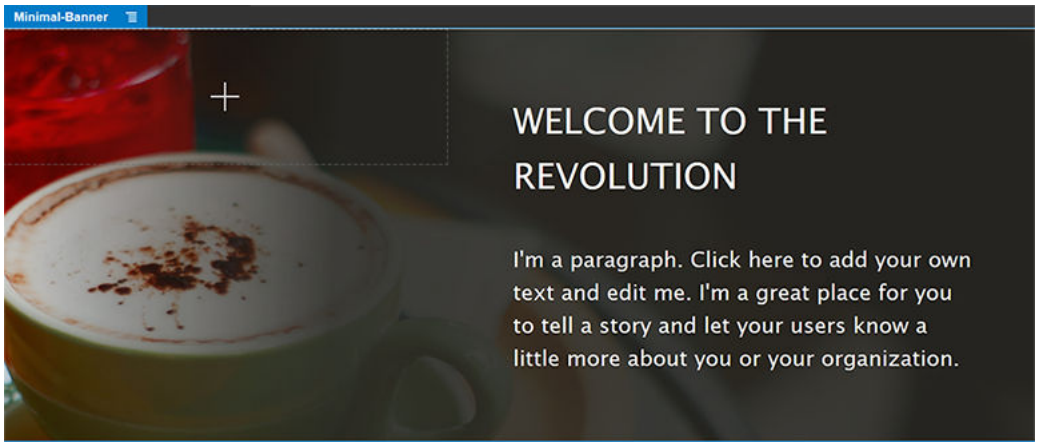
Property	Value
Top	1.8vw
Bottom	6vw
Left	6vw
Right	6vw

10. Click in the Paragraph component and enter the following text:
"I'm a paragraph. Click here to add your own text and edit me. I'm a great place for you to tell a story and let your users know a little more about you or your organization."

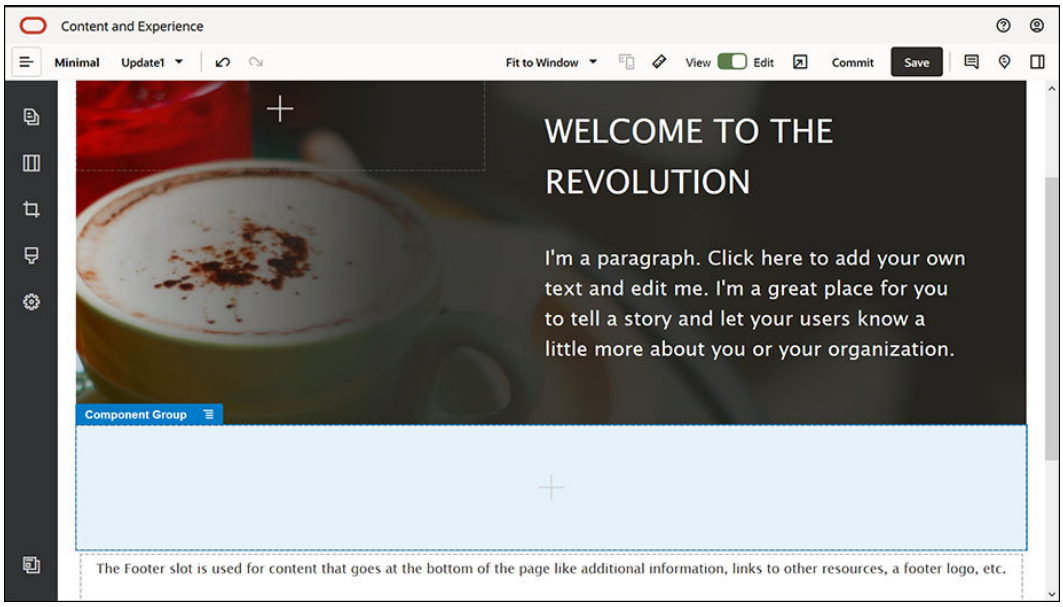
    Select the text and set its size to 24 in the text editor. Also, set its font color to **White** in the text editor.



11. Now, the banner is ready. Let's save this component group as a custom component group so that we can use it later in the other website pages. Click the component group's menu icon and then click **Save**. In the Save Component Group dialog, in the **Name** field, enter "Minimal-Banner" and then click **Save**. You'll notice that the name (**Minimal-Banner**) now shows up for the component group.

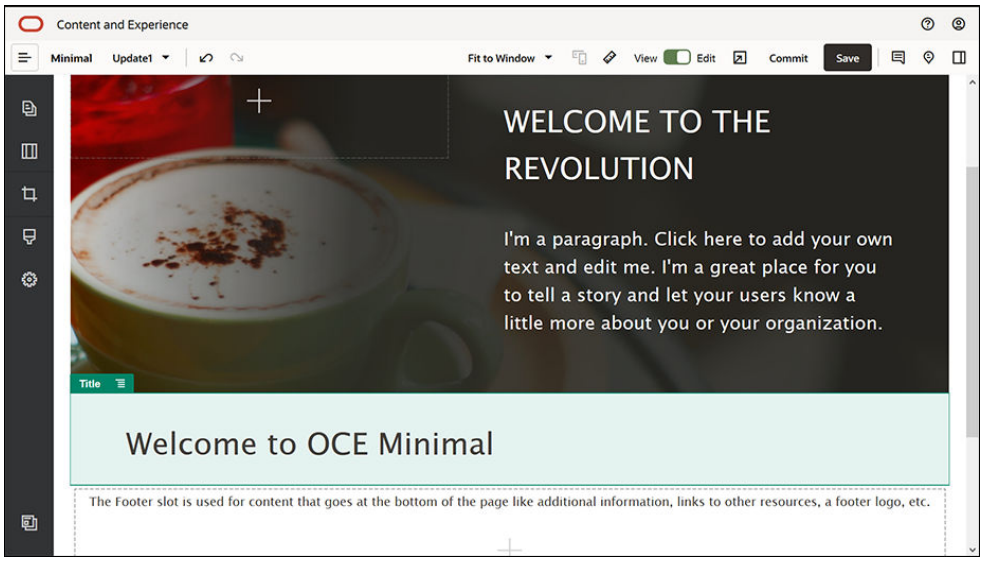


12. From the left sidebar, drag and drop another Component Group into the Body slot, below the Minimal-Banner component group you've already added.



13. Drag and drop a Title component into the component group.
14. Click in the Title component and enter "Welcome to OCE Minimal".
15. Complete the title component's settings in the **General** tab.

Property	Value
Top	3vw
Bottom	1.8vw
Left	6vw
Right	6vw

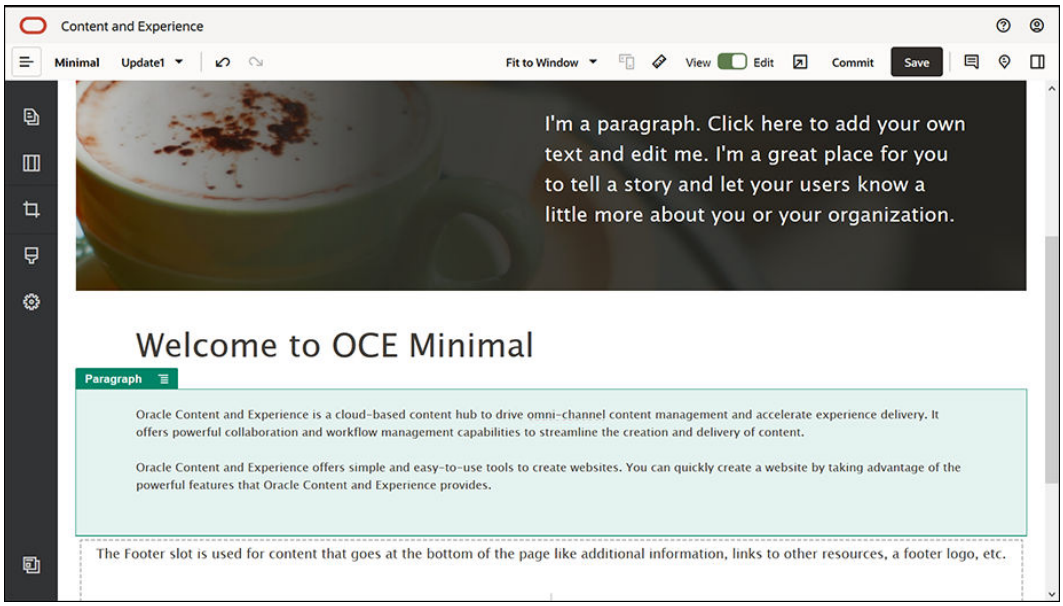


16. From the left sidebar, drag and drop a Paragraph component below the Title component, into the component group. Click in the Paragraph component and enter the following text:
"Oracle Content Management is a cloud-based content hub to drive omni-channel content management and accelerate experience delivery. It offers powerful collaboration and workflow management capabilities to streamline the creation and delivery of content."

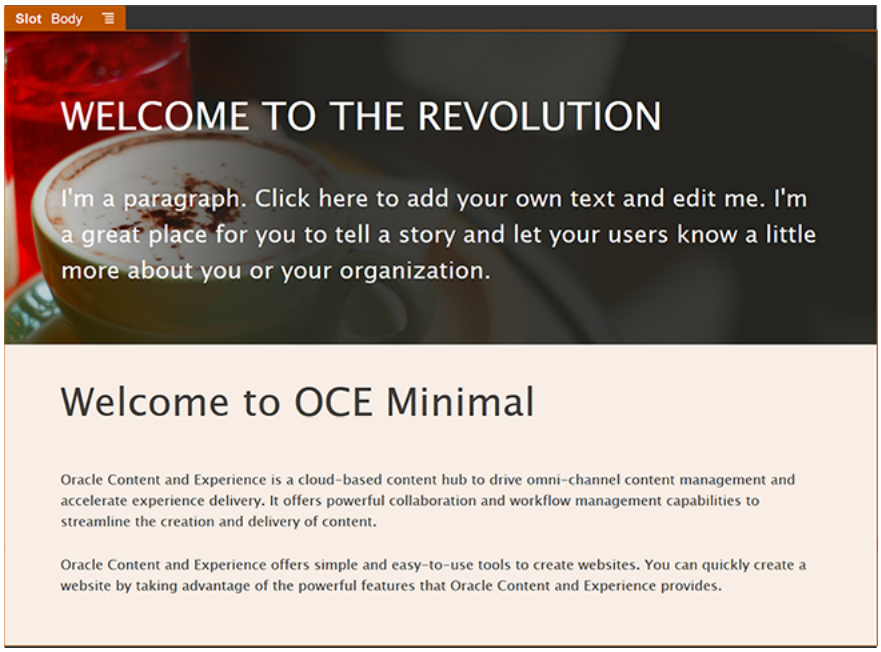
    "Oracle Content Management offers simple and easy-to-use tools to create websites. You can quickly create a website by taking advantage of the powerful features that Oracle Content Management provides."

17. Complete the Paragraph component's settings in the **General** tab.

Property	Value
Top	20px
Bottom	50px
Left	6vw
Right	6vw

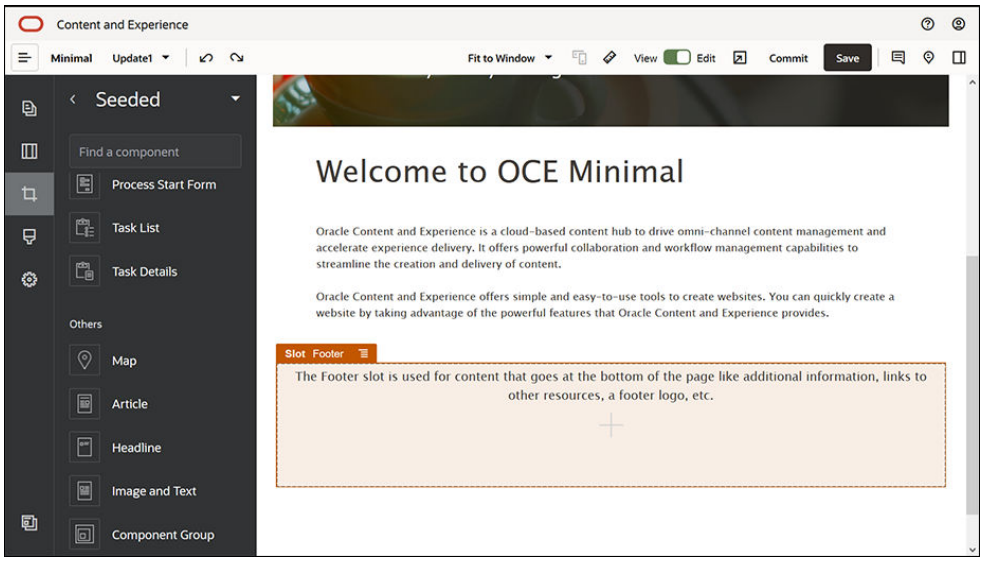


18. We've completed the body slot. Let's save the component group as a custom component group so that we can use it later in the other website pages. Click the component group's menu icon and then click **Save**. In the Save Component Group dialog, in the **Name** field, enter "Minimal-Body" and then click **Save**.
19. Click **Save** in the upper right of Site Builder to save your changes. The Body slot should now look like the following image:



Let's complete the Footer slot:

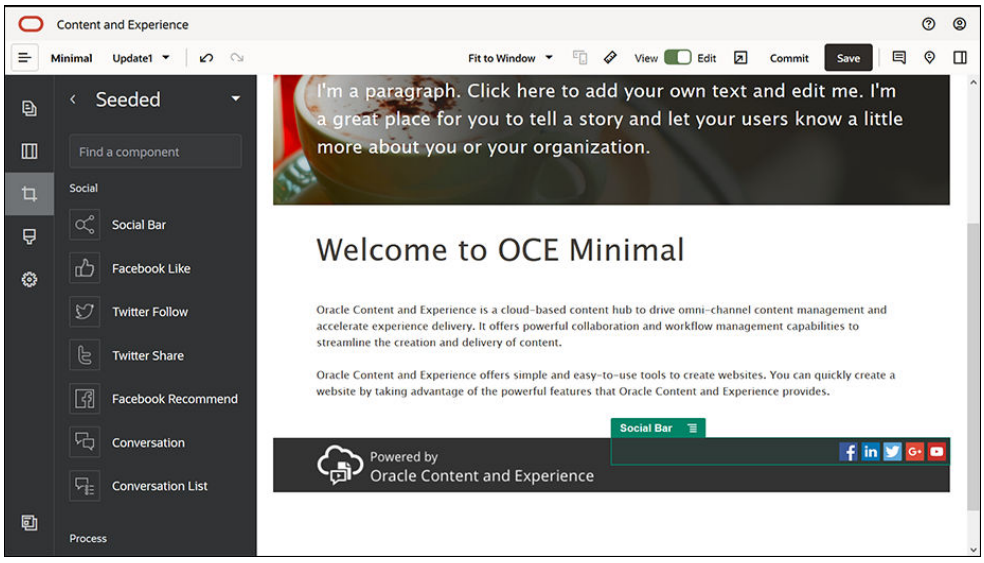
1. From the left sidebar, drag and drop a Component Group into the Footer slot. In the component group's settings, set the **Color** field to **#333333**.



2. Drag and drop an Image component into the component group and complete its settings in the **General** tab.

Property	Value
Select	Powered_by_OCE.png from the Minimal-Images folder
Alignment	Left
Width	Deselect **Set Width**
Top	0.9vw
Bottom	0.9vw
Left	6vw
Right	0

3. From the left sidebar, drag and drop a Social Bar component into the component group, to the right side of the Image component.



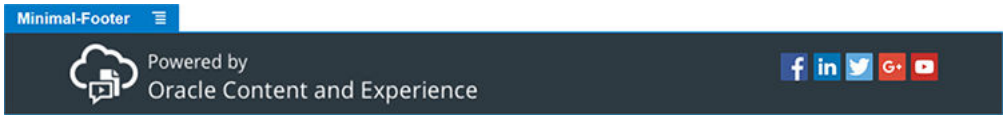
Complete the settings for the Social Bar component in the **General** tab.

Property	Value
Top	1.8vw
Bottom	1.8vw
Left	0.3vw
Right	6vw

In the **General** tab, click **Icons** and then click an icon name to complete the settings.

Property	Value
URL	• https://www.facebook.com/Oracle/ (for Facebook) • https://www.linkedin.com/company/oracle/ (for LinkedIn) • https://twitter.com/Oracle (for Twitter) • https://www.youtube.com/oracle/ (for YouTube)
Target	Open in New Window

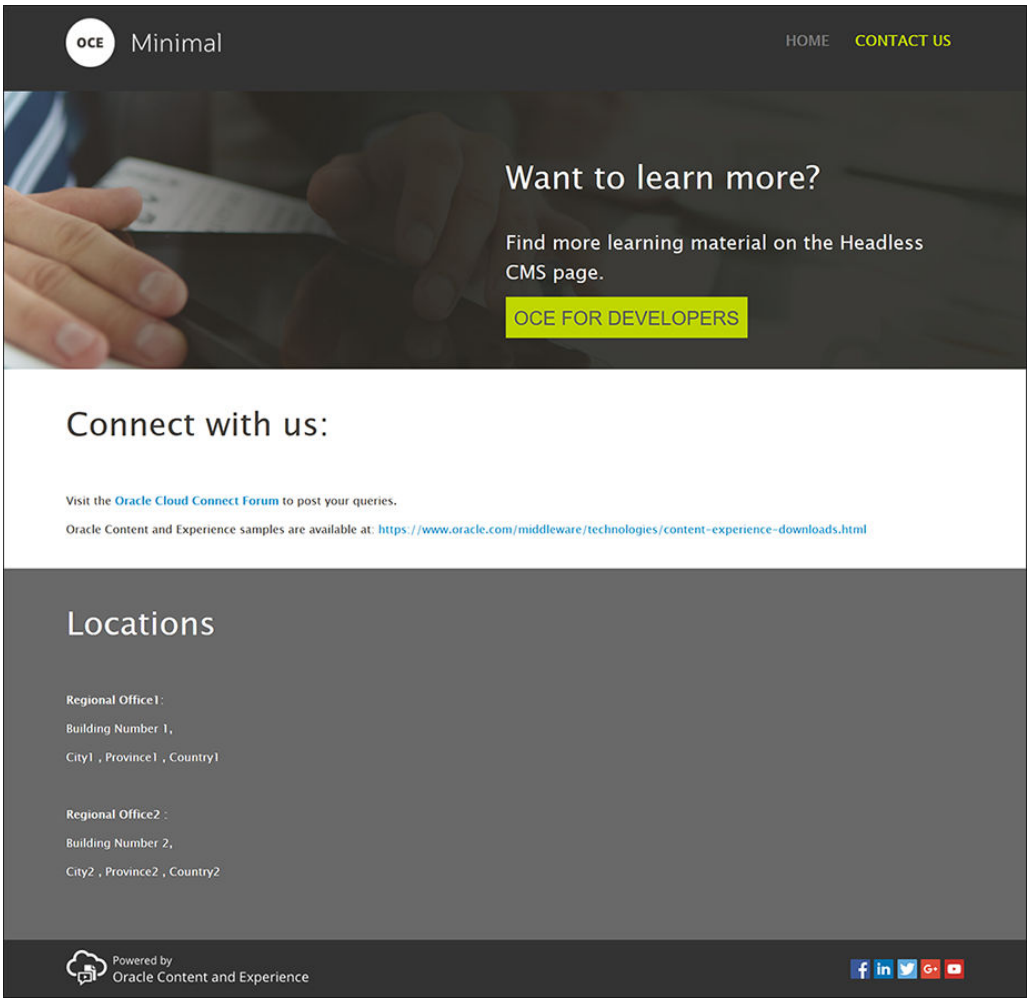
4. Now, the footer is ready. Let's save this component group as a custom component group so that we can use it later in the other website pages. Click the component group's menu icon and then click **Save**. In the Save Component Group dialog, in the **Name** field, enter "Minimal-Footer" and then click **Save**. The component group should look like:



5. Click **Save** in the upper right of Site Builder to save your changes.

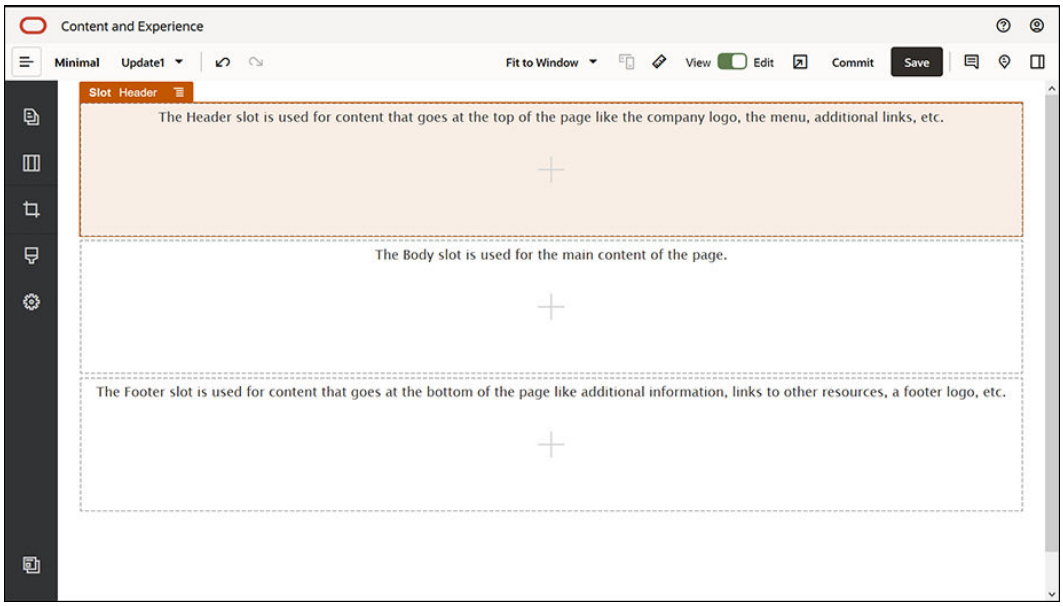
Preview the first page of your website by clicking in the upper right of Site Builder. The website is still not published and cannot be viewed by others now.

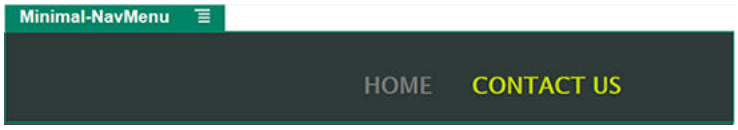
You've completed creating the HOME page. Let's build the CONTACT US page. This is what the contact page will look like once it's completed:



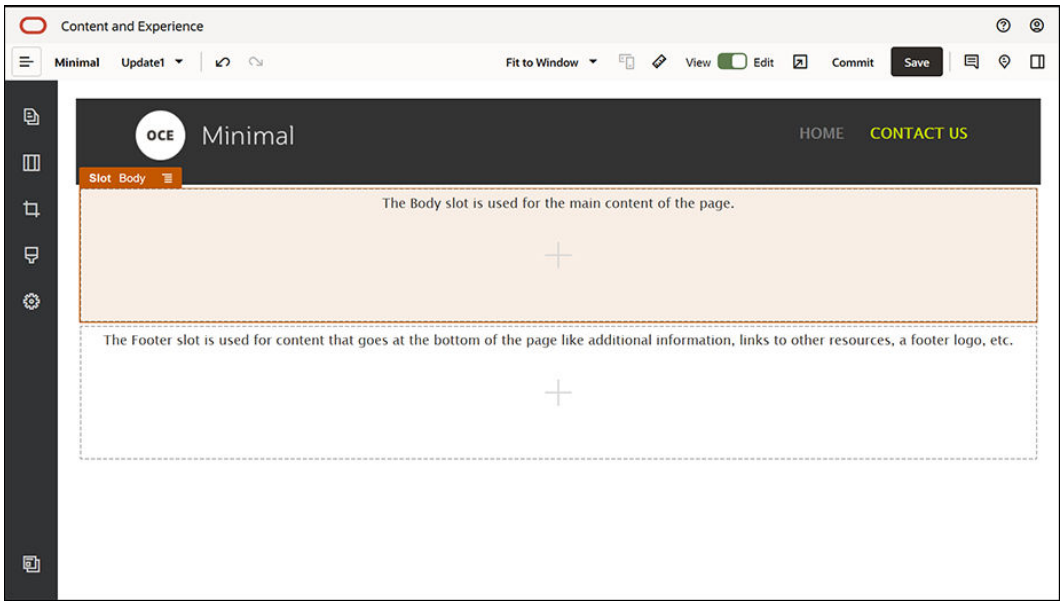
Let's add components into the various slots:

1. In the left sidebar, click and then click **Add Page**.
2. Enter "CONTACT US" in the **Page Name** field and click **Close**. You've added a new page to your website.
3. In the left sidebar, click and then click **Custom**.
4. Drag and drop a Minimal-Header component (which you created and saved earlier) into the Header slot.

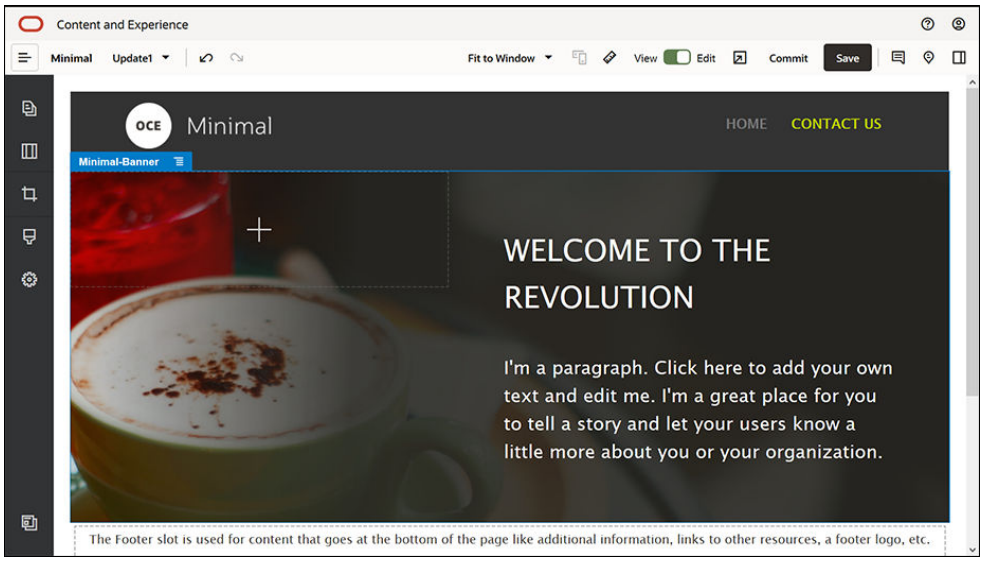




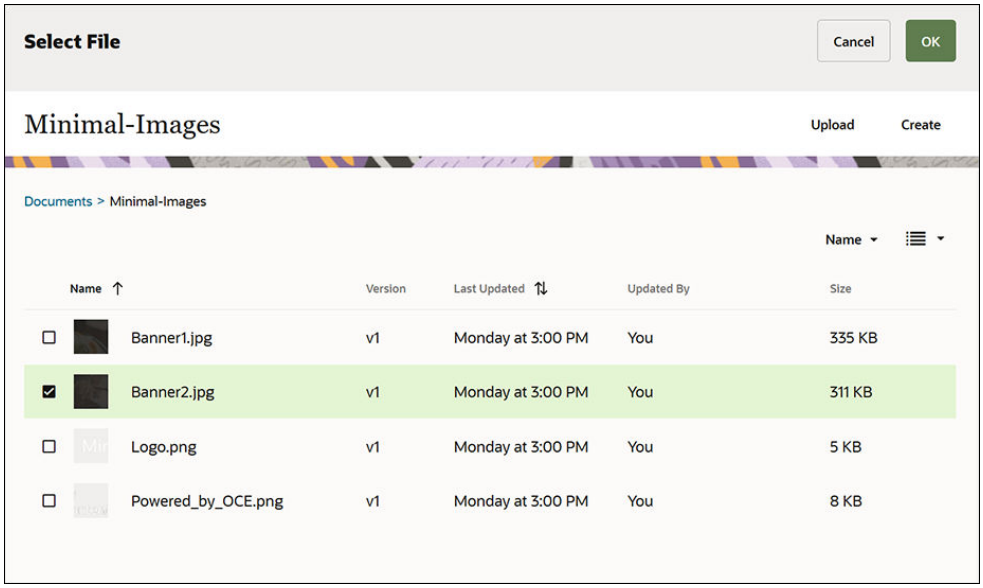
Notice that the Minimal-NavMenu component has automatically picked up the new CONTACT US page you just created.



5. From the left sidebar, drag and drop a Minimal-Banner component into the Body slot.



6. Let's modify the banner so that it looks different from the HOME page's banner and suits the CONTACT US page. Within the component group, modify the settings for the Two Columns section layout:
In the **Background** tab, in the **Image** field, click **Select Image** and then select **Banner2.jpg** from the **Minimal-Images** folder you created earlier in **Documents**.



7. Within the Two Columns section layout, modify the text in the Title and Paragraph components.
   - Enter "Want to learn more?" in the Title component.
   - Enter "Find more learning material on the Headless CMS page." in the Paragraph component.
8. In the left sidebar, click  and then click **Seeded**.

9. The image for CONTACT US page has a button called **OCE FOR DEVELOPERS** as part of the banner, so let's add this button to the banner. Within the Minimal-Banner component group, drag and drop a Button component into the second column of the Two Columns section layout (below the recently added Paragraph component). Complete the Button component's settings in the **General** tab.

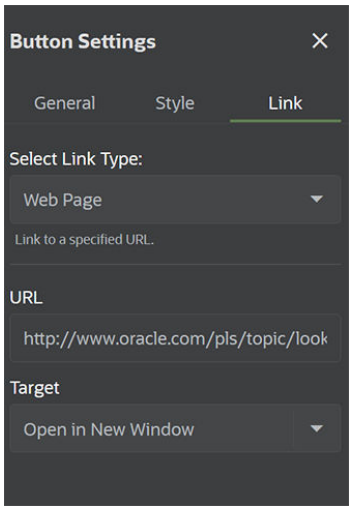
Property	Value
Label	OCE FOR DEVELOPERS
Top	0.3vw
Bottom	3vw
Left	6vw
Right	0.3vw

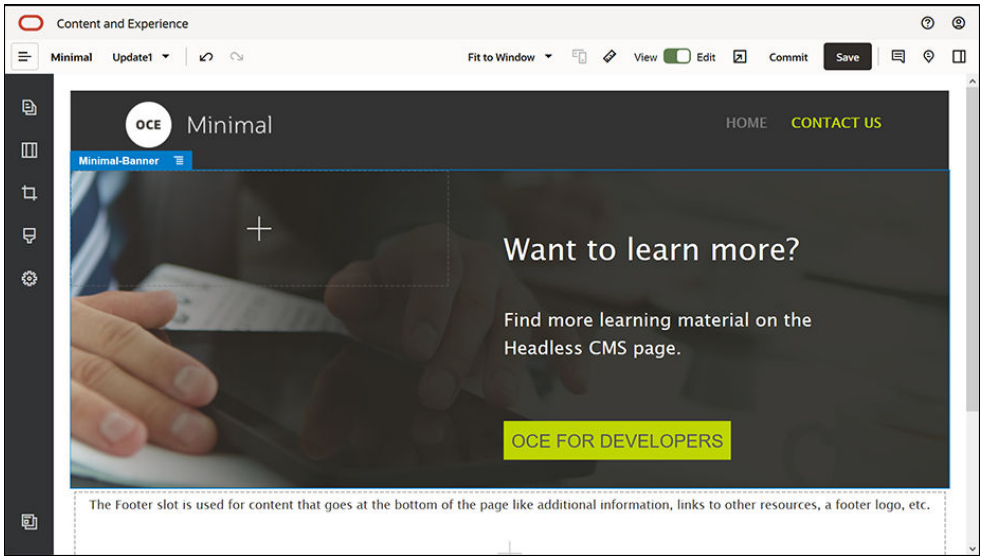
In the **Style** tab, select **Customize** and complete the settings.

Property	Value
Background Color	#c0d600
Font	• Enter 24 as the size. • Enter color #58595b.
Border	None
Hover Color	• Set **BACKGROUND** to #e1fa00. • Set **FONT** to #58595b. • Set **BORDER** to #2222dd.
Corners	0

Complete the settings in the **Link** tab.

Property	Value
Select Link Type	Web Page
URL	http://www.oracle.com/pls/topic/lookup?ctx=cloud&id=content-cloud-headless
Target	Open in New Window





10. In the left sidebar, click and then click **Custom**.
11. Drag and drop a Minimal-Body component below the Minimal-Banner component, into the Body slot.
12. Within the Minimal-Body component, modify the text in the Title and Paragraph components.
    a. Enter "Connect with us:" in the Title component.
    b. Enter details such as an email address and other support-related links in the Paragraph component:
    "Visit the Oracle Cloud Connect Forum to post your queries."

    "Oracle Content Management samples are available at: https://www.oracle.com/middleware/technologies/content-experience-downloads.html"
13. From the left sidebar, drag and drop another Minimal-Body component into the Body slot, below the Minimal-Body component you added earlier.
14. In the component group's settings, in the **Background** tab, set the **Color** field to **#696969**.
15. Within the Minimal-Body component, modify the Title and Paragraph components.
    a. Enter "Locations" in the Title component. Select the text and set its font color to **White** in the text editor.
    b. Enter the following text in the Paragraph component.
    "Regional Office1:

    Building Number 1,

    City1, Province1, Country1

    Regional Office2:

    Building Number 2,

    City2, Province2, Country2"

Select the text and set its font color to **White** in the text editor.

16. Now, let's add the footer to the CONTACT US page. Drag and drop a Minimal-Footer component into the footer slot.
17. Click **Save** in the upper right of Site Builder to save your changes.

Preview the website by clicking in the upper right of Site Builder to make sure everything looks good. Make sure the logo image on the CONTACT US page takes you back to the home page when clicked. Test the menu to make sure the navigation between the website pages works correctly.

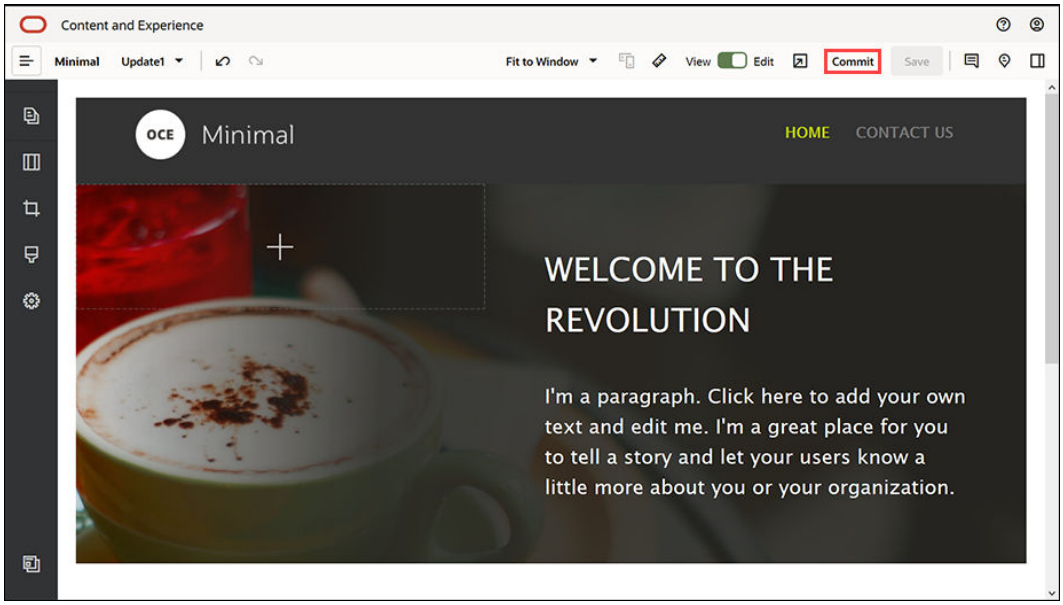
Your website is ready to be published.

Next step: Publish the website

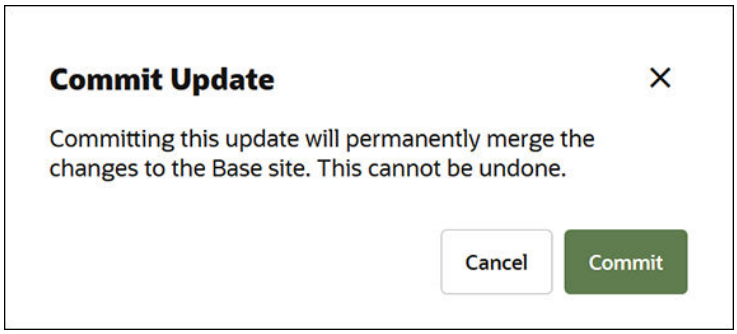
Step 3: Publish the Website

Now that you have successfully created your website, you can publish it to make it available online for your users to see.

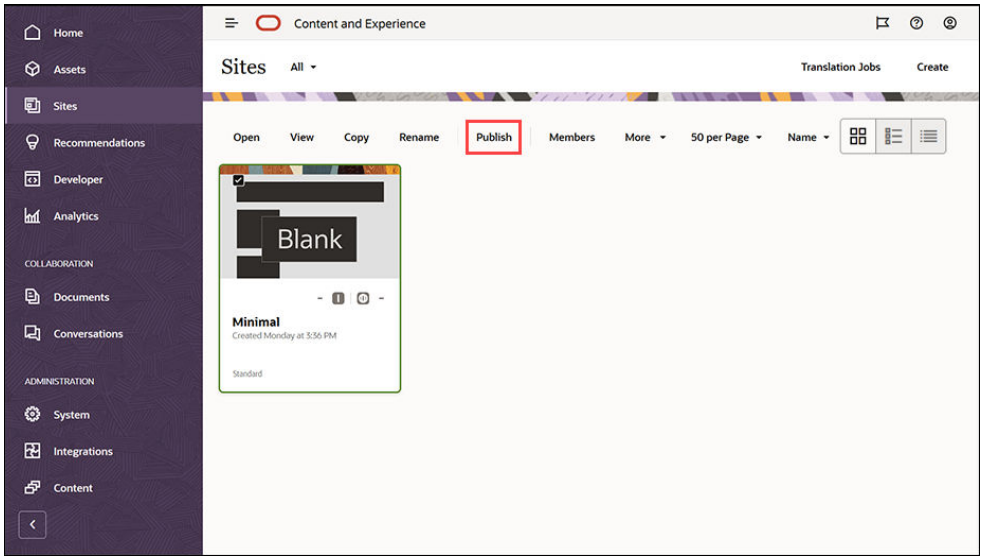
1. Once everything looks good, you can commit your changes to the base website by clicking **Commit** in the upper right of Site Builder.



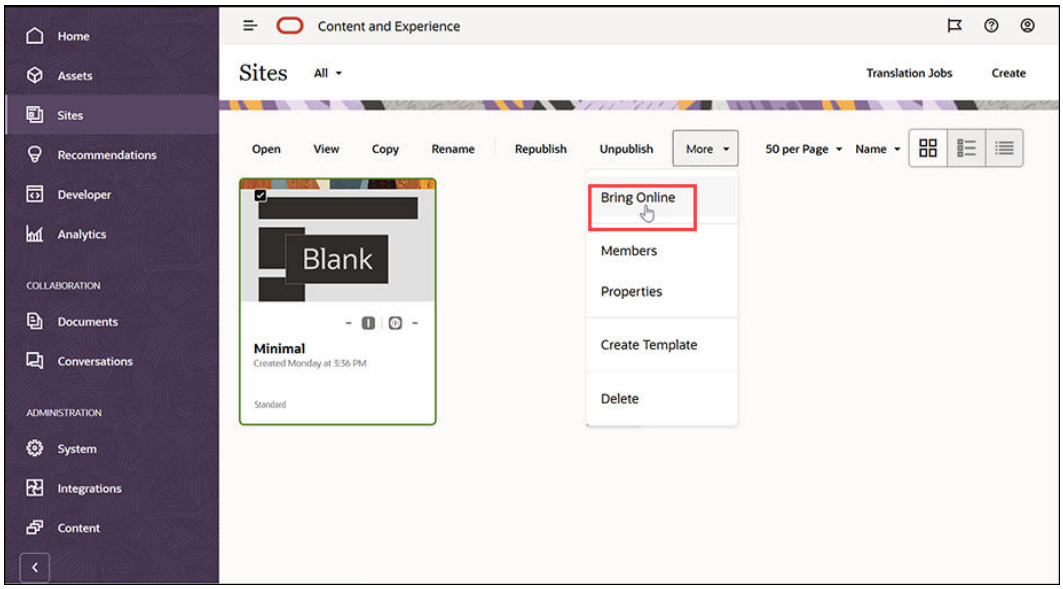
2. In the Commit Update dialog, click **Commit**.

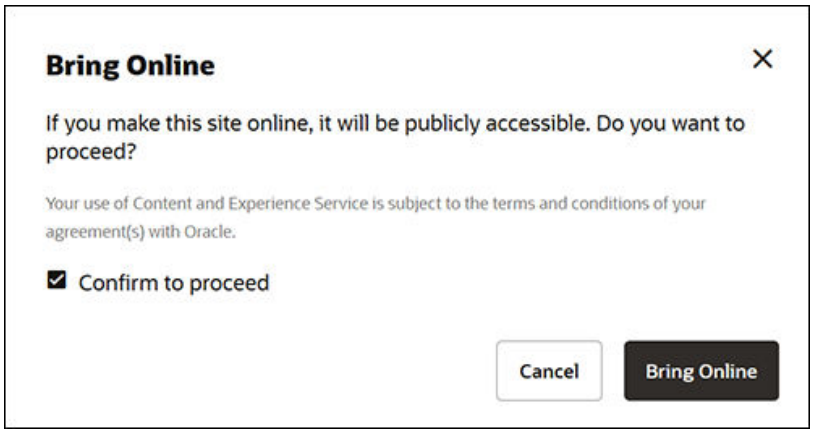


3. Once the changes are committed, the website is ready to be published.
4. Close Site Builder.
5. Click **Sites** in the left navigation menu in the Oracle Content Management web interface and select the website.
6. Choose **Publish** from the menu bar or right-click menu.



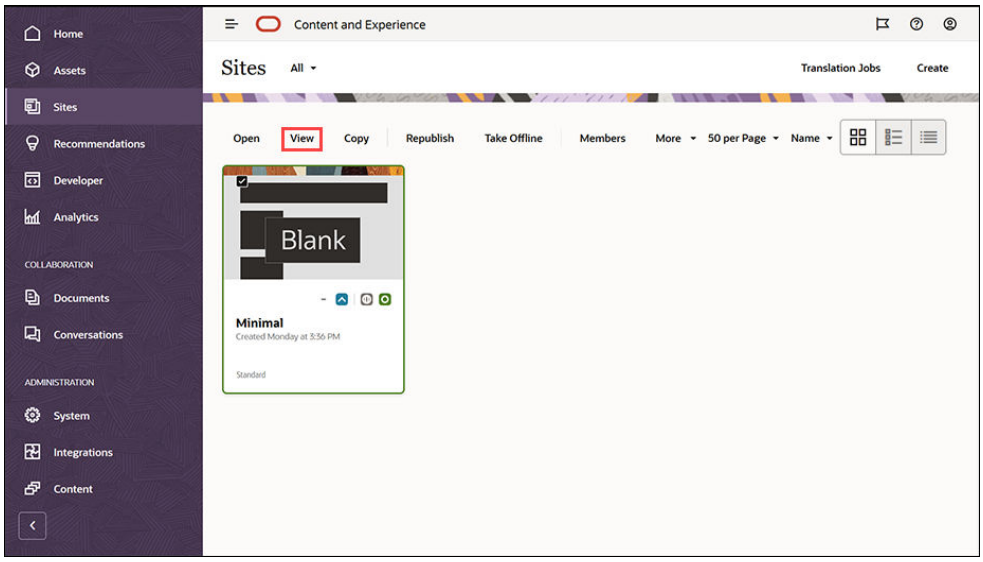
7. Once the website is published, a notification is displayed at the top of the page mentioning the website has been published.
8. On the Sites page, select the website again, and then choose **Bring Online** from the menu bar or right-click menu. In the Bring Online dialog, choose **Confirm to proceed** and click **Bring Online**.





9. That's it. Your website is online and others can view it.

   To view your public website, select **View** from the menu bar or right-click menu.



Do More

You can customize your website to suit your organization's needs. Here are a few useful links to help you get started:

- Use Styles and Formatting
- Editing Tips and Tricks
- Manage Sites and Site Settings
- Upload Site Files
- Add Pages
- Move Pages
- Arrange Page Content
- Change Page Settings
- Change the Page Layout
- Work with Tables
- Set Search Engine Properties

Part II
Creating and Editing Sites

This part details how to begin creating sites using templates and themes, and edit sites using components and layouts to orgainze and add content. It includes the following chapters:

- Create Sites
- Edit Sites
- Use Site Templates and Themes in Sites
- Manage Custom Components and Layouts
- Work with Site Pages
- Arrange Page Content
- Use Built-In Components

4

Create Sites

Anyone with the proper permissions is capable of building a website with Oracle Content Management. You don’t need to use any proprietary tools or code or software. The user interface is graphical, intuitive, and friendly.

When you create a site, you begin with a template. A template has everything you need to get started with your site, including the site code framework, a default site with sample pages and content, a theme with styling, resources such as images, and even custom components. See Understand the Site Creation Process.

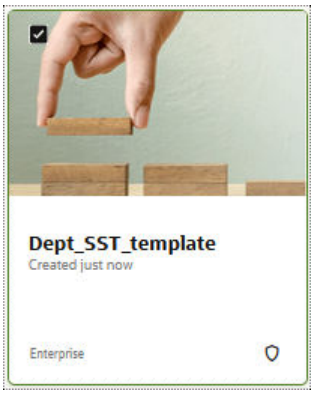
Oracle Content Management's site governance simplifies and accelerates experience delivery for business users while giving your IT departments an easy way to control and track experiences from a centralized location with the ability to fully manage the entire experience lifecycle, reducing the cost to create and maintain each new experience a company needs. Governance is built into the core of Oracle Content Management; it just needs to be enabled by your administrator. See Configure Sites and Assets Settings in *Administering Oracle Content Management*.

Create Sites

To create a site, select a template, name the site, and add content. If governance is enabled, before you can add content you must submit your site request, then, after it's approved, add content.

Before you can create a site, your administrator must enable site creation and make templates available to you. If you don’t see the **Create** option on the Sites page or there aren't any templates available, contact your administrator. See Get Started with Sites and Understand Site Governance.

If you're creating an enterprise site, which enables the use of assets, multilingual sites, and secure site taxonomies, your site will be associated with a repository, a localization policy, possibly a default language, and if the template requires the use of a site security taxonomy (SST), a parent category. If a repository and localization policy aren't available, create them, or ask your content administrator to create them. If there isn't an option to choose a parent category, then it is likely the site template you selected doesn't require using a site security taxonomy. If it does, the template would have a small badge in the corner.



> **Note:**
>
> With Oracle Content Management Starter Edition, translations are unavailable, governance can't be enabled, and only one site can be created. For a full feature set and unlimited sites, upgrade to Oracle Content Management Premium Edition.

1. Click **Sites**.
2. Click **Create**.
3. On the Choose Template page, select the template for your site.
   - If governance is enabled, the template will determine if the site you create is a standard site (doesn't use an asset repository or localization policy) or an enterprise site (uses an asset repository and localization policy).
4. On the Configure Site page, you see what approval is needed before the site is created, the minimum level of security required for the site, and images of the site pages. If approval is limited to specific people, click **Show Approvers** to see who can approve your site request.
   - If governance is not enabled, select **Standard** or **Enterprise** for the type of site you want to create.
   - If you're creating a standard site, click **Next** to go to the next page.
   - If you're creating an enterprise site, complete the following steps:

   a. Choose the default repository used to manage the content for the site. If you are a repository administrator and no repository is available or you would like to use a new repository, select **Create a new repository** and follow the steps to create a repository. For information about using multiple repositories in a site, see Give a Site Access to Multiple Repositories in *Managing Assets with Oracle Content Management*.

   b. If the template you selected requires the use of a site security taxonomy, select the parent category for the site. Your site assets will be categorized into a site category created for the new site under the selected parent category. The custom roles created for the new site will use this site category to restrict access to the site assets. For information on using a site security taxonomy to control access to assets in a shared repository, see Manage Sites in a Shared Repository.

c. Select a localization policy policy for the site. If you are a repository administrator and no policy is available or you would like to use a new policy, select **Create a new localization policy** and follow the steps to create a policy.

d. If the template's localization policy doesn't have a set default language, select one now. You see only those languages that are required by the localization policy.

e. Click **Next** to go to the next page.

5. On the Add Details page, enter the following information.

- Enter a name for the site. The name is used in the site URL. You can use letters, numbers, underscores (_), and hyphens (-). The URL is case sensitive. If you enter a space, it's automatically replaced with a hyphen. Do not use the name of an existing site with different capitalization. For example, if a site exists called *ABC*, do not create another site called *Abc*.

> **Note:**
>
> Don’t use the following names for site templates, themes, components, sites, or site pages: authsite, content, pages, scstemplate_*, _comps, _components, _compsdelivery, _idcservice , _sitescloud, _sitesclouddelivery, _themes, _themesdelivery. Although you can use the following names for site pages, don’t use them for site templates, themes, components, or sites: documents, sites.

- If you're creating an enterprise site, the template policy may allow you to edit the default site prefix for friendly URL values. This prefix will be appended to content item slug values (the part of the URL specific to the page or asset).
- If you're creating an enterprise site that requires a site security taxonomy, then repository assets are automatically duplicated by default. If creating from an enterprise template that does not require a site security taxonomy, you're given the option to duplicate, override, or update assets in the selected repository.

Option	Description
Duplicate Assets	New assets with unique GUIDs are created in a selected repository.
Override Assets	Existing Assets in a selected repository are replaced with versions from the template with same GUIDs.
Update Assets	Existing assets in a selected repository are updated to new revisions from the template only if template assets are newer with same GUIDs. Otherwise if assets in repository are newer than those in template, newer assets remain in repository at current versions.

These options can be useful if there is no need to duplicate assets that already exist, or if you only need to update those that are newer in the template.

- Optionally, enter a description for the site.
- Optionally, enter a justification for this site request. This will help the site administrator determine whether the request should be approved.

6. Click **Finish**.

If the site request requires approval, an email is sent to the person who needs to approve it. If approved, you will receive an email notifying you it has been approved and the site will be created. If your request is denied, you will received an email notifying you it has been denied. You can see your pending requests on the Sites page, by selecting **Requests** in the filter menu. If the request was denied, you should see a message as to why it was denied, so you can correct the issue and submit your request again.

If the site request is set to be automatically approved, the site is automatically created. A progress bar shows the new site name and creation status. When the site is created, the name appears in the list of sites, and its status is offline.

If you created an enterprise site, a corresponding site collection is created in the repository you selected. If you share the site with a user, the user has the same permissions on the associated collection.

After your site is created, you can share the site, change the security, add and edit content, publish the site, and bring it online. See Manage Sites and Site Settings and Edit Sites.

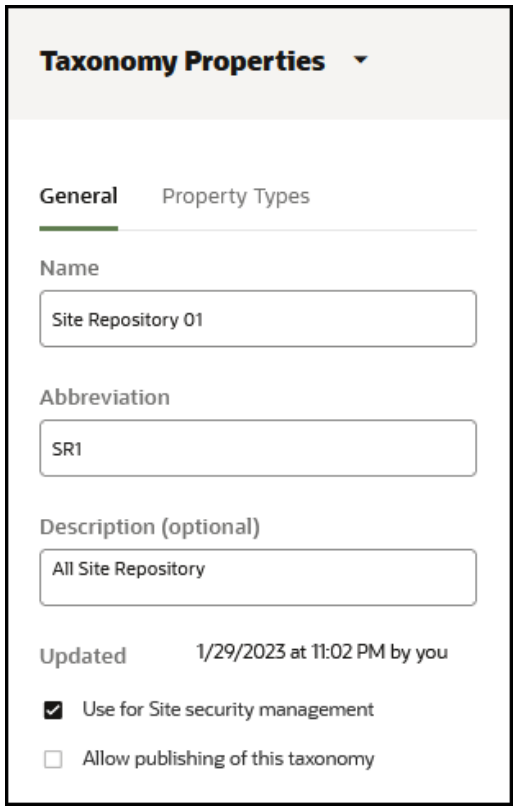
Manage Sites in a Shared Repository

Many companies have an extensive number of sites that are created and managed by various business teams. Sharing the same repository for all sites can ease overhead and streamline site management.

Sites can be public, such as a marketing site, internal, such as a team site, or secure so that only specific users, internal and external, can see them. In all cases, the business team managing the site typically does not want people outside the team to edit or even see content used on the site before it is published. Creating a repository for each site in order to segregate each site's content is not a tenable solution, as that would cause an overhead of managing potentially thousands of repositories.

Instead, Oracle Content Management can apply granular permissions on assets to separate site content based on category, allowing sites to share one repository while controlling access to specific sites and site assets. Separate repositories would then be used only if access to confidential content is highly restricted, such as financial documents or information on mergers and aquisitions.

To manage access to a shared repository, a site developer creates or edits a template to require the use of a site security taxonomy (SST). They then either create a new taxonomy, or edit the general properties of an existing taxonomy, and specifiy it for use with site security management. Once a taxonomy is done, it must be promoted and added to the repository being used by the site.



Note:

SST templates automatically create categories in a taxonomy during site creation, and so taxonomies used with site security management must be promoted before they can be used when creating sites. If a draft of the taxonomy exists when an SST template is used to create a site, even if a version of the taxonomy has been promoted, then site creation fails.

When a member of the team creates a site using an SST template (denoted by a badge icon in the template corner), they are required to select a parent category for site content, and name the site. When the site is created, a new site category is automatically inserted into the taxonomy as a child of the parent category, using the site name. For example:

- <*selected_parent_category*>
  - <*site-name*>_site_category

All site assets are categorized with the new site category. This allows assets for each site to use a single repository while being discrete.

In addition, two new groups also named using the site name are created and added to the repository:

- <*site-name*>_viewer
- <*site-name*>_contributor

> **Note:**
>
> Do not remove site assets from the site category. Doing so removes access to site assets from the site content viewer and contributor groups when those assets are also catogorized outside of the secure site taxonomy.

A person is automatically assigned to a group when an SST site is shared with them, based on the role they are given. A person given the view or downloader role is assigned to the site content viewer group. A person given the manager or contributor role is assigned to the site content contributor group.

By default, those with the viewer role can see all asset types in the site category of the repository. Those with the contributor role can categorize, create, edit, and delete all asset types in the site category. These default permissions can be edited to refine access at a more granular level if needed, controlling who has access to which sites and content. For more information, see Granular Permissions.

Copy Sites

You can copy a site to get a head start. Everything from the original site, including the theme, all outstanding updates, the pages, the page content, recommendations, all other assets such as images, and the policies are copied to the new site under the new name you provide.

> **Note:**
>
> If you are copying a site that uses content from multiple repositories, you need to do so using the Oracle Content Management Toolkit. See Develop with Oracle Content Management Toolkit and Use the cec Command-Line Utility

1. Click **Sites**.
2. Select the site you want to copy and choose **Copy** from the right-click menu or  in the actions bar.
3. On the Configure Site page, you see what approval is needed before the site is created, the minimum level of security required for the site, and images of the site pages. Complete the following steps:
   a. In the **Copy** drop-down list, select whether to copy only the base site or to include updates. If you include updates, your copied site will include any outstanding updates from the original site. If you're copying a standard site, click the arrow to go to the next page, and skip to step 4.
   b. If you're copying an enterprise site, the default asset repository, localization policy, required languages, and default language are set based on the original site. You can optionally choose a different repository to manage the content for the site.
   c. Click **Next** to go to the next page.
4. On the Add Details page, enter the following information.

- Enter a name for the site. The name is used in the site URL. You can use letters, numbers, underscores (_), and hyphens (-). The URL is case sensitive. If you enter a space, it's automatically replaced with a hyphen. Do not use the name of an existing site with different capitalization. For example, if a site exists called *ABC*, do not create another site called *Abc*.

  > **Note:**
  >
  > Don’t use the following names for site templates, themes, components, sites, or site pages: authsite, content, pages, scstemplate_*, _comps, _components, _compsdelivery, _idcservice , _sitescloud, _sitesclouddelivery, _themes, _themesdelivery. Although you can use the following names for site pages, don’t use them for site templates, themes, components, or sites: documents, sites.

- If you're copying an enterprise site, you can optionally edit the default site prefix for friendly URL values. This prefix will be appended to content item slug values (the part of the URL specific to the page or asset).
- Enter an optional description for the site.
- Enter an optional justification for this site request. This will help the site administrator determine whether the request should be approved.

5. Click **Finish**.

If the site request requires approval, the site will be created after the site administrator approves it. You can see your pending requests on the Sites page, by selecting **Requests** in the filter menu. If your request is denied, you should see a message as to why it was denied, so you can correct the issue and submit your request again.

If the site request is set to be automatically approved, the site is automatically created. A progress bar shows the new site name and creation status. When the site is created, the name appears in the list of sites, and its status is offline.

If you copied an enterprise site, a corresponding site collection is created in the repository you selected. If you share the site with a user, the user has the same permissions on the associated collection.

After your site is created, you can share the site, change the security, add and edit content, publish the site, and bring it online. If you are sharing a site that has access to multiple repositories, only the default repository is shared. Any additional repository will need to be shared separately. See Manage Sites and Site Settings and Edit Sites.

Export and Import Sites

When you need to move a site from one Oracle Content Management instance to another, rather than copy the site, it is easier to export it from one instance and import it to another instance. This is useful, for example, when moving a site from testing to production.

Export a Site

You export a site to a folder with one or more .zip site archives, depending the size of the site and number of assets it uses. That folder can easily be moved for import into another Oracle Content Management instance.

> **Note:**
>
> If a site is configured to use assets from more than one repository, the exported site archive will include assets for the primary repository only. Use Oracle Content Management Toolkit if you must export assets from multiple repositories.

To export a site:

1. Click **Sites** and select the site you want to export on the Sites page.
2. Click  on the action bar or select **Export** from the right-click menu.
3. Specify a name for the export job, add an optional description, and click **Select** to navigate to a folder in Oracle Content Management as the destination.
4. Optionally, enable **Include unpublished content items and digital assets in your job**. If enabled, the export job will include unpublished content items and digital assets. If disabled, then published versions are exported.
5. Click **Export**. The site is exported to one or more .zip archives created in the folder you specified. If the exported site uses a large number of assets, then more than one .zip archive may be generated. There is also a .zip archive of the export report that you can use to validate the site export.

   The .zip archives and report can be downloaded and used to import the site to another Oracle Content Management instance or stored as a site backup.

Import a Site

You import a site from a .zip site archive that was exported from an Oracle Content Management site. Exported sites contain one or more .zip archives, depending the the size of the site and the number of assets it uses, as well as a report .zip archive.

> **Note:**
>
> If you import a site to a different server, some links in the site may not be valid in the new server context. If the site uses reference links to images or other content rather than copying the content directly into the site, that content is not available on the new server. Even if you copy the content to the new server, the content will have a different internal ID, and the link will not be valid.

To import a site:

1. Upload the .zip archive files to a folder in the Documents section of the Oracle Content Management instance to which you are importing the site.
2. Click **Import Site** on the Sites page.
3. Select the folder containing a site archive in the Oracle Content Management instance to which you are importing the site. If the site archive has not yet been

uploaded, click **Upload** to upload a .zip archive of the site to the Documents section of the target instance.

4. Once you have selected a folder containing the site archive, click **Next** and then choose the import type.
   - **Update Site from Archive**—This type replaces a site that already exists on an Oracle Content Management instance.
   - **Duplicate Site from Archive**—This type copies the archived site to an Oracle Content Management instance.
   - **Import Site from Archive**—This type creates a new site from the selected archive package.
5. Click **Next**.
6. Select the repository to use for the site assets or create a new one.

> **Note:**
>
> **Not Ready for Use Assets** must be enabled in the selected repository or the import will fail. This allows assets to be saved with default or missing values in required fields, to be completed later. All required fields must still be completed before an asset can be submitted for review or publication.
>
> You can enable **Not Ready for Use Assets** when creating a new repository, or enable it on an existing repository by editing the repository if you are a repository administrator.

7. Specify how the selected repository should handle assets.
8. Verify or change the localization policy and publishing channels.
9. Optionally, enable **Accept all validation warning messages** to override any conflicting changes to the import.
   - If you enable this, the import job will complete regardless of any warnings. Once the job has completed, you can open the **Report** tab of the job details panel to review any issues or download a full report.
   - If you leave this option disabled, the job will pause each time it encounters a problem and you will need to manually assess the validation and resume the job. To resume a paused job:
   a. In the job list, click the pending job you need to review to open the job details panel.
   b. Click the **Report** tab to review the validation summary report. You may need to select the report section to find the issue you need to review.
   c. Expand the issue to see why the import has paused. If the reason is acceptable, click **Resume Import**. If it is not acceptable, return to the job list and cancel the import.

View Export Site and Import Site Job Details

To view a listing of export site or import site jobs, select the option from the **View Jobs** menu on the Sites page to open the jobs panel. In the listing, you can search for jobs, download log files, see job status, and resubmit jobs that completed, but with errors.

Status Icon	Meaning
	**Complete**—The job has successfully finished without issues.
	**Complete with Errors**—The job finished but has some minor issues requiring attention. Review the reports and address the errors, then resubmit.
	**Pending**—The job is currently in progress.
	**Paused**—The job is currently in progress but has paused and needs attention. Review the reports and address the errors.
	**Canceled**—The job has been canceled.
	**Failed**—The job has failed. Review the reports to help fix issues and resubmit.

Select a job to enable the action bar with actions to cancel, delete, or view details of a job (). To see details, click on a job name to open a panel with a tab showing job details and a report tab showing a summary of validation warnings, if any, such as changes in content types or other problems with the specific job.

If the import job has completed but has errors (), selecting the job also gives you the option to resubmit the job (). This option is available to you only if you created the import job.

Manage Site Requests

If site governance is enabled, sites might need to be approved before they're created. You can view pending site requests on the Sites page, with the Requests filter.

> **Note:**
>
> With Oracle Content Management Starter Edition, governance can't be enabled. For a full feature set and unlimited sites, upgrade to Oracle Content Management Premium Edition.

See Understand Site Governance.

To view pending site requests, on the Sites page, in the filter menu, select **Requests**. If you're a *site administrator*, you see all site requests in your system, including requests that require approval from other people. If you're an *approver*, you see all site requests for which you are an approver. *All users* see site requests they've submitted.

In the requests list, you can see a thumbnail of the site, the name of the site, who requested it, when it was requested, the site description, the status of the request, and an icon showing whether a site requires log in or not.

Depending on your role, you can perform the following actions:

- To view more details of the request, such as the minimum security required, template used, an optional justification for the site, and thumbnails, click the site name or select the request, and then click **View**.
- If you're a site administrator or an approver, you can approve the request by selecting it, and then clicking **Approve**. The site will automatically be created after it's approved.
- If you're a site administrator or an approver, you can deny the request by viewing the details, and then clicking **Reject**. Enter a reason for denying the request, and then click **Reject**.
  Although you can alternatively deny the request by selecting it in the list of requests, and then clicking **Reject**, you won't be able to add a reason for rejecting it.
- If your request failed or was rejected, you can view the details, edit your request as necessary, and then **Resubmit** the request.
- If you're the site creator, you can delete your request by selecting it, and then clicking **Delete**.

> **Note:**
>
> If the site requester has been deleted, a site administrator can delete the site request.

After the site is created, you can share the site, change the security, add and edit content, publish the site, and bring it online. See Manage Sites and Site Settings and Edit Sites.

Change Site Request Details

The site request details tab shows the site name, description, template used, justification, the owner (the person making the request), and the site images.

You can update site request details if you created the request (you're the request owner).

To view or change site request details:

1. Click **Sites** and then, in the sites menu, click **Requests**.
2. Click the request you want to view or edit.
3. If you're the approver, you can **Approve** or **Reject** the request. If you're the requester, you can edit the information as necessary, and then **Resubmit** your request.

View Site Request Policies

The site request policies tab shows the type of site (standard or enterprise), who can access the published site, and any approval required before the site is created. For enterprise sites, you also see the asset repository, localization policy, required language, and default language used for the site.

To view site request policies:

1. Click **Sites** and then, in the sites menu, click **Requests**.
2. Click the request you want to view or edit, and then click the **Policies** tab.
3. If you're the approver, you can **Approve** or **Reject** the request. If you're the requester, you can change the asset repository as necessary, and then **Resubmit** your request.

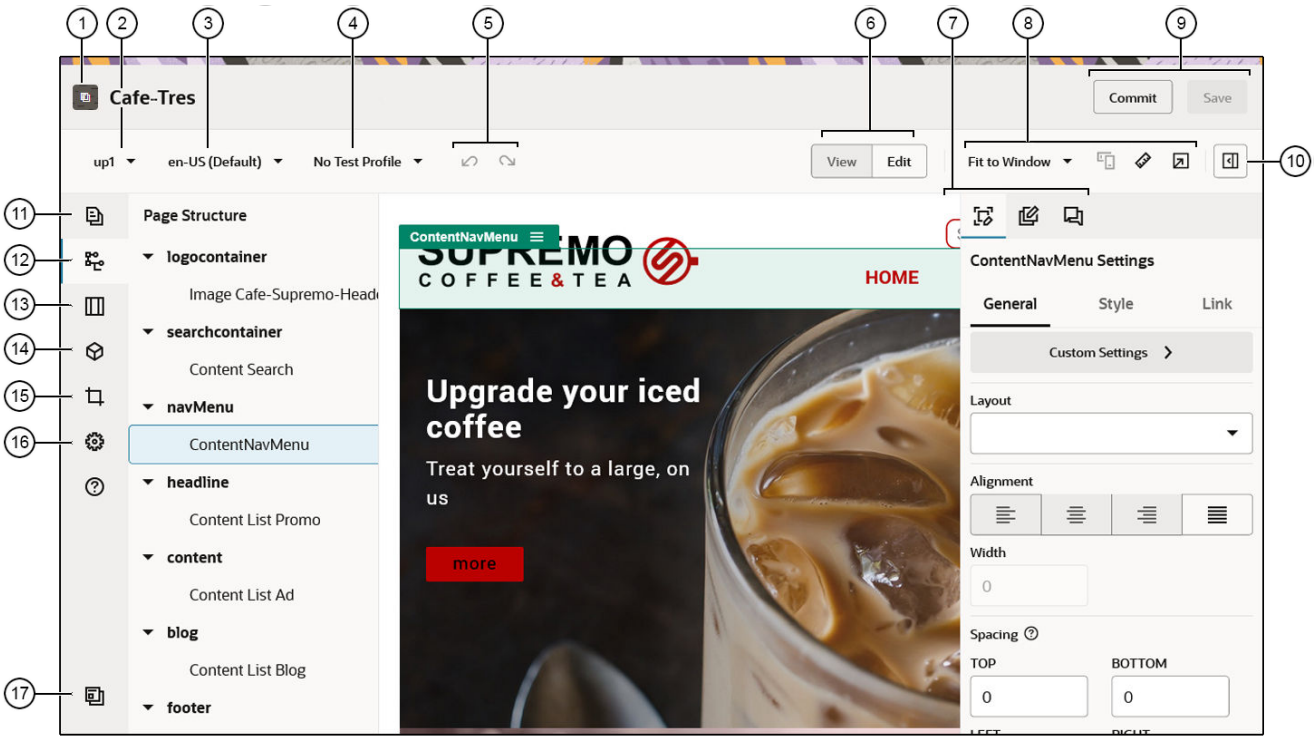
5

Edit Sites

Let’s get to know the editor a little and see what you can do with it.

Get to Know the Site Builder Page

When you edit an existing update or create an update for a site, the update opens in Site Builder. Take a minute to become familiar with the layout of the page and the tools available.



Here are a few things to note:

Callout	Description
1	The **name** of your site is listed next to the site icon.
2	Check the **update** for the name of the update you're currently working on. If there's more than one update available, you can switch to a different update.
3	Check the **language** to see which version you're currently working on.
4	**Test Profiles** give you the opportunity to create profiles with various audience attributes and use the profiles to represent different site visitors to test recommendations. Test profiles are set up when a site administrator creates a recommendation.
5	Use **Undo** to reverse the last edit or change you made in the editor. Use **Redo** to reapply the most recent change you reversed using undo. You can use undo multiple times to reverse a series of changes in the current update including changes in content, style, and page organization.
6	Select **View** to see a preview of the site and **Edit** and make any changes to an update.
7	Option menu for the side panel. Click (item 10) to open the side panel. What you can do in the side panel depends on the option you select. • Element settings opens the options for configuring component elements selected on your page. Available options will depend on the type of element selected. This option is only available when editing a site. • Page settings opens the options for configuring your page settings. Available settings include page name, page layout, mobile layout and more. There are also tabs to add custom page properties and configure backgrounds. This option is only available when editing a site. • Click to access or start a conversation for the site, where you can create or participate in a discussion about this site. See Use Conversations in *Collaborating on Documents with Oracle Content Management*. Use the **annotation controls** to add annotations or view existing annotations. – To add an annotation, click , click the component where you want to add the annotation, then enter your comment in the text box, and click **Post**. See Add Annotations in *Collaborating on Documents with Oracle Content Management*. – To view annotations, click . When you click an annotation, the annotation note will get focus in the conversation.

Callout	Description
8	Use the **preview options** to see how the content will look under different circumstances. You can navigate to any page and see the base site with the changes from the current update applied. Click **Fit to Window** and choose a dimension to view the page as it will appear on a device with that screen size. Several sizes are given and you can create your own device size. Click  to see markings. Click an interval on the ruler to quickly see how the site appears at different sizes. You can also select  to see how a site will appear on a mobile device depending on orientation. Themes with a responsive design automatically arrange page content for the best use on the selected screen size. Click  to preview the site. This shows the page as it will appear to your site visitors, without slot and component borders and other visual aids used while editing. You can use the links on the pages to move around the site, including links in navigation menus, links in text, and so on. Links to other sites open in a new window for security purposes. Links to pages in the current site open in the same window or a new window depending on link target you specify. **Note:** Links to site pages don't work in preview mode.
9	Use the available **options** to process your changes. Click **Commit** to merge your changes to the **base site**, or click **Save** to save your changes to the **update**.
10	Click  to open and close the side panel. What you can do in the side panel depends on the option you select (see item 7 for a list of options).

Click an icon (11–17) in the sidebar to open the panel and manage pages, add components, and more. Click the selected icon again to close the panel.

Note:

You must be set to **Edit** before you can use the tools in the sidebar.

Callout	Click...	To...
11		Edit and add pages, manage nested page structures, reorganize pages, and change page settings.

Callout	Click...	To...
12		View the structure of your page with the elements displayed in a hierarchy. Drag-and-drop the elements in the hierarchy to move them on the page. To adjust component settings, select an element in the hierarchy, open the right side panel and select the element settings option (). If a component or slot has become orphaned, usually because of switching to a different layout that has a different set of slots, then those items get moved out of the page layout grid and you will find them at the bottom of the hierarchy panel in the **Orphaned Component** and **Orphaned Slots** sections. Components can be moved back into the page by dragging them into one of the current slots, and orphaned slots can be deleted only, to clean up page data. From there you can either move them back into the grid or delete them from the site. Note that only the top-level of an orphaned hierarchy is listed in the orphaned component section. For example, if a section layout has orphaned components, the orphaned component section will only show the section layout, not any orphaned components in it. To see them, you need to drag the section layout into the frame. Also, if any inline components are used, they're displayed at the top of the hierarchy in the **Inline Components** section.
13		Add section layouts, such as horizontal or vertical layouts.
14		Add assets to your site. You can select any assets that are in the repository that was selected when the site was created.
15		Insert components into your site. You can choose from different types of components, such as themed, custom, seeded, or see all available components.
16		Adjust the settings for the site, such as keywords for search engine optimization (SEO) and site redirects. • SEO • Site • Theme • Redirects • Analytics • Properties • Locales • RSS Feeds
17		Return to the Sites page to manage your sites.

Understand Site Updates

An update is a named collection of changes to the current base site. The changes remain in the update until you commit them and permanently update the base site.

Each time you view or edit a site in the editor, you use an update. Any changes you make in the editor are part of that update. You can have one or more updates and you can continue to add changes to an update over time. Updates give you flexibility with

how you manage edits to a site. For example, you can have several people working on their own updates for different parts of the site. You can review and modify individual updates, and when you're ready, you can commit the updates to the base site. You'll still need to publish the site to make the updates available online.

You can organize updates in several ways:

- Page-specific changes in an update named for the page
- Changes made by a specific user in an update with the user's name
- Changes made on a given day or for a particular project milestone

> **Note:**
>
> An update shows the changes in that update against the base site. Although it's easy to switch between available updates within the editor, you can view only one update at a time. If there are multiple updates to a given page, you might not know if you have multiple changes to the same content area. To prevent conflicting changes on a page, target individual updates to specific pages or areas of the site.

When you commit the changes in the current update, the changes are made to the base site and the update is deleted. You'll need to publish the site (by bringing it online or republishing it) to make those changes visible online to anyone with access to the site.

To edit a site:

1. Select the site in the list and choose **Open** in the right-click menu or  in the actions bar.
2. The editor opens in preview mode. To make changes or to use the navigation options in the sidebar, make sure the Edit switch  is set to **Edit**.
3. If this is the first update for a site, enter a name for the update and an optional description, then click **OK**. You can use letters, numbers, underscores (_), and hyphens (-) in the name. If you enter a space, it's automatically replaced with a hyphen. If you already have updates to the site, select an update from the list and click  .
4. To edit a particular page, locate the page using the site tree in the left sidebar, searching for the page, or using the site's own navigation. To show the site tree, click  on the left, then click  .
5. Add and change page content as necessary. Select **Fit to Window** to use the layout options to see how the page will look on different devices and with different sizes. Several sizes are given and you can create your own device size. Click  to see markings. Click an interval on the ruler to quickly see how the site appears at different sizes. You can also select  to see how a site will appear on a mobile device depending on orientation.
6. When you're done editing your site, save your changes in one of the following ways:
   - Click **Save** to save your changes to the current update. You can continue working in the current update or return to the update later.
   - Click **Commit** to apply your changes to the base site.

You'll still need to publish your site before your website users see the changes. See Bring a Site Online or Take It Offline or Publish Site Changes.

When you commit the changes in the current update, the changes are made to the base site and the update is deleted. You must use an active update each time you view or edit a site in the editor, so you're returned to the site list where you can create a new update.

With an update, the process is linear:

- The original site (base site) exists. Let's call it Version 1.
- You create an update. When you merge the update (with Commit), the update is permanently written to the existing site. You now have a new version (Version 2) of your base site.
- If you create and merge another update, then the update is permanently written to the existing site. You now have a new version (Version 3) of your base site.

With updates, remember that:

Current Base Site + A Merged Update = New Version of the Base Site

Use an Update

Each time you edit a site in the editor, you use an update. Any changes you make in the editor are part of that update.

To create and use an update:

1. On the Sites page, select the site and choose **Open** in the right-click menu or click  in the actions bar.
2. Set the Edit switch  to Edit.
3. If this is the first update for a site, enter a name for the update and an optional description, then click **OK**. You can use letters, numbers, underscores (_), and hyphens (-). If you enter a space, it's automatically replaced with a hyphen. To use an existing update, click the update in the list.
4. The name of the site and current update are displayed in the top bar in the editor. If you have multiple updates, you can switch updates by selecting a different update from the list of updates.
5. Add and change page content as necessary. Use different display size options to see how the page will look on different devices.
6. When you're done editing, click **Save**. When you're ready to merge your changes with the base site, you can commit the update. You'll still need to publish the site before website users see the changes. See Publish Site Changes.

Editing Tips and Tricks

Here are some things about the editor that will help you get started.

- Hiding or Showing the Sidebar
- Getting Around

- Managing Pages
- Drag and Drop Editing
- Components
- Adjusting the Size and Spacing of Components
- Editable Layout Content
- Styles and Formatting
- Undoing Your Changes

Hiding or Showing the Sidebar

Click to show options for managing and editing pages and page content.

Set to **Edit** to access the sidebar. Click to hide the sidebar and increase your viewing area when you preview a page.

Getting Around

To select another page using the site tree, click in the sidebar. Use the search box at the top of the page list to search for a page. You can also use the site navigation or links on the pages themselves.

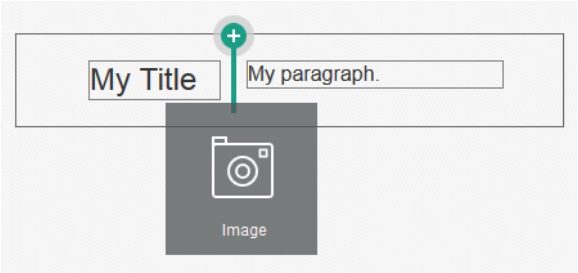
Any changes you make on a page are stored when you switch to another page. You can also click **Save** to save changes in the current update.

Managing Pages

- To add a page, select the level or branch where you want to add the page, then click **Add Page**, or, to add a child page, click , then click . "New Page" is added to the bottom of the site tree and you're prompted to name the page and specify other settings.
- To delete a page, select it, then click .
- To move a page, select it, and drag it to the new position. Alternatively, you can cut and paste the page to a new location. To cut the page, click , click . To paste the page, select the branch where you want to paste it, click , then click .
- To copy a page, select it, click , then click . To paste the page, select the branch where you want to paste it, click , then click .
- To change page settings, such as metadata, header, footer, and other options, click . See Change Page Settings.

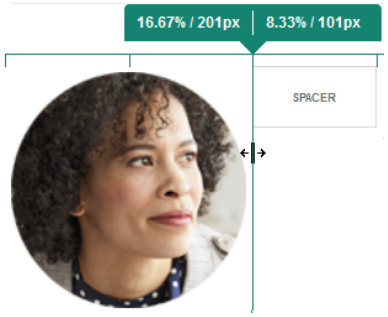
Drag and Drop Editing

To add a component from the sidebar, or to move a component on the page, click, drag, and drop the item to the location on the page. When you drag an item to the page, the boundaries of available slots and any existing items are shown. A placement bar indicates where the new content can go (above, below, left, right):



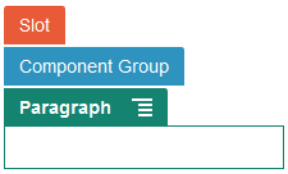
You can have multiple items in a slot and move items on the page just by dragging them to a new location. You can also adjust the relative width of two components in a slot by clicking and dragging the boundary between the two components. The component snaps to the next grid line indicated in the “ruler” displayed above the components.

The size of each component is displayed both in pixels and as a percent of the available space in the slot. To adjust widths to values other that those defined by the grid, press and hold the **Ctrl** key while you click and drag the component boundary.

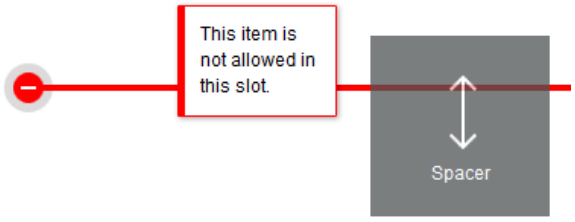


Components

After you place a component on the page, you can adjust the alignment, spacing, and other properties by selecting the item, clicking the item’s menu icon, and choosing **Settings**. If you click the component name instead of the menu icon, you can see and select the menu icon for the slot and component group (if the current component is part of a component group). If you select one of the other tabs, you can see the menu icon and set properties for that element:



Theme designers can specify which components are included with the theme, so some components available with one site may not be available with another. Theme designers can also specify which components are allowed in a given slot in a given page layout. If a component isn’t allowed in a particular slot, the placement bar changes color and symbol (minus) and a message similar to the following is displayed:



Adjusting the Size and Spacing of Components

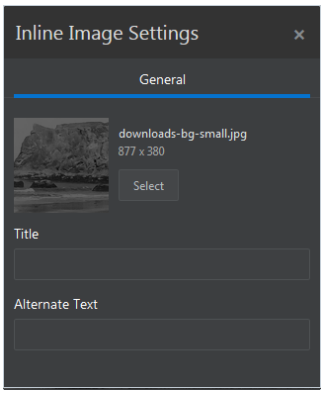
You can adjust the size of many components, such as galleries or images, to different CSS units, as well as adjust the spacing around the component. For example:

- px (pixels): the default. If only a numeric value is specified, pixels is assumed.
- % (percent): sets the item to a percentage of its parent HTML element size. Example: 25%
- em: sets the size of the component in em-spaces. Example: 20em
- vw: sets the size as a percentage of the width of the viewing area and is responsive to the size of the viewing area. Example: 10vw

Editable Layout Content

Theme designers can add “built-in” content to page layouts, such as copyright notices, that can’t be changed in the editor. Theme designers can also designate simple text and image content as editable, including digital assets. This allows a contributor to change the text or image content, but not alter the location or other layout attributes. Editable text elements have a menu with options for specifying bold, italic, and underline text and for changing or removing a link:

Editable graphic elements have a settings icon  and panel where you can specify an image, title, and alternate text.

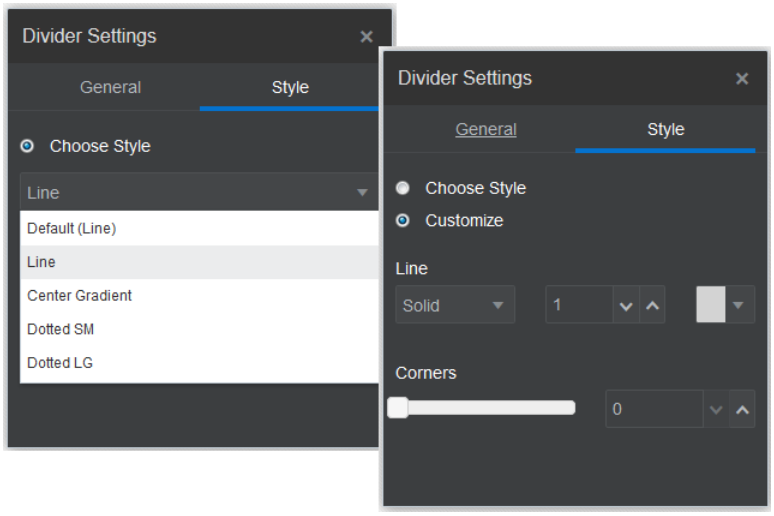


The frame that encloses the text or image adjusts to accommodate the length of the text or the size of the image.

> **Note:**
>
> The changes you make apply to the current page only. The original content is stored with the layout in the theme and is the default when the layout is first applied to a page.

Styles and Formatting

Most components have one or more base styles defined by the theme that specify aspects of the component's appearance. You can easily switch styles or override a style. To choose between available styles, open the component's Settings panel, click the **Style** tab, click **Choose Style**, and choose a style from the menu. To specify your own values for the properties specified in the style definition, click **Customize** and specify the formatting options.



Undoing Your Changes

Use to reverse the last edit or change you made in the editor. You can use undo multiple times to reverse a series of changes.

Use to reapply the most recent change you reversed using undo. You can use redo multiple times if you have used undo multiple times in succession.

You can undo changes in content, style, and page organization in the current update. Some actions aren't included in the undo chain:

- If you switch to a different update, the undo chain is reset and you can't undo the changes made in the update you worked on earlier. Within an update, you can undo changes even after you save them.
- If you change views in the editor, such as switching pages or changing the size for a given page, you must manually reverse those types of changes.
- If you edit text components, such as titles or paragraphs, the text editor has its own undo chain. When you leave the text editor, you can no longer undo those changes.

Use Styles and Formatting

Most components have one or more base styles defined by the theme that specify aspects of the component's appearance. You can easily switch styles or override a style with options that you choose.

1. Navigate to the page you want to edit and make sure that  is set to **Edit**.
2. To set the base style for a component, click the component's menu icon  and choose **Settings**. Click the **Style** tab.
   - To use a style from the site's theme, click **Choose Style** and choose the style from the menu. Styles are defined for individual components, so the list of styles may vary. For example, the styling for an image is different from that for a paragraph.
   - To specify your own base formatting options, click **Customize** and specify the formatting options.
3. To copy and paste the base style to one or more similar components, click the component's menu icon  and choose **Copy *component* Style**. Click a similar component's menu icon  and choose **Paste *component* Style**.
4. To format the text within a title or paragraph component, click in a text component. A tool bar with formatting options is shown. Select the text that you want to format, then select any of the options, such as font, color, or alignment. The changes you make are applied immediately. To remove the formatting, select the text and click  . Formatting changes are applied over the base style. If you change the base style, the overrides stay in place.

Customize Site Settings

You can customize certain settings in each site.

- SEO—Set keywords, edit headers and footers, and enable various other options for search engine optimization.
- Site—Set favorite icons, edit the controller.html file, add sitemap and robot files, and select other options.
- Redirects—Upload a custom redirects.json file to configure permanent or temporary redirects.
- Analytics—enable data recording of asset consumption to track click, load, view, play, and download events.
- Properties—Define custom site properties as name/value pairs for use in components and scripts.
- Locales—Provide aliases for each locale specified in the site's localization policy to shorten and simplify URLs.

Edit Site Settings

You can customize site settings to add site icons, a controller file, a sitemap, robot files, auxiliary files, and to specify a map provider.

These settings are stored in an update until you commit the update. After publishing, the files are stored in the root folder of the theme so any sites using that theme will use these files.

To change the icons and link behavior for the entire site:

1. Open a site for editing.
2. Click  in the sidebar and then click  **Site**.
3. In the **Favorite Icons** section, choose an image to use for the site when the site is minimized in a browser or on a different platform, such as a mobile device. The icon must be stored as a digital asset that you can access. Click **Select file to upload**, navigate to the icon and select it, then click **OK**. Customization is needed for the favorite icon to work all browsers:

   - **For Chrome and Safari** there must be a reference to the favorite icon, including the site prefix, in the controller file, as shown in this example:

     ```
     <link rel="shortcut icon" href="/mySitePrefix/favicon.ico" />
     ```

     For Internet Explorer 11 and Firefox, a similar entry must be included in the page templates. However, a page template is part of a theme and it might be used in multiple sites. Therefore, it can't use a fixed site prefix and must use a token instead. See this example:

     ```
     <link rel="shortcut icon" href="<!--$SCS_SITE_PATH-->/
     favicon.ico" />
     ```

     The token will be swapped with the site prefix when the page is delivered.

   Note that Internet Explorer and Firefox load a favorite icon from the controller and again from the page template. Therefore, the icon will flash unless the same one is referenced from both the controller and the page template. Chrome and Safari only load the icon that's referenced from the controller.

4. In the **Controller File** section, you can add a file that alters the way browsers process link requests. In addition to referencing favorite icons, you can also add OpenGraph tags or metadata tags for webmaster site verification, or for sharing the site on social media. You download the default controller file and edit it, or upload your own file. The file must be stored as a digital asset that you can access. Click **Select file to upload**, navigate to the file and select it, then click **OK**.

   For example, if your site contains a Facebook Share button, you may want to provide metadata that Facebook can use to show details about your site in Facebook, as in this example:

   ```
   <meta property="og:image" content="https://my.domain.com/fb-
   image.jpg"/>
   <meta property="og:title" content="My Site Title on FB!"/>
   <meta property="og:url" content="https://my.domain.com"/>
   <meta property="og:site_name" content="My Site Name on FB"/>
   ```

You can add a Google webmaster verification tag, similar to this example:

```
<meta name="google-site-verification"
content="GCVURS9d2fP6jev5upt0Yt1AIp71C9D__ALqS8pg" />
```

5. In the **Sitemap and Robot Files** section, you can upload a custom sitemap and robot files. A *sitemap* is an XML file that you can use to list URLs for a website and information about each URL, such as when it was last updated. A *robot file* is a text file you can create to instruct search engine robots how to index pages on your website. The files must be stored as a digital assets that you can access. Click **Select file to upload**, navigate to the file and select it, then click **OK**.
6. In the **Auxiliary Files** section, you can upload more files if needed, such as those needed to verify site ownership. The file must be stored as a digital asset that you can access. Click **Select file to upload**, navigate to the file and select it, then click **OK**.
7. You can choose a provider for the map component and the links used. Select either **Oracle Maps** or **Google Maps**.
8. When you publish the update, the changes are published and put into use.

Set Search Engine Properties

You can provide keywords and text to help search engines identify the content of the site.

You can define search engine optimization (SEO) settings at the site level and at the page level. The site-level settings augment or override similar settings for individual pages as described in the table below.

Option	Site Level	Page Level
Description or Page Description	Provides general information about the site that isn't included in the site itself. The site description is included on each page in the site This description is also used as the page-level description for the home page only if there's no value set using **Page Settings** for the home page.	Provides general information about the page that isn't included in the page itself. The page description is in addition to the site description included with every page in the site.
Keywords	Identifies terms or concepts that apply to all pages on the site. These values are added (appended) to the keywords specified for individual pages.	Identifies terms or concepts that apply to the individual page. Page keywords may be useful to identify terms or concepts that don't appear in the text of the page or that appear in images.
Header or Page Header	Add scripting or tags for analytics or tracking to your site. The site header content is included on each page in the site.	Add scripting or tags for analytics or tracking to your site. The page header content is in addition to the site header content included on each page.
Footer or Page Footer	Add scripting or tags for analytics or tracking to your site. The site footer content is included on each page in the site.	Add scripting or tags for analytics or tracking to your site. The page footer content is in addition to the site footer content included on each page.

If you select the following search exclusion options at the site level, the setting applies to all pages and overrides the setting on the individual pages. If you don't select the option at the site level, then only those pages that individually specify the option use the option.

Option	Site Level	Page Level
Hide from search engines	If selected, add the NOINDEX meta tag to every page so that search engines won't index the content of any page on the site. In this case, the site and all its pages won't shown up in the web search results.	If selected, add the NOINDEX meta tag to the current page so that search engines won't index the content of the page. In this case, the individual page won't shown up in the web search results.
Hide page links from search engines	If selected, add the NOFOLLOW meta tag to every page so that search engines won't follow links (and then index the destination) on any page on the site.	If selected, add the NOFOLLOW meta tag to the current page so that search engines won't follow links (and then index the destination) on the page.
Hide page descriptions from search engines	If selected, add the NOSNIPPET meta tag to every page so that search engines won't include the description (specified above) after the page in the search results.	If selected, add the NOSNIPPET meta tag to the current page so that search engines won't include the page description (specified above) after the page in the search results.

To change search engine optimization (SEO) settings:

1. Open a site for editing.
2. Click  in the sidebar and then click  **SEO**.
3. Provide an optional description for the site. The site description is included on each page in the site.

   This description is also used as the page-level description for the home page only if there's no value set using **Page Settings** for the home page.
4. Optionally, specify keywords separated by commas to help search engines identify the content of the site.

   Site keywords identify terms or concepts that apply to all pages on the site. These values are added (appended) to the keywords specified using **Page Settings** for individual pages.
5. Optionally, add header scripting or tags for analytics or tracking to your site. The header content is included on each page in the site. Validate any code you use in the header to make sure that it works properly and that it does not introduce any security risks to your site.
6. Optionally, add footer scripting or tags for analytics or tracking to your site. The footer content is included on each page in the site. Validate any code you use in the footer to make sure that it works properly and that it does not introduce any security risks to your site.
7. Optionally, select one or more of options to exclude information from appearing in search results, as described in the previous table.
8. To save all pending changes in the current update, click **Save**.

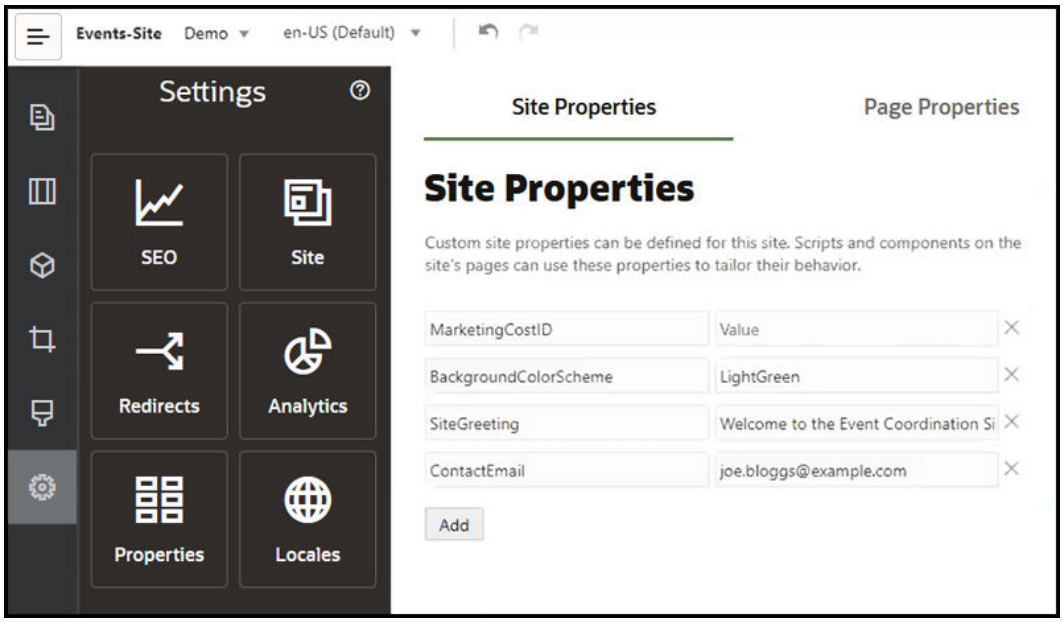
Add Custom Site Properties

You can add custom properties to sites and site pages in the form of name/value pairs. These properties are stored with the site and are made available to scripts and components on the site's pages. These can help to parameterize or customize the site without having to change the underlying scripts and component code.

For example custom properties can be used to change the page background color, refine search results, populate lists, and in general control site-dependent variables.

Custom site properties are added using the settings panel when editing a site.

1. Open a site for editing.
2. Click  in the sidebar and then click  **Site Properties**.
3. Click **Add**.
4. Enter a name and value for the custom site property. You can add up to 50 custom site properties. There is a 200 character limit on the name field and a 2000 character limit to the value field.



5. Click the **X** next to a name/value pair to delete it.
6. When you are finished adding or removing your custom site properties, click **Commit**.

> **Note:**
>
> Changes are not merged with existing custom site properties. Committing changes to custom site properties overwrites any existing custom site properties in the base site.

Once defined, custom site properties can be used in scripting throughout the site and site components, for example in the footer or in the Additional Query String field in components

supporting SCSMacro expansion, such as content list, or through tokens in title and paragraph components.

This scripting works with the SCSRenderAPI during runtime and while designing. Also, the custom site property values are available to layout and component code in the template compiler via the SCSCompileAPI. This API has a new function analogous to the SCSRenderAPI, getCustomSiteProperty, the allows layout and component code to read the value of a custom section property.

For example, let's say you want to define a custom section layout that uses the SCSRenderAPI to call and retrieve custom site properties for the header, contact name and contact email. The following script builds an HTML string using custom site properties that gets appended to the DOM.

```
define([
    'jquery'
], function( $ ){
    'use strict';

    function SectionLayout( params ) {
    }
    SectionLayout.prototype = {
        render: function( parentObj ) {
            var html = '';

            try {
                html += '<div>';
                    html += '<h1>' +
SCSRenderAPI.getCustomSiteProperty('SiteGreeting') + '</h1>;
                    html += '<div>For more information, contact <a
href="mailto:' + SCSRenderAPI.getCustomSiteProperty('ContactEmail')
                        '"> +
SCSRenderAPI.getCustomSiteProperty('ContactName') + '</a></div>';
                html += '</div>';
                $(parentObj).append( html );
            } catch( e ) {
                console.error( e );
            }
        },
    };
```

Custom properties are preserved when creating a template from a site with added custom site properties and when creating a site from a template that has custom site properties.

Add Default Common Page Properties

You can define common page properties along with the site properties at the site level. Common page properties are available on all pages by default. Additional custom page properties specific to a particular page can also be added in the page settings properties tab.

To add default page properties common to all pages:

1. Open a site for editing.

2. Click  in the sidebar and then click  **Properties**.
3. Click the **Page Properties** tab.
4. Click **Add**.
5. Enter a name and value for the page property. You can add up to 50 custom properties. There is a 64 character limit on the name field and a 300 character limit to the value field.
6. Click the **X** next to a name/value pair to delete it.
7. When you are finished adding or removing your page properties, click **Close**.

> **Note:**
>
> Changes are not merged with existing page properties when you commit changes in an update. Committing changes to page properties overwrites any existing page properties in the base site.

Generate RSS Feeds in Site Builder

Oracle Content Management provides a simple and effective way to generate RSS feeds. This gives RSS aggregators the ability to subscribe to press releases, partner announcements, and other pertinent information published on your site dynamically, without having to check for new material.

Using Oracle Content Management Site Builder, you can configure an RSS feed that generates the feed in XML format, previews the generated xml, and then publishes the feed to your site. Using Site Builder, you can define how the feed information is displayed and sorted, edit the feed, delete it, or disable it temporarily. Add as many additional feeds as you need to help site visitors subscribe to only the type of information they want. You can even create your own custom RSS feed template to use.

Configure an RSS Feed

Adding and editing an RSS feed is done through Site Builder using an update.

1. Open a site for editing.
2. Click  in the sidebar and then click  **RSS Feeds**.
3. Click **Add** to create a new feed. If there are already feeds listed, notice that you can edit, preview, or remove them.

> **Note:**
>
> If you inadvertently remove a feed or make a mistake when editing one, you can undo your changes by clicking  . Multiple actions can be undone or redone  .

4. In the Create RSS Feed panel, enter the requested information.

Element	Description
Enable this feed from the site	Toggle this on to enable the RSS feed, off to disable it. It is enabled by default.
Feed Name	The name used for the feed configuration.
Feed Title	The title for the feed when displayed in an RSS aggregator.
Feed URL	The address used by an RSS aggregator to access and load the feed.
Override	Toggles to allow you to change the feed URL entry.
Description	An optional description of the feed configuration.
Content Type	Select the content type that populates the feed, for example, BlogPost or Cartoon. Available content types will depend on the content types used in the Oracle Content Management site repository.
Maximum Items	The maximum number of feed items listed when displayed in an RSS aggregator.
Order By	Select how the items listed in the feed are sorted. Options are Name (A-Z), Name (Z-A), Date (Newest first), Date (Oldest first).
Page to Display Individual Item	Specify the page used to display an item's detailed information in the feed aggregator when an item in the feed list is selected.
RSS Template	Oracle Content Management comes with a default xml template for an RSS feed, but you can create your own as well. RSS templates are created and stored in the developer components section of Oracle Content Management. RSS templates you create are available in this field.
Language	If you translate your site into multiple languages, then you must specify which language to use for your RSS feed. If you want to offer your feed in multiple languages, then you must create multiple feeds.
Time to Live (TTL) in Minutes	The RSS feed time-to-live which controls how long, in minutes, a channel can be cached before refreshing from the source. By default, the RSS feed will be cached on the Akamai CDN that Oracle Content Management provides.

5. Click **Save** in the Create RSS Feed panel to save the feed configuration and close the panel.
6. On the RSS Feeds page you can click **Preview** to validate and view the xml feed results, **Edit** to open the Edit RSS Feed panel to make changes, or **Remove** to delete the feed configuration.
7. Once you have finished configuring and managing your feeds, click **Close**.

You can click **Save** to save your changes to an update at any time. When you are finished configuring your feeds and are ready to merge your changes with the site, click **Commit**.

> **Note:**
>
> Once the feeds are configured and you commit the update, publishing the site also publishes the enabled RSS feeds configured in the site. The RSS template component is also published when the site using it is published, unless it is already published separately.

Create an RSS Template

As a developer, when you create an RSS template component, you are given a fully-functional sample that you can modify for your own site or business requirements.

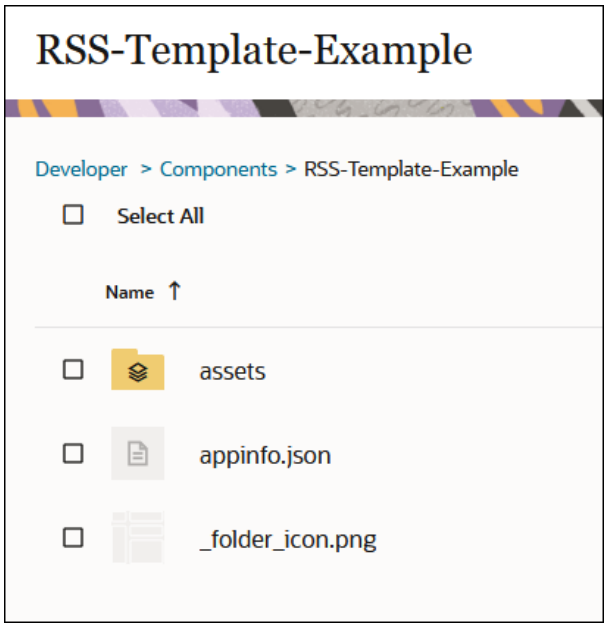
Creating an RSS template component is similar to creating any other local component.

1. Click **Developer** in the left navigation of Oracle Content Management.
2. Click **View All Components**.
3. Select **Create RSS Template** from the **Create** menu on the Components page.
4. Enter a name and optional description and click **Create**.

Editing the RSS Template

The RSS template component is stored in Oracle Content Management and listed on the Components page.

1. Scroll or search to find the RSS template component.
2. Click the RSS template component title to open the component.



3. Click **assets** to open the assets folder.
4. Click **template.xml** to open the xml file.
5. Review the file and click **Edit** to edit the file, or download to edit offliine.

The template.xml file substitutes values into the XML using a Mustache-inspired syntax. There are a few built-in Mustache extensions (lambdas) that Oracle Content Management adds to help you when creating the RSS XML:

Extension	Description
{{#fnLowerCase}}	Transforms the argument into lower case.
{{#fnNativeLink}}	Creates a native link from a CaaS link for use with digital assets.
{{#fnRssDate}}	Converts from a CaaS date value to an RSS date value.

Extension	Description
{{#fnTrim}}	Remove the leading and trailing whitespace from the argument value.
{{#fnUpperCase}}	Transforms the argument into upper case.

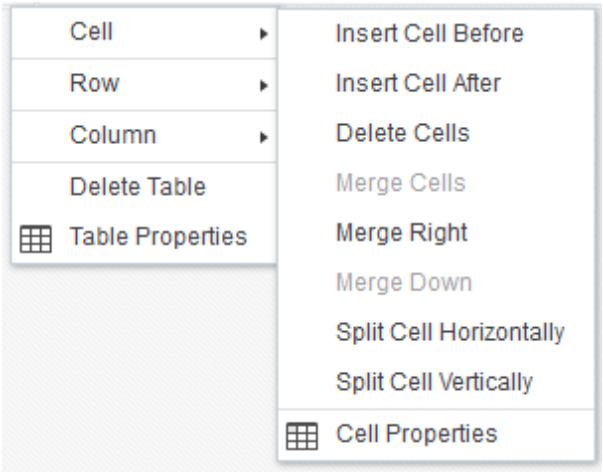
Work with Tables

Within a paragraph component, you can include tables that you create or that you paste from an existing HTML source.

1. Go to the page you want to edit and make sure that  is set to **Edit**.
2. Add the component to the page or click in an existing paragraph component.
3. To add a table from another HTML page, just click and drag to select the table, then copy and paste it into the paragraph component. To create a new table at the current cursor position, click  . Choose the number of rows, column, the width, and other formatting options. You can add an optional **Caption** centered above the table and an optional **Summary**, which gives additional context for assistive technologies such as automated screen readers.
4. Click **OK** when done to close the window. To change these table settings later, right-click in the table and choose **Table Properties**. To delete the table, right-click in the table and choose **Delete Table**.

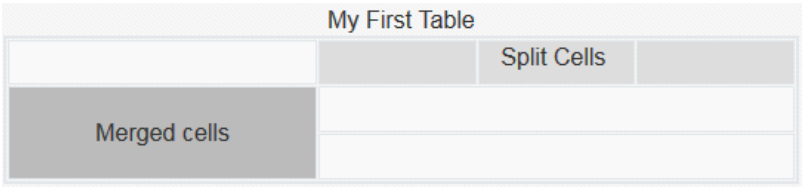
> **Note:**
>
> If developing for differing display sizes such as mobile screens, you can use code found in the StarterTheme design.css file to create a responsive table that enables the stacking of row data when displayed on mobile devices.

5. To add, remove, or modify specific rows, columns, or cells, right-click in the row, column, or cell a choose from the menu of options. For example:



> **Note:**
>
> You can change cell properties for only one cell at a time.

6. You can merge and split cells to create complex table layouts. For example:



   - To split a cell into two cells, right-click in the cell, choose **Cell**, and then choose **Split Cell Horizontally** or **Split Cell Vertically**.
   - To merge two horizontal cells, right-click in the left cell, choose **Cell,** and then choose **Merge Right**.
   - To merge two vertical cells, right-click in the top cell, choose **Cell,** and then choose **Merge Down**.

7. Use the general formatting menu options to changes the format and alignment of text within cells.

Upload Site Files

You can upload images and documents for use with your site at any time using the Oracle Content Management interface. You can also upload files from within Site Builder when working with background images, and with image, gallery, and document components.

When you use image and document files with sites, you can use images stored with the site or in another location you can access. You can also use images that were shared with you or that you upload from a local or network file location.

Upload Files

To upload one or more files from a local or network location:

1. Click **Documents** and go to the location where you want to store the file. Click **Create** to add a new folder in the current location.
2. Click **Upload**.
3. Locate and select one or more files and then click **Open**.

Upload Files within the Editor

When working with background images or with components such as documents or images, you can upload files directly if the file you want to use isn't in an Oracle Content Management location.

For example, to upload one or more files from a local or network location to use with a gallery component:

1. To add images to a gallery, click its menu icon  , choose **Settings**, and click **Images** on the **General** tab.
2. Click **Add Images**.
3. Go to the location where you want to store the file or click **Create** to add a new folder in the current location.
4. Click **Upload**.
5. Locate and select one or more files and then click **Open**.
6. Select one or more images in the repository and click **OK**.

> **Note:**
>
> The window displays all available files. Choose the file type that's appropriate for the context. For example, if you're choosing an image file, select a file with a valid image format (GIF, JPG, JPEG, PNG, or SVG). To link to the file, select **Use a reference to the original file instead of copying the file to the site**. If you don't select this option, a copy of the file is stored with the site and referenced from the site.

6

Use Site Templates and Themes in Sites

A site template has what you need to get started with a site, including the site code framework, a default site with sample pages and content, a theme with styling, resources such as images, and even custom components. A *theme* defines the general look-and-feel—the overall style—of a site, including color scheme, font size, font type, and page backgrounds.

Site Templates

Themes

Understand Site Templates

A site template has everything you need to get started with your site, including the site code framework, a default site with sample pages and content, a theme with styling, resources such as images, and even custom components.

Default Templates

Oracle Content Management provides a number of site templates that you can use to create sites. Just select a template, name the site, and you can start adding content right away. These site templates are typically installed by a service administrator when the service is initialized.

Site Template	Description
Blank Site Template	The Blank template is a single page with header, body, and footer slots, allowing you complete freedom to use your own design.
New Product Launch	The New Product Launch template has a right justified horizontal menu and layouts for case studies, details on features, pricing, your company, and contact information. The home page has a rotating banner image plus text.

Site Template	Description
Products and Services Overview	The Products and Services Overview template has layouts for case studies, details about the product offering, your company, privacy policy and more. The home page has a rotating banner image plus text. The template is fully responsive.
Starter Site Template	Use the Starter template to create your own ready-made site solutions. The starter template provides a simple, yet fully functional example that you can explore and expand with components and interactions. It includes the site code framework, a default site with sample pages and content, a theme with styling, resources such as images, and a custom component with trigger and action functionality. The sample pages include information about creating site templates with links to resources providing more detailed information.
Learn	This template is responsive and features a custom JavaScript menu component within the group. It features custom component groups that offer a number of standard components. It also shows the logged in user.
Relate	This template is responsive and features a custom JavaScript menu component within the group. It features custom component groups that offer a number of standard components.
Share	This template is a site that features a single, long page. It is responsive and features a custom JavaScript menu that navigates to locations on the page rather than to separate pages.
JET Starter Site Template	Oracle JET (JavaScript Extension Toolkit) is a modular, open source toolkit using a collection of open source JavaScript libraries. See Oracle Jet for complete details.

You can also create a template from an existing site, or you can export an existing template, modify it offline, and import it as a new template. Your organization may have site templates for you to use.

How Templates Work

When you create a site, the template is used as follows:

- If site governance is enabled, sites might require approval before they're created. See Understand Site Governance.
- The default site in the template is copied to the new site to provide a starting point for your pages.
- All necessary support files are copied to the new site.
- If the template theme doesn't exist in the themes folder, the theme is copied to the themes folder. The site references the theme from its location in the themes folder. If the theme exists, the new site simply references the existing theme.
- If there are custom components that don't exist in the components folder, they're copied to the components folder. The site references the components from their locations in the components folder. If any of the components exist, the new site references the existing components.

Create Custom Site templates

If you're a web developer, a template collects all the pieces you need to construct a website in one package including the site, layout, navigation, sample content, and so on. You can add components and interactions to the site to provide ready-made site solutions that meet your business needs.

A site template is represented by a folder structure that you can work with like other folders. Some elements of the site template, such as the theme and custom components, are referenced from their associated locations in Oracle Content Management. For example, a site template references the associated theme from the list of available themes like a site references a theme.

If you create a site template from an existing site, the new site template uses a copy of the site as its default site. The site template references the theme used by the site and any custom components used in the site pages. The theme and custom components aren't copied to the site template, but are referenced in the same way they are by the site.

> **Note:**
>
> The site template reflects the site used at the time the site template is created. Further changes to the site that was used to create the site template aren't reflected in the site stored with the site template.

You can create content site templates which can be used to share content models, which includes content layouts, content items, and digital assets that are needed to support a content model (such as sample content). Content site templates should be created from sites with published content items and digital assets. Note that content types are not created when a site template is imported from a package with content. They are created when a site is created from the content template. Therefore, content types are not owned by the user who imports the site template. Instead, content types are owned by the user who creates a first site from that site template. That user can then share the site template with other users as needed. This feature may not be available depending on the Content Management Cloud subscription type and start date of your service.

When you export a site template, all elements of the site template, including a copy of the theme and any components, are collected in a site template package that you can download and work with offline.

If you import a site template you modified offline, and if the site template, theme, or custom component names or IDs already exist, you're prompted to resolve the conflicts. You may be given the option to create a new site template, theme, or custom component, or in some cases, you can overwrite the existing site template, theme or custom component with the imported version. See Develop Site Templates.

You can also import a site template to a specific repository. When you do, you are given the choice to update or duplicate existing assets. See Import Templates into a Specific Repository.

Share Site Templates

When you create a site template, whether by importing, copying, or creating from a site, the site template can't be used by anyone else until you share it.

> **Note:**
>
> This is also true for the site templates provided with Oracle Content Management and installed by an administrator. If you don't see any site templates, contact your administrator. They may not have been shared with you.

When you share a site template with a user for the first time, the associated theme and any associated custom components are automatically shared with the user and given the downloader role to ensure that they're available if the user creates a site from the site template. Subsequent changes in the site template to the role for that user do not update the sharing information for the associated theme or custom components.

If site governance is enabled, you make site templates available through site template policies. See Understand Site Governance and Change Site Template Policies.

Create a Site Template from a Site

If you have a site that you want to use as a starting point for other sites, you can create a site template from that site.

> **Note:**
>
> To create a site template from a site, you must have the Downloader, Contributor, or Manager role for the site.
> In addition, your administrator must enable the options in the **Create** menu. If you don't see the **Create** menu on the site templates page, contact your administrator.

These steps show how to create a site template from a site. You can also import a site template package that you created or modified offline. See Export and Import Site Templates.

> **Note:**
>
> If you are creating a site template from a site that uses content from multiple repositories, you need to do so using the Oracle Content Management Toolkit. See Develop with Oracle Content Management Toolkit and Use the cec Command-Line Utility.

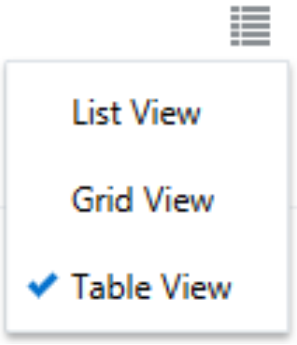
To create a site template from a site using the Oracle Content Management web interface:

1. Click **Sites** and select the site you want to use.
2. Choose **Create Site Template** in the right-click menu or click  in the actions bar.
3. Enter a name for the site template. You can use letters, numbers, underscores (_), and hyphens (-). If you enter a space, it's automatically replaced with a hyphen.

   Don't use the following names for site templates, themes, components, sites, or site pages: authsite, content, pages, scstemplate_*, _comps, _components, _compsdelivery, _idcservice , _sitescloud, _sitesclouddelivery, _themes, _themesdelivery. Although you can use the following names for site pages, don't use them for site templates, themes, components, or sites: documents, sites.
4. Enter an optional description for the site template.

5. Choose whether you'll include unpublished content items and digital assets in the site template.
6. When you're ready, click **Create Site Template**.

When the site template is created, the name appears in the list of site templates. To see all site templates, click **Developer** and then **View All Templates**. You can control how site templates are displayed by clicking the view icon and selecting an option from the list.



To view the folders and files associated with a site template, choose **Open** from the right-click menu or click in the actions bar. To view or change the name, description, and other details about the site template, such as requiring a site security taxonomy, click the site template name or choose **Details** in the right-click menu or click in the actions bar.

The new site template uses a copy of the site as its default site. The site template references the theme used by the site and any custom components used in the site pages. The theme and custom components are not copied to the site template, but are referenced in the same way they are by the site.

If the site template was created from an enterprise site, the localization policy and default language used by the original site will be selected by default for any new sites created from the site template.

The site template reflects the site used to create it at the time the site template is created. Any future changes to the site used to create the site template are not reflected in the site stored with the site template.

Change Site Template Details

The site template details tab shows the site template name, author, description, site template theme, custom components included in the site template, whether or not the site template requires a site security taxonomy, and the site template preview images. If site governance is enabled, you also see the status of the site template (whether the site template is available for use when creating sites).

If you have the appropriate permissions, you can change or update site template properties such as the name and description. You can also add or remove preview images of the site template.

You can update site template details if you created the site template (you're the site template owner), you're a site administrator, or if someone shared a site template with you and gave you a Contributor or Manager role.

If you add preview images, the files are stored in the site template's asset folder. These files don't appear in any associated website, but allow the site template owner to provide information about the site template itself.

To view or change site template details:

1. Click **Developer** and then click **View All Templates**.
2. Select the site template, and choose **Details** in the right-click menu or click  in the actions bar.
3. Edit the information as necessary. If you can't edit the information, you don't have the Contributor or Manager role.
4. To add a preview image, click . Find the image you want to use, or click **Upload** to upload an image stored locally. The image should be at least 180px by 135px (aspect ratio of 4:3). Smaller images might not display optimally and images much larger might affect performance.

   If all the preview image spots are filled, you'll need to delete an image before you can add a new one. To delete a preview image, click  on the image.
5. Click **Save**.

Change Site Template Policies

The site site template policies tab shows the type of site site template (standard or enterprise), the approval required, and the minimum security for sites created from the site site template. You also see the status of the site site template (whether the site site template is available for use when creating sites).

You see the site site template policies tab only if site governance is enabled and you are a site administrator. See Understand Site Governance.

To view or change site site template policies:

1. Click **Developer** and then click **View All Templates**.
2. Select the site site template, and choose **Details** in the right-click menu or click  in the actions bar.
3. Select the **Policies** tab.
4. Select whether site requests created from this site site template require approval from the site administrator, if they are automatically approved, or if they require approval from specific people. If you want to limit approval to specific people, start entering the name or email of the person or group you want to add as an approver, then select the person or group from the search results. To remove an approver, click the **X** next to their name.
5. Select the minimum security required for sites created from this site site template. Site creators can select a higher level of security for their site if desired.
   - **Specific service users** - Only selected users that can sign into this instance of Oracle Content Management can access the site. The site creator selects the Oracle Content Management users after the site is created. See Change Site Security.

- **Specific cloud users** - Only selected users that can sign into your domain can access the site. The site creator selects the cloud users after the site is created. See Change Site Security.
- **Service users** - Any user that can sign into this instance of Oracle Content Management can access the site.
- **Cloud users** - Any user that can sign into your domain can access the site.
- **Everyone** - Anyone can access the site without signing in.

6. Select an expiration policy to determine when a site using the site template expires. Site administrators can specify whether expired sites are taken offline or deleted. Site owners and managers are notified by email prior to site expiration and are given the opportunity to extend the expiration time. If they do not extend the expiration, they are notified by email when the site has either been taken offline or deleted. At that time they can extend the expiration and bring the site back online or restore it from trash if necessary. Expiration policy options are:
   - **Never**
   - **1 year**
   - **2 years**
   - **Custom**
     Selecting **Custom** allows you to set an expiration shorter than 1 year or longer than 2 years.

     > **Note:**
     >
     > After a site is created, site administrators can change a site's expiration policy in the **Site Properties** dialog.

7. If you're editing an enterprise site template, you can select how to create the site prefix for friendly URL values. This prefix will be appended to content item slug values (the part of the URL specific to the page or asset). You can have the prefix automatically generated based on the site name, or you can let the user enter a prefix.
8. If you're editing an enterprise site template, choose the localization policy used to determine which languages are required for the site.
9. When you're done, click **Save**.

Change Site Template Status or Audience

The site template audience tab shows the status of the site template (whether the site template is available for use when creating sites) and who can use the site template to create sites.

You see the site template audience tab only if site governance is enabled and you are a site administrator. See Understand Site Governance.

To view or change site template audience:

1. Click **Developer** and then click **View All Templates**.
2. Select the site template, and choose **Details** in the right-click menu or click  in the actions bar.

3. Select the **Audience** tab.
4. To change the site template status, click the status switch. You must make the site template active for users to create sites from the site template. By default, when you make activate a site template, it's available to all site creators. If you want to change who can use the site template, change the setting on this tab.
5. Select who can use this site template to create sites.
   - **Specific people**—Only selected people can use this site template to create sites. Start entering the name or email of the person or group you want to be able to use this site template, then select the person or group from the search results. To remove a person or group, click the **X** next to their name.
   - **Everyone**—Anyone who can create sites can use this site template.
6. When you're done, click **Save**.

Manage Site Templates

You can copy, rename, and delete a site template folder like you would any other folder, but a site template has special considerations when you import or share them.

Open the site site template page by selecting **Developer** and then **View All Site Templates**. Use options on the actions bar or right-click menu on the Templates page to perform any of these tasks.

Task	Description
Create a site site template	If you have a site that you want to use as a starting point for other sites, you can create a site site template from that site. **Note:** You must have the Downloader, Contributor, or Manager role for the site, and your administrator must enable the option. If you don't see the **Create** menu on the site templates page, contact your administrator. Also, if you are creating a site site template from a site that uses content from multiple repositories, you need to do so using the Oracle Content Management Toolkit. See Develop with Oracle Content Management Toolkit and Use the cec Command-Line Utility. On the site templates page, click **Create**, then select **Create from existing site**. Select a site, name your site template and click **Create**. See Create a Site Template from a Site.

Task	Description
View and edit site template files	You can explore the folders and files that make up the site site template by selecting the site site template and choosing **Open** in the right-click menu or click  in the actions bar. If you're a web developer, you can download and modify individual site site template files or you can use the desktop app and synchronize the changes you make on your local system. You can also export the site site template package, which includes the associated theme and any custom components, and work with it offline in your preferred development environment. A site site template is represented by a folder structure that you can work with like other folders. When you export a site site template, all elements of the site site template, including a copy of the theme and any components, are collected in a site site template package that you can download and work with offline. See Develop Site Templates.
Create a site	To create a site from a site site template, select the site site template and choose **Create Site** in the right-click menu or click  in the actions bar. See Create Sites.
Rename a site site template	Select the site site template you want to rename and choose **Rename** from the right-click menu or click  in the actions bar. Enter a name for the site site template that's different from any other site site template on the same server. For naming guidelines, see Create a Site Template from a Site.
Copy a site site template	You can create a site site template by copying an existing site site template and making changes to the copy. **Note:** When you copy a site site template, sharing information associated with the site site template isn't copied. Select the site site template you want to copy and choose **Copy** from the right-click menu or click  in the actions bar. Enter a name for the site site template that's different from any other site site template on the same server. For other naming guidelines, see Create a Site Template from a Site A progress bar shows the new site site template name and copy status.

Task	Description
Delete or restore a site site template	If you have the appropriate permissions, you can delete a site site template folder and its contents. When you delete a site site template, the site site template folder and all its associated folders and files are moved to the trash. You can delete or restore a site site template if you created the site site template (you're the site site template owner) or if someone has shared a site site template with you and has given you the manager role. **Note:** When you delete a site site template, the associated theme and custom components are not deleted. To delete a site site template, select the site site template you want to delete and choose **Delete** from the right-click menu or click  in the actions bar. You are prompted to move the site site template to the trash. A deleted site site template stays in the trash until: • You restore the site site template. • You permanently delete the site site template. • Your trash quota is reached. • The trash is automatically emptied based on the interval set by your service administrator. The default value is 90 days. To restore a site site template, select **Trash** in the Templates drop-down list, then select the site site template and choose **Restore** from the menu bar or the right-click menu.
Export or import a site site template	You can export a site template to modify it offline and then import it either as a new site template or to replace the existing site template. You can also export a site template to move it to another instance and import it there. When you export a site template, you copy the site template to a folder as a single .zip file. You can download the site template package directly from the folder to unpack and work with the individual files. When you're done working with the files, create a .zip file containing the site template package, import it, and overwrite the original site template or create a new one. See Export and Import Site Templates.

Task	Description
Share a site template and manage member roles	If you're the owner or a manager of a site template, you can share a site template with other users and assign a role that defines what the user can do with the site template. If governance *isn't* enabled, assigning any role to a user gives them permission to create a site from the site template. If governance *is* enabled, you must activate the site template and make sure the user is included in the site template audience to give them permission to create a site from the site template. See Change Site Template Status or Audience. The site template's theme and custom components are stored as separate objects, and are just referenced in the site template. Therefore, theme and custom component membership is managed separately. When you share a site template with a user, no matter what role you give them for the site template, they're automatically given the Downloader role for any associated theme and custom components to ensure that those objects are available to the user if they create a site from the site template. Select the site template you want to share, choose **Members** in the right-click menu or actions bar. Enter one or more user names or email addresses, and assign one of these roles: • **Viewer**: Viewers can view the folders and files of the site template, but can't change things. • **Downloader**: Downloaders can also download files and save them to their own computer. • **Contributor**: Contributors can also edit site template details and site template files, upload new files, delete files, and delete the site template itself. • **Manager**: Managers can also add users and assign their roles. The owner (creator) of a site template is automatically assigned the manager role.
View or edit site template details	To view site template details, select the site template, and then choose **Details** in the right-click menu or click  in the actions bar. The site template details include the site template name, author, support files, description, theme, custom components, and the thumbnails. If you have the appropriate site template role, you can edit the details. See Change Site Template Details.
View or edit site template policies (with governance enabled)	If governance is enabled and you're a site administrator, you can view site template policies, such as required approval, minimum security, site expiration, and localization. Select the site template, choose **Details** in the right-click menu or click  in the actions bar, then select the **Policies** tab. See Change Site Template Policies.
Make a site template available for site creation and manage who can use the site template to create sites (with governance enabled)	If governance is enabled and you're a site administrator, you can make the site template available for site creation and manage who can use the site template to create sites. Templates that are active (available for site creation) show  next to them. To change the status or audience, select the site template, and choose **Details** in the right-click menu or click  in the actions bar. • To make the site template available for site creation, click the status switch to activate it. • To change who can use the site template to create sites, click the **Audience** tab. See Change Site Template Status or Audience.

Export and Import Site Templates

You can export a site site template to modify it offline and then import it either as a new site site template or to replace the existing site template. You can also export a site template to move it to another Oracle Content Management instance and import it there

Export Site Templates

When you export a site template, you essentially copy the site template to a folder in Oracle Content Management as a single .zip file. You can download the site template package directly from the folder to unpack and work with the individual files. When you're done working with the site template files, create a .zip file containing the site template package and import it into your site and overwrite the original site template or create a new one.

> **Note:**
>
> When you export a site template, sharing information for the site template isn't included.

To export a site template:

1. Click **Developer** and click **View All Site Templates**.
2. Select a site template and choose **Export** from the menu bar or the right-click menu.
3. Navigate to an existing folder or create a new folder by clicking **Create**, and providing a name and an optional description.
4. Choose a folder by selecting its checkbox and clicking **OK**.

Import Site Templates

> **Note:**
>
> If you import a site template to a different server, some links in the default site may not be valid in the new server context. If the site uses reference links to images or other content rather than copying the content directly into the site, that content is not available on the new server. Even if you copy the content to the new server, the content will have a different internal ID, and the link will not be valid. When you import the site template, you are notified of the pages that contain invalid reference links.

To import a site template package:

1. Click **Developer** and click **View All Site Templates**.
2. Click **Create** and choose **Import a site template package**.
3. If you've uploaded a site template package, navigate to the folder that contains the site template package and open the folder. If you haven't uploaded the site

template package yet, go to the folder where you want to upload it or create a new folder. Click **Upload** and then locate and select the site template package and click **Open**.

4. To use a site template, click the checkbox next to the site template file name and click **OK**. New folders are created for the site template, its associated theme, and any custom components. If the site template, theme, or custom component names or IDs exist, you're prompted to resolve the conflicts. You may need to create a new site template, theme, or component, or you can overwrite the existing items with the imported version.

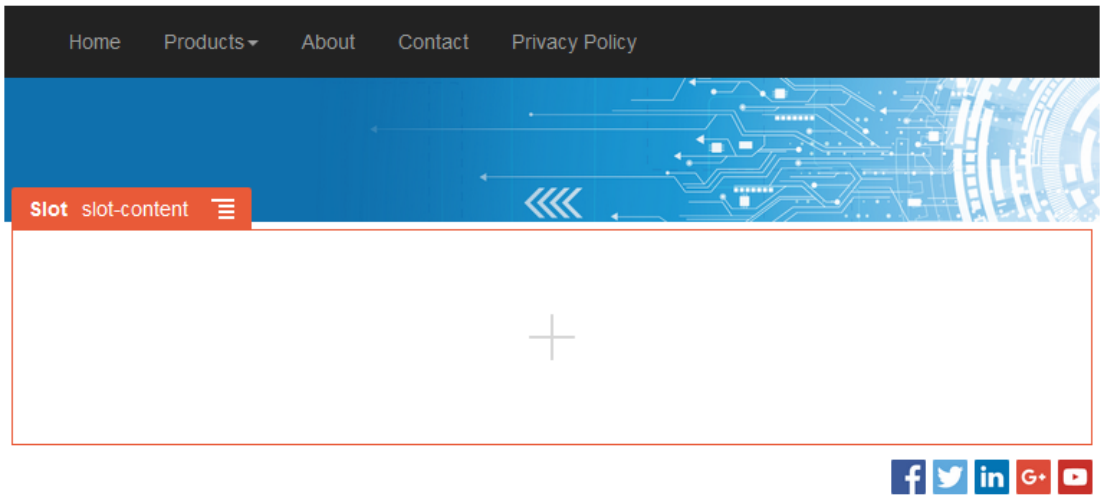
Import Site Templates into a Specific Repository

You can select a repository and import a site template package specifically to it.

1. Click **Content** and select **Repositories** from the menu.
2. Select the repository where you want to import the site template package and click **Import Content**.
3. If you've uploaded a site template package, navigate to the folder that contains the site template package and open the folder. If you haven't uploaded the site template package yet, go to the folder where you want to upload it or create a new folder. Click **Upload** and then locate and select the site template package and click **OK**.
4. Choose if you want assets that already exist in the repository to be updated with the new revisions, or if you want duplicates of the assets made and click **Import**.
5. Click **Details** to see a list of assets and content types that were imported. Click **OK** when done.

Understand Themes

A *theme* defines the general look-and-feel—the overall style—of a site, including color scheme, font size, font type, and page backgrounds. Themes provide visual consistency between the pages in the site. You adjust the design and add content to create a site that sells your style, your brand, and your vision.



A theme includes:

- Page layouts
- Cascading style sheet (CSS)

- Variations on the style sheet
- Configuration files
- Background code that defines site navigation

Each website uses a theme. When you create the site from a site template, you inherit the theme from the site template. You can change the theme for a site at any time.

Oracle Content Management provides a number of site templates with themes that you can use to get started. To create a new theme, copy an existing theme. You can download and modify the theme files, or you can use the desktop app and synchronize the changes you make on your local system. For information about other ways to create themes, see Develop Themes.

Developers can also create a theme which uses a specific subset of components. When creating a site using one of these themes, you'll be shown only those components that are allowed with that theme. In this way, you don't have to worry about whether or not a component is valid for the theme you've chosen. You can still use any seeded components along with the themed components.

If a site uses a new, unpublished theme, the theme is automatically published with the site when you place the site online for the first time. If you make changes to a theme and want to update online sites to show the changes, you must explicitly publish the theme. Only the theme owner or a user with Manager privileges can explicitly publish a theme.

> **Note:**
>
> If you publish changes to a theme, all online sites that use the theme reflect the change. For example, if you change the default font in the theme and publish the theme, all sites using the theme will use the new default font.

Manage Themes

You can copy, rename, and delete a theme folder like you would any other folder, but a theme does have special considerations because they can be shared by more than one site.

You can see all themes by clicking **Developer** and then clicking **View All Themes**. You use options on the menu bar or the right-click menu on the Themes page to perform these tasks.

Task	Description
Create a new theme	To create a new theme, copy an existing theme. For information about other ways of creating themes, see Develop Themes.

Task	Description
Copy a theme	You can create a new theme by copying an existing theme and making changes to the copy. **Note:** When you copy a theme, sharing information associated with the theme isn't copied. Select the theme you want to copy and choose **Copy** from the menu bar or the right-click menu. Enter a name that isn't used by another theme. You can use letters, numbers, underscores (_), and hyphens (-). If you enter a space, it's automatically replaced with a hyphen. Click **Copy**. Don't use the following names for site templates, themes, components, sites, or site pages: authsite, content, pages, scstemplate_*, _comps, _components, _compsdelivery, _idcservice , _sitescloud, _sitesclouddelivery, _themes, _themesdelivery. Although you can use the following names for site pages, don't use them for site templates, themes, components, or sites: documents, sites.
Share a theme	If your administrator enabled sharing and you're the owner or a manager of the theme, you can share your theme with other users. When you share a theme, you assign a role defining what the user can do with the theme. **Note:** When someone shares a site template with a user, no matter what role they're given for the site template, the user is automatically given the Downloader role for any associated theme to ensure that the theme is available to the user if they create a site from the site template. Right-click the theme you want to share, choose **Share**, and click **Add Members**. Enter one or more user names or email addresses, and assign one of these roles: • **Viewer**: Viewers can view the folders and files of the theme, but can't change things. • **Downloader**: Downloaders can also download files and save them to their own computer. • **Contributor**: Contributors can also edit the theme, upload new files, delete files, and delete the theme itself. • **Manager**: Managers can also add users and assign their roles. The creator of a theme (the owner) is automatically assigned the manager role.

Task	Description
Edit a theme	If you're a web developer, you can download and modify individual theme files or you can use the desktop app and synchronize any changes you make on your local system. You can also export a site template package, which includes the associated theme and any custom components, and work with it offline in your preferred development environment. See Develop Themes. If you make changes to a theme, you must publish the theme for those changes to be reflected by the sites using the theme.
Publish a theme	If a site uses a new, unpublished theme, the theme is automatically published with the site when you place the site online for the first time. If you make changes to a theme and want to update online sites to show the changes, you must explicitly publish the theme. You can publish a theme if you're the owner or if you're assigned the Manager role. Select the theme in the list and click **Publish** (or **Republish** for previously published themes) in the menu bar. A published icon is added next to the theme in the list. See Publish Themes.
Delete or restore a theme	If you have the appropriate permissions, you can delete a theme folder and its contents. When you delete a theme, the theme folder and all its associated folders and files are moved to the trash. You can delete or restore a theme if you created the theme (you're the theme owner) or if someone shared a theme with you and gave you the manager role. **Note:** You can't delete a theme if it's used by any site. To delete a theme, right-click the theme you want to delete and choose **Delete**. You're prompted to move the theme to the trash. A deleted theme stays in the trash until: • You restore the theme. • You permanently delete the theme. • Your trash quota is reached. • The trash is automatically emptied based on the interval set by your service administrator. The default value is 90 days. To restore a theme, click **Trash** in the Theme page menu bar, then right-click the theme from the list and choose **Restore**.

Publish Themes

A theme defines the general look-and-feel of a site. You can update a theme to change the appearance of any sites using the theme.

If a site uses a new, unpublished theme, the theme is automatically published with the site when you place the site online for the first time. If you make changes to a theme and want to update online sites to show the changes, you must explicitly publish the theme. To publish changes to a theme, you must be the theme owner or have the Manager role.

> **Note:**
>
> If you publish changes to a theme, all online sites using the theme reflect the change. Make sure you tested your changes offline and that you understand the impact on the associated sites before you publish updates to a theme.

To publish a theme:

1. Click **Developer** and then **View All Themes**.
2. Select an existing theme from the list of themes.
3. Click **Publish** from the menu bar or the right-click menu.
4. Click **Confirm to proceed** and then click **OK**. A published icon is added next to the theme in the list.

7

Manage Custom Components and Layouts

As a developer, you can create and manage custom components and layouts. As a site contributor, you can register third-party components (apps) and component groups.

Custom components include component groups you create in the editor, and local and remote components you create using options described in this section. Custom layouts include *section layouts* for arranging components in a slot on a page and *content layouts* for arranging the fields in a content item.

- Understand Custom Components
- Understand Layouts
- Use Custom Components and Layouts
- Register Remote Components
- Create Local Components, Layouts, Content Field Editors, or Content Forms
- Export or Import Components or Layouts

To learn how to use components with your site, see Arrange Page Content.

For information about using individual components, see Use Built-In Components.

For details about how to create your own components, see Develop Components.

Understand Custom Components

Components are the individual parts of a web page. When you look at a web page, what do you see? You probably see a few titles, some paragraphs of text, and several links to other pages in the site. You might also see images, buttons, dividers, maps, and galleries. Each of these items is a component.

To add a component to a page, make sure that  is set to **Edit**, click  and choose which type of component you want to use.

Drag the component from the panel and drop it into a slot on the page. That's it. Drag and drop titles, paragraphs, images, and other components where you want them on a page.

You can easily register and incorporate remote components (apps) and even create your own components using options in the component manager. Click **Developer** and then click **View All Components**.

Click **Create** and select the associated option to create a new local component or register a remote component. Components you create and share in this way are listed in the custom components panel in the editor.

For details about how to create your own components, see Develop Components.

Remote Components

If you have a third-party component (app) that you want to use, just register it and use it in your site. It's that easy.

When you register a remote component, you specify the URL for the remote component itself and a second URL for any settings that a web author can specify for the remote component.

After you register a remote component, you can share it with other users. Any registered remote components that you own or that are shared with you are listed in the Custom Components panel in Site Builder.

You can change the properties for a remote component within the editor the same way you do for any other component. Just click the menu icon and choose **Settings**. The **Custom** button opens the settings URL you specified when you registered the remote component.

> **Note:**
>
> Because remote components are hosted on a server other than the one hosting Oracle Content Management, they're enclosed in an inline frame (using an iframe element) for security purposes. Not all remote components can be enclosed in an inline frame. Check with the provider to find out if it can be enclosed.

Local Components

Developers can create components with access to the same features and capabilities as those provided by Oracle Content Management. You can insert the component directly into the page or enclose it in an inline frame with the iframe element.

When you click **Create** to create a local component, a fully-functional sample component is added to the list of components with a name you specify and a unique identifier. As a developer, you can modify the sample to create your own solutions.

For details about how to create your own components, see Develop Components.

Understand Layouts

A page layout arranges slots and content on a page. A section layout arranges content within a slot. A content layout arranges the fields in a content item.

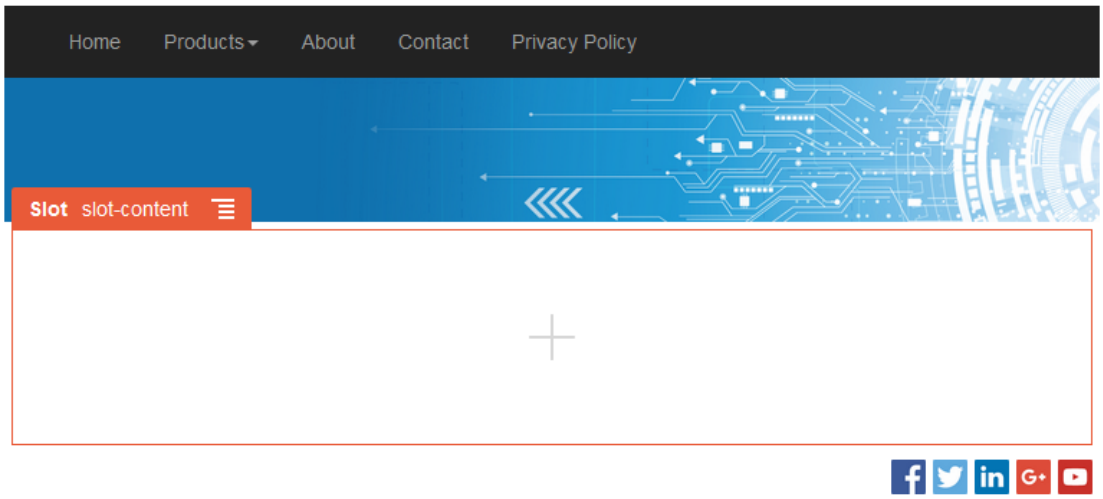
In general, a layout specifies the presentation of content, but not the content itself. Separating the content from its presentation makes it easier to present the same content in different ways or to change the presentation without having to touch the content.

Page Layout

When you add a page to a site, you select a layout to use for that page. Each layout has areas on the page, called slots, where a contributor can drag and drop content. A page layout defines the number and position of slots on the page. A layout can also include content that is predefined and positioned on the page. This content can be

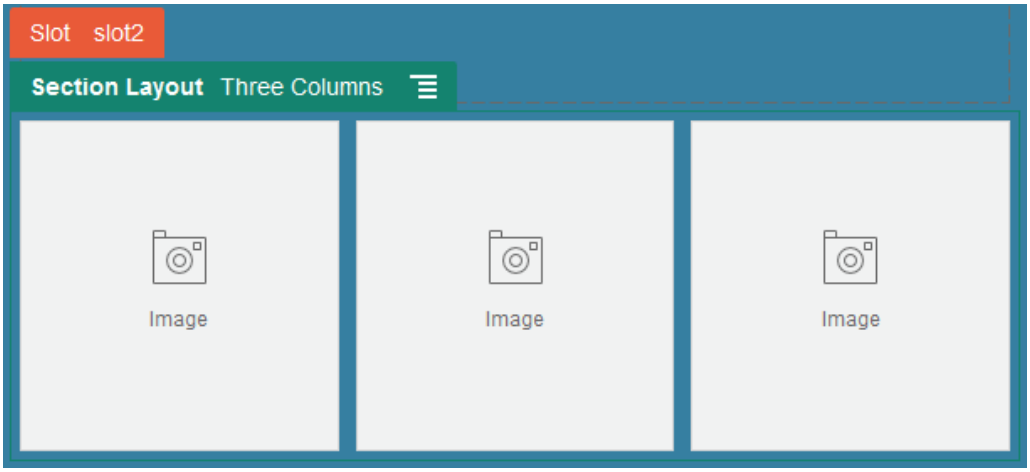
static and not editable, such as a company logo, or it can be minimally editable, such as heading text that a contributor can change, but the position or appearance of which they can't.

Page layouts are stored in the theme. Themes can have one or more page layouts. As a developer, you can copy and modify an existing theme to create a new theme. See Develop Layouts.



Section Layout

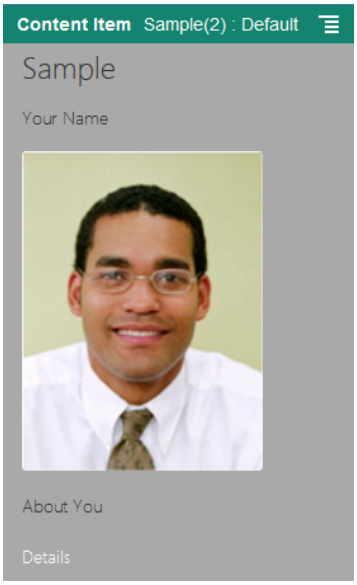
A section layout automatically organizes content added to it, making it easy for a contributor to add content without spending time formatting it on the page. For example, a section layout can automatically organize content into multiple columns or into a vertical list. A site contributor can add one or more section layouts to a slot to organize content.



In addition to the layouts provided, a developer can create additional section layouts to solve particular layout problems or to simplify authoring for contributors. A theme designer can even build section layouts into a slot in a page layout. See Develop Layouts.

Content Layout

If you are an enterprise user, you can create and use content items based on content types and layouts provided for you. Content structured in this way makes it possible for you as a contributor to assemble the content for a content item outside of Site Builder. Multiple content layouts associated with the content type make it possible for the site designer to display the content item in different contexts without having to touch the assembled content. As a developer, you can create new layouts from the default layout provided. See Develop Layouts.



Use Custom Components and Layouts

As a developer, you can create and manage custom components and layouts. As a site contributor, you can register third-party components (apps) and component groups.

Custom components include component groups you create in the editor, and local and remote components you create using options described below. If you have a third-party app (remote component) that you want to use, just register it and use it in your site. Developers can also create local components with access to the same features and capabilities as those provided by Oracle Content Management.

Custom layouts include *section layouts* for arranging components in a slot on a page and *content layouts* for arranging the fields in a content item.

Use options on the menu bar or right-click menu on the Components page to perform the following tasks.

Task	Description
Register a remote component	When you register a remote component, you specify the URL for the remote component itself and a second URL for any settings that a web author can specify for the remote component. Before you can create a component, your administrator must enable the options in the **Create** menu. If you don't see the **Create** menu on the components page, contact your administrator. Click **Create** and select the associated option to register a remote component. See Register Remote Components.
Create a local component or layout	When you create a local component or layout, you are given a fully-functional sample component or layout that you use as a basis for creating your own component or layout. Before you can create a component or layout, your administrator must enable the options in the **Create** menu. If you don't see the **Create** menu on the components page, contact your administrator. Click **Create** and select the associated option to create a new local component or layout. See Create Local Components, Layouts, Content Field Editors, or Content Forms. For information about other ways to create components, see Develop Components.
Copy a component or layout	You can create a new component or layout by copying an existing component or layout and making changes to the copy. **Note:** You can't change the name of a component or layout after you create or register it. You can copy a component or layout and specify a different name for the copy. All the other registration information, including the key value for remote components, are preserved. Sharing information is independent from registration information and isn't copied. Right-click the component or layout you want to copy and choose **Copy**. Enter a name and click **Copy**. You can use letters, numbers, underscores (_), and hyphens (-). If you enter a space, it's automatically replaced with a hyphen.
Create Content Field Editor	You can create a content field editor that can be promoted and used when creating a content type to control the appearance of the data field. All data types are supported except *Media* and *Reference*. The component must be promoted before it is available for use when creating a content type.

Task	Description
Share a component or layout	You can share your component or layout with other Oracle Content Management users. You can share a component or layout if you're the owner or if you're assigned the Manager role. When you share a component or layout, you must assign a role to the user defining what the user can do with the component or layout. Right-click the component or layout you want to share, choose **Share**, and click **Add Members**. Enter one or more user names or email addresses, and assign one of these roles: • **Viewer**: Viewers can see and use the component or layout in the list of custom components and layouts in the editor. They can also view the folders and files of the theme, but can't change things. • **Downloader**: Downloaders can also download files and save them to their own computer. • **Contributor**: Contributors can also edit the component or layout, upload new files, delete files, and delete the component or layout itself, provided it is not in use in a site. • **Manager**: Managers can also add users and assign their roles. The creator of a component or layout (the owner) is automatically assigned the manager role.
Edit a component or layout	If you're a web developer, you can download and modify individual component or layout files or you can use the desktop app and synchronize any changes you make on your local system. You can also export a component or layout individually or as part of a template package, which includes the any custom components and section layouts, and work with it offline in your preferred development environment. If you make changes to a component or layout, you must publish the component or layout for those changes to be reflected by the sites that use the component or layout.
Export or import a component or layout	You can export a component or layout to modify it offline and then import it either as a new component or layout or to replace the existing component or layout. You can also export a component or layout to move it to another instance and import it there. You can export a component or layout individually or as part of a template package, which includes any custom components and layouts. When you export a component or layout, you copy the component or layout to a folder as a single .zip file. You can download the component or layout package directly from the folder to unpack and work with the individual files. When you're done working with the component or layout files, create a .zip file that contains the component or layout package, import it, and overwrite the original component or layout or create a new one. See Export or Import Components or Layouts.
Publish a component or layout	If a site uses a new, unpublished component or layout, the component or layout is automatically published with the site when you place the site online for the first time. If you make changes to a component or layout and want to update online sites to show the changes, you must explicitly publish the component or layout. You can publish a component or layout if you're the owner or if you're assigned the Manager role. Select the component or layout in the list and click **Publish** (or **Republish** for previously published components or layouts) in the menu bar. A published icon  is displayed next to the component or layout in the list.

Task	Description
Delete or restore a component or layout	If you have the appropriate permissions, you can delete a component or layout folder and its contents. When you delete a component or layout, the component or layout folder and all its associated folders and files are moved to the trash. You can delete or restore a component or layout if you created the component or layout (you're the component or layout owner) or if someone shared a component or layout with you and gave you the manager role. **Note:** You can't delete a component or layout if it's used by any site or update, including sites or updates in the trash. To delete a component or layout, right-click the component or layout you want to delete and choose **Delete**. You're prompted to move the component or layout to the trash. A deleted component or layout stays in the trash until: • You restore the component or layout. • You permanently delete the component or layout. • Your trash quota is reached. • The trash is automatically emptied based on the interval set by your service administrator. The default value is 90 days. To restore a component or layout, click **Trash**, then right-click the component or layout from the list and choose **Restore**.

Register Remote Components

To use a remote component in a site, you must first register it in Oracle Content Management.

You can register third-party remote components and remote components that you developed.

Note:

Before you can register a remote component, your administrator must enable the options in the **Create** menu. If you don't see the **Create** menu on the components page, contact your administrator.

To register a remote component for use in Oracle Content Management:

1. Click **Developer** and then click **View All Components**.
2. Click **Create** and choose **Register Remote Component**.
3. In the Register Remote Component window, enter or select information including:
   - **Name**: Name of the component that users will see.
   - **Description**: Description of the component that users will see.
   - **Component URL**: The end point used in an iframe to render component content in a page. It must be HTTPS.

- **Settings URL**: The end point used in an iframe to render the settings of a remote component added to a page. It must be HTTPS.
- **Settings Width**: Sets the default width of the component settings panel in pixels.
- **Settings Height**: Sets the default height of the component settings panel in pixels.
- **Key**: A 192–bit AES key associated with the remote component and used to create a signed hash token when the component is provisioned. It's used to encrypt and ensure component settings are read and written securely.

4. Click **Register**.

   When the remote component is created, the name appears in the list of components. You can explore the files used to register the component by clicking the component name in the list of components.

The component registration information is stored in the catalog used by sites created in the same Oracle Content Management instance, but the component remains a remote service.

As the component owner, component icon is added to the Custom Components panel in the editor with the name you assigned to the component. You can share the component with other users and they will see the component in the Custom Components panel in the editor.

Create Local Components, Layouts, Content Field Editors, or Content Forms

As a developer, when you create a local component or layout, you are given a fully-functional sample that you can modify for your own site or business requirements.

Before you can create a component or layout, your administrator must enable the options in the **Create** menu. If you don't see the **Create** menu on the components page, contact your administrator.

As a developer, you can create components with access to the same features and capabilities as those provided by Oracle Content Management.

To create a sample local component or layout:

1. Click **Developer** and then click **View All Components**.
2. Click **Create**, and select the type of component or layout you want to create.
3. Enter a name for the component or layout. You can't use a name used by another component or layout.

   You can use letters, numbers, underscores (_), and dashes (-). If you enter a space, it's automatically replaced with a dash.

   Don't use the following names for site templates, themes, components, sites, or site pages: authsite, content, pages, scstemplate_*, _comps, _components, _compsdelivery, _idcservice , _sitescloud, _sitescloudddelivery, _themes, _themesdelivery. Although you can use the following names for site pages, don't use them for site templates, themes, components, or sites: documents, sites.
4. Optionally, enter a description for the component or layout.

5. For component type, select one of the following options:

Option	Description
Mustache	Renders with a Mustache template.
Preact	Renders with a JSX template.
React	Renders with a JSX template.
Knockout (Sandboxed)	Renders with Knockout in an inline frame using an iframe element.
Knockout	Renders with Knockout and contains nested components. The component is inserted directly into the page.

> **Note:**
>
> If selecting one of the template options (Mustache, Preact, or React), the local component is not Knockout-based and so cannot use nested components when placed on a page. For example this means that editing title or body text must be done through custom settings in the settings panel of the component, rather than selecting the component text as displayed on the page.

6. Click **Create**.

   A progress bar shows the creation status. When the component, layout, or form is created, the name appears in the list of components. You can explore the folders and files that make up the component or layout by clicking the component or layout name in the list of components.

7. To select an icon other than the default icon assigned to the component or layout:
   a. Select the component or layout from the list.
   b. Click **Properties**.
   c. Click the **Component Logo** tab.
   d. Click a logo from the gallery of logos and then click **Done**.

For detailed information about how to create your own components, see Develop Components.

After you have customized your component, layout, content field editor or content form, you can share it with others or promote it so they can use it in the following ways:

- Custom Component: When you use Site Builder, the component icon is added to the Custom Components panel in the editor with the name you assigned the component.
- Section Layout: When you use Site Builder, the section layout icon is added to the Section Layouts panel in the editor with the name you assigned the layout.
- Content Layout: A content administrator can assign the layout to one or more content types, either as the default view or added to a list of layouts that a site designer can select in Site Builder to specify how a content item of that type is displayed on the page.
- Content Field Editor: To make a content field editor available for use when creating content types, select it and click **Promote**, then confirm and click **OK**. Once promoted, content field editors are available to control the appearance of all data fields except *Media* and *Reference* when creating content types.

- Content Form: To make a content form available for use when creating content types, select it and click **Promote**, then confirm and click **OK**. Once promoted, content forms are available for use with all content types and asset types.

Export or Import Components or Layouts

You can export a component to modify it offline and then import it either as a new component or to replace the existing component. You can also export a component to move it to another Oracle Content Management instance and import it there.

Exporting

When you export a component, you essentially copy the component to a folder in the Oracle Content Management as a single .zip file. You can download the component directly from the folder to unpack and work with the individual files. When you are done working with the component files, create a .zip file that contains the component folders and files, import it, and overwrite the original component or create a new one.

> **Note:**
>
> When you export a component, sharing information for the component isn't included.

To export a component:

1. Click **Developer** and then click **View All Components**.
2. Select a component or layout and choose **Export** from the menu bar or the right-click menu.
3. Navigate to a folder or create new folder by clicking **New**, providing a name and an optional description, and clicking **Create**. To open a folder, click the folder icon or the folder name.
4. Select a folder by clicking the checkbox for the associated folder and click **OK**.

   A component or layout package file is created in the selected folder with the component or layout name and a .zip extension.

Importing

Before you can import or create a component, layout, or content field editor, your administrator must enable the options in the **Create** menu. If you don't see the **Create** menu on the components page, contact your administrator.

To import a component, layout or content field editor:

1. Click **Developer** and then click **View All Components**.
2. Click **Create** and choose **Import component**. Choose this option to import a component, a section layout, or a content layout.
3. If you uploaded the component or layout package, navigate to the folder that contains the component or layout and open the folder. If you haven't uploaded the package yet, go to the folder where you want to upload the component or layout or create a new folder. Click **Upload**, then find the component or layout package and click **Open**.

4. Click the checkbox next to the component or layout file name and click **OK**. A new component or layout is created and added to the component list. If the component or layout name or ID already exists, you're prompted to resolve the conflicts. You may need to create a new component or layout or you can overwrite the existing component or layout with the imported version.

8

Work with Site Pages

Let's create a page, choose a layout, and decide where the page goes in the site.

Navigate to a Page

To view and navigate the hierarchy of pages:

1. Open a site for editing. Make sure that  is set to Edit.
2. Click  to show options for managing and editing pages and page content.
3. Click  to list the first level of pages.
4. Click a page to view the page.

   Pages with an arrow (>) have a nested layer of pages. Click the page to show the nested pages.
   You can also use the search box at the top of the page list to find the page you want.
5. To view or change settings for a particular page, click  for that page. See Change Page Settings.

View Pages

To view the pages in a site, open the site in the editor and use the different preview, size, and orientation options.

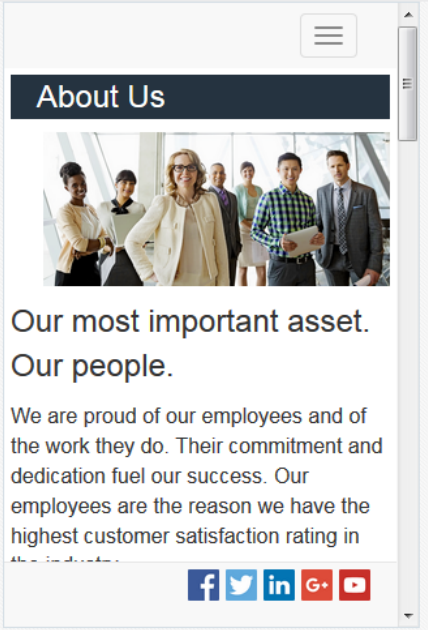
To change how you view pages while using the editor:

1. Open a site for editing.
2. Navigate to a page.
3. To preview the page in the editor, make sure that is set to Preview. This shows the page as it appears to your site visitors, without the visual aids used while editing.

4. To preview the current update applied to the base site in a separate browser window, click .

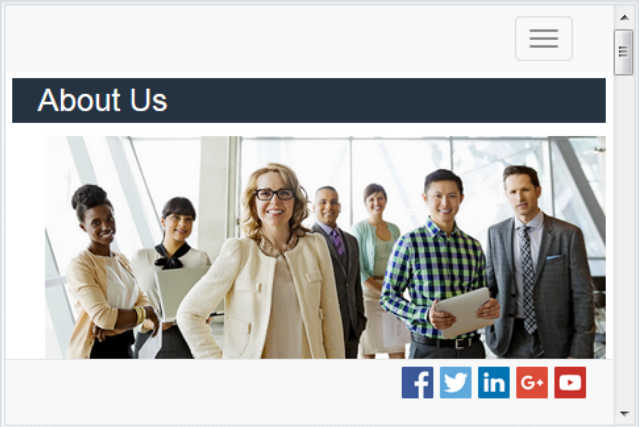
> **Note:**
>
> Links to site pages don't work in preview mode.

5. To preview the page as it appears on a device with a particular screen size, click **Fit to Screen** in the top menu bar and choose a set of screen dimensions from the list. Themes with a responsive design automatically arrange the page content for the best use on the selected screen size.



6. To create a custom size, click **Fit to Window** and choose **Create a device preset..**. Complete the necessary fields. To delete a custom size, click next to the size. You can also activate the ruler and select any interval on the ruler to quickly see different sizes.

7. To alternate the page orientation between portrait and landscape, click .



Add Pages

The pages in a site are structured in a folder-like hierarchy, or *site tree*. You can add a page, define settings for the page, and decide where the page goes in the site.

You can choose to add a page or add a link to an external page.

- **Web Page**: The page and content reside with the site. You name the page and define its content, specify where the page goes in the site, and specify how the page behaves in the context of the current site.
- **External Link**: The page is referenced from another location specified by a URL. You can name the page, specify where goes in the site, and specify how the page behaves in the context of the current site. Because you're using a page from a live site, you can't change the content of a linked page.

  In the site tree, a page that links to an external URL has before the page name.

Add a Web Page

To add a web page to the site:

1. Open a site for editing.
2. Select the level or branch where you want to add the page, then click **Add Page**, or, to add a child page, click . "New Page" is added to the bottom of the site tree and you're prompted to name the page and specify other settings.
3. Select **Web page** as the page type. Use this option if you want to manage the content on the page instead of reusing a page from another site.
4. Give the page a name. You can use letters, numbers, underscores (_), and hyphens (-). Don't use the following names for site templates, themes, components, sites, or site pages: authsite, content, pages, scstemplate_*, _comps, _components, _compsdelivery, _idcservice , _sitescloud, _sitescloudddelivery, _themes, _themesdelivery. Although you can use the following names for site pages, don't use them for site templates, themes, components, or sites: documents, sites.
5. Specify the page URL. By default, the page name is used for the page URL. Spaces are automatically replaced with hyphens. To use a URL other than the default, click **Override** and add the file name used in the URL. You can use letters, numbers, underscores (_), and hyphens (-). Be sure to include a file extension. The default file extension is .html.

6. Choose a page layout. This defines the general structure of the page, but not the content. A layout contains one or more named *slots* where you can put content. The number and type of page layouts depends on the theme associated with your site.
7. Choose a mobile page layout. This defines the general structure of the page when viewed on a mobile device.
8. Specify a page title. It doesn't show up on the page itself, but in the browser title bar or browser tab when the page displays. To add a title on the page, use the editor to add a title component.
9. Add an optional description. This description doesn't show up on the page , but lets you add information about the page for other contributors or for your own use.
10. Specify optional keywords separated by commas to help search engines identify the content of the page. Keywords are useful to identify terms or concepts that don't appear in the text of the page or that appear in images. The keywords don't appear on the page, but search engines use them to locate and identify your site. Good descriptions, keywords, and synonyms can increase traffic to your website.
11. Add an optional page header scripting or tags for analytics or tracking. Validate any code you use in the header or footer to make sure it works properly and doesn't introduce any security risks to your site.
12. Add an optional page footer scripting or tags for analytics or tracking. By default, the footer contains the text for pop-ups displayed regarding the use of cookies on the site with a link to the Privacy Policy. If you're a developer, you can edit the text that's displayed. Validate any code you use in the header or footer to make sure it works properly and doesn't introduce any security risks to your site.
13. Optional: You can use any of the available page options:
    - **Error page**: If an error prevents a requested page from displaying, show this page instead of the default error page. By default, when you select this option, **Hide page from navigation** and all the search engine options are also selected. You can deselect the options if you want the error page to be included in the navigation or in search engine results.
    - **Hide page from navigation**: Don't include the page in the automatically generated navigation for the site.
    - **Detail page**: Use this page to display detail information for a content item selected on another page. Structured content is available for enterprise users only. By default, when you select this option, **Hide page from navigation** is also selected. You can deselect the option if you do want the detail page to be included in the navigation.
      When structured content items are configured to use the detail page and a user clicks the link for more details on a structured content item, the detail page is displayed with detailed information for the content item.
    - **Search page**: Use this page to display results from a search. You can select this page in the Link settings of a Content Search component. See Content Search. By default, when you select this option, **Hide page from navigation** is also selected. You can deselect the option if you do want the search page to be included in the navigation.
14. Optional: Select a search engine optimization (SEO) option:
    - **Hide page from search engines**: Notify search engines not to index the content of the page so it doesn't show up in search results.

- **Hide page links from search engines**: Notify search engines not to follow links on the page and consequently don't index the link destinations.
- **Disable search engine page caching**: Notify search engines not to cache this page.
- **Hide page descriptions from search engines**: Notify search engines not to include the description (specified above) after the page in the search results.

15. Optional: select if this page will have a Cobrowse button to use with browsing sessions. A cobrowse session uses Oracle Cobrowse Cloud Service to manage a screen sharing experience with a site visitor. See Enable Cobrowse Integration and Using Cobrowse on a Page.
16. Optional: Override the site's analytics tracking snippet, and add a snippet specific to this page. Adding a snippet of JavaScript tracking code for web analytics tracking, makes it easier to integrate with external analytics providers like Google, Adobe, or Oracle Infinity. See Add Analytics Tracking.
17. To save all pending changes in the current update, click **Save**.

Add a Linked Page

To add an external link page to the site:

1. Open a site for editing.
2. Go to the page at the level in the site tree where you want to add the page. Pages are added at the current level, but you can easily move them to another location in the site tree.
3. Click **Add Page**. "New Page" is added to the bottom of the site tree and you're prompted to name the page and specify other settings.
4. Select **External link** as the page type.
5. Give the page a name. You can use letters, numbers, underscores (_), and hyphens (-). The name is used in the site tree to identify the page. Spaces are automatically replaced with hyphens.
6. Specify the full URL to the page. For example: `https://www.example.com/sharedpage.htm`
7. If you want the page to open in a new browser window or tab, select **Open link in a new window**. If you don't select this option, the page opens in the current window, replacing your site page and navigation.
8. Optionally, select any of the available options:
   - **Error page**: If an error prevents a requested page from displaying, show this page instead of the default error page. By default, when you select this option, **Hide page from navigation** is also selected. You can deselect the option if you want the error page to be included in the navigation.
   - **Hide page from navigation**: Don't include the page in the automatically generated navigation for the site.
   - **Detail page**: Use this page to display detail information for a content item selected on another page. Structured content is available for enterprise users only. By default, when you select this option, **Hide page from navigation** is also selected. You can deselect the option if you do want the detail page to be included in the navigation. When structured content items are configured to use the detail page and a user clicks the link for more details on a structured content item, the detail page is displayed with detailed information for the content item.

- **Search page**: Use this page to display results from a search. You can select this page in the Link settings of a Content Search component. See Content Search. By default, when you select this option, **Hide page from navigation** is also selected. You can deselect the option if you do want the search page to be included in the navigation.

9. To save all pending changes in the current update, click **Save**.

Move Pages

You can drag and drop pages to another location in the current level of the site tree, or cut and paste a page to another level.

There's always a top page in the hierarchy, the "home" page for your site, and all other pages are listed beneath it. You can drag and drop pages at a given level to change their order. You can also cut and paste pages to another location in the hierarchy.

Pages can have nested pages. If you move a page with nested pages, the nested pages are also moved and remain nested beneath the moved page. If the theme you use includes automated navigation, changing the order of pages also changes the order in which they're listed in the navigation menus.

To move a page to a different location in the site tree:

1. Open a site for editing.
2. Go to the page you want to move.
3. To move a page to a different position in the current level of the site tree, click and drag the page name to a different location in the list of pages, then drop it.
4. To move a page to a different level of the site tree, click the page you want to move and click  for that page. Click  to cut the page from its current location or  to leave a copy of the page in the current location. Go to the level where you want the page and click  . If you select a page at that level, the pasted page will be nested under it. Click  to paste the page to the current location. To reorder the pages, drag and drop a page to another location in the current level of the site tree.

Delete Pages

You can delete pages as part of an update.

A deleted page is recorded as part of an update when you click **Save**. The page isn't deleted from the base site until you publish the update. Anyone with the Contributor or Manager roles can delete a page.

> **Note:**
>
> When you publish an update that deletes a page, the page is permanently deleted and can't be recovered. If you delete a page from a multilingual site, the translated version of the page will also be deleted when the site is published. If managing page additions or deletions is a concern, you can isolate those actions using dedicated updates when adding or deleting pages from a site.

To delete a page from the site:

1. Open a site for editing.
2. Navigate to the page that you want to delete and click  .
3. Click **OK** to confirm the deletion. Click **Save** to save all pending changes in the current update.

Change Page Settings

You can change page settings such as name, title, URL, headers, footers, and other options. You can also use the Properties tab to add custom page properties.

1. Open a site for editing.
2. Click the page for which you want to change the settings, then click  .
3. Change the settings as needed:
   - **Page Type**:
     - **Web Page**: The page and content reside with the site. You name the page and define its content, specify where the page goes in the site, and specify how the page behaves in the context of the current site.
     - **External Link**: The page is referenced from another location specified by a URL. You can name the page, specify where goes in the site, and specify how the page behaves in the context of the current site. Because you're using a page from a live site, you can't change the content of a linked page.

       In the site tree, a page that links to an external URL has  before the page name.

   If you select Web Page, you see the following settings:

   - **Page Name**: You can use letters, numbers, underscores (_), and hyphens (-). Don't use the following names for site templates, themes, components, sites, or site pages: authsite, content, pages, scstemplate_*, _comps, _components, _compsdelivery, _idcservice , _sitescloud, _sitesclouddelivery, _themes, _themesdelivery. Although you can use the following names for site pages, don't use them for site templates, themes, components, or sites: documents, sites.
   - **Page URL**: By default, the page name is used for the page URL. Spaces are automatically replaced with hyphens. To use a URL other than the default, click **Override** and add the file name used in the URL. You can use letters, numbers, underscores (_), and hyphens (-). Be sure to include a file extension. The default file extension is .html.

- **Page Layout**: The page layout defines the general structure of the page, but not the content. A layout contains one or more named *slots* where you can put content. The number and type of page layouts depends on the theme associated with your site.
- **Mobile Page Layout**: The mobile page layout defines the general structure of the page when viewed on a mobile device.
- **Page Title**: The page title doesn't show up on the page itself, but in the browser title bar or browser tab when the page displays. To add a title on the page, use the editor to add a title component.
- **Page Description**: The description doesn't show up on the page , but lets you add information about the page for other contributors or for your own use.
- **Keywords**: Specify optional keywords separated by commas to help search engines identify the content of the page. Keywords are useful to identify terms or concepts that don't appear in the text of the page or that appear in images. The keywords don't appear on the page, but search engines use them to locate and identify your site. Good descriptions, keywords, and synonyms can increase traffic to your website.
- **Page Header**: Add an optional page header scripting or tags for analytics or tracking. Validate any code you use in the header or footer to make sure it works properly and doesn't introduce any security risks to your site.
- **Page Footer**: Add an optional page footer scripting or tags for analytics or tracking. By default, the footer contains the text for pop-ups displayed regarding the use of cookies on the site with a link to the Privacy Policy. If you're a developer, you can edit the text that's displayed. Validate any code you use in the header or footer to make sure it works properly and doesn't introduce any security risks to your site.
- **Page Options**:
  - **Error page**: If an error prevents a requested page from displaying, show this page instead of the default error page. By default, when you select this option, **Hide page from navigation** and all the search engine options are also selected. You can deselect the options if you want the error page to be included in the navigation or in search engine results.
  - **Hide page from navigation**: Don't include the page in the automatically generated navigation for the site.
  - **Detail page**: Use this page to display detail information for a content item selected on another page. Structured content is available for enterprise users only. By default, when you select this option, **Hide page from navigation** is also selected. You can deselect the option if you do want the detail page to be included in the navigation.
    When structured content items are configured to use the detail page and a user clicks the link for more details on a structured content item, the detail page is displayed with detailed information for the content item.
  - **Search page**: Use this page to display results from a search. You can select this page in the Link settings of a Content Search component. See Content Search. By default, when you select this option, **Hide page from navigation** is also selected. You can deselect the option if you do want the search page to be included in the navigation.
- **SEO Options**:

  - **Hide page from search engines**: Notify search engines not to index the content of the page so it doesn't show up in search results.
  - **Hide page links from search engines**: Notify search engines not to follow links on the page and consequently don't index the link destinations.
  - **Disable search engine page caching**: Notify search engines not to cache this page.
  - **Hide page descriptions from search engines**: Notify search engines not to include the description (specified above) after the page in the search results.
- **Cobrowse**: If your administrator has enabled cobrowse for your system, and cobrowse has been enabled for this site, you see an option to select if this page will have a Cobrowse button to use with browsing sessions. A cobrowse session uses Oracle Cobrowse Cloud Service to manage a screen sharing experience with a site visitor. See Enable Cobrowse Integration and Using Cobrowse on a Page.
- **Analytics Options**: If this site includes an analytics tracking snippet, you can override the snippet for this page. Click the override switch, and edit the script. For more information, see Add Analytics Tracking. If you edit the site snippet, a message says that the script has been modified. To remove your customizations: **Restore to Latest Site Script**.

If you select External Link, you see the following settings:

- **Page Name**: You can use letters, numbers, underscores (_), and hyphens (-). The name is used in the site tree to identify the page. Spaces are automatically replaced with hyphens.
- **Link URL**: Specify the full URL to the page. For example: `https://www.example.com/sharedpage.htm`.
- **Open link in a new window**: If you want the page to open in a new browser window or tab, select this option. If you don't select this option, the page opens in the current window, replacing your site page and navigation.
- **Page Options**:
  - **Error page**: If an error prevents a requested page from displaying, show this page instead of the default error page. By default, when you select this option, **Hide page from navigation** is also selected. You can deselect the option if you want the error page to be included in the navigation.
  - **Hide page from navigation**: Don't include the page in the automatically generated navigation for the site.
  - **Detail page**: Use this page to display detail information for a content item selected on another page. Structured content is available for enterprise users only. By default, when you select this option, **Hide page from navigation** is also selected. You can deselect the option if you do want the detail page to be included in the navigation.
    When structured content items are configured to use the detail page and a user clicks the link for more details on a structured content item, the detail page is displayed with detailed information for the content item.
  - **Search page**: Use this page to display results from a search. You can select this page in the Link settings of a Content Search component. See Content Search. By default, when you select this option, **Hide page from navigation** is also selected. You can deselect the option if you do want the search page to be included in the navigation.

4. To save all pending changes in the current update, click **Save**.

Add Custom Page Properties

Custom properties specific to a particular page can be added in the page settings properties tab. Unlike common page properties, custom page properties are exclusive to the page for which they're defined.

1. Open a site for editing and select the page to which you want to add custom properties, then click  for the page, next to the page name.
2. Click the **Properties** tab. If any common properties are defined in site properties, they are listed in the common properties section and are available for use on this page. You can also edit common properties here and the edits apply only to this specific page. To reset the values for any common property you edit back to the default, click  .
3. In the custom properties section, click **Add**.
4. Enter a name and value for the custom page property. You can add up to 50 custom site properties. There is a 64 character limit on the name field and a 300 character limit to the value field.
5. Click the **X** next to a name/value pair to delete it.
6. When you are finished adding or removing your custom page properties, click **Close**.

> **Note:**
>
> Changes are not merged with existing custom page properties. Committing changes to custom page properties overwrites any existing custom page properties in the base site.

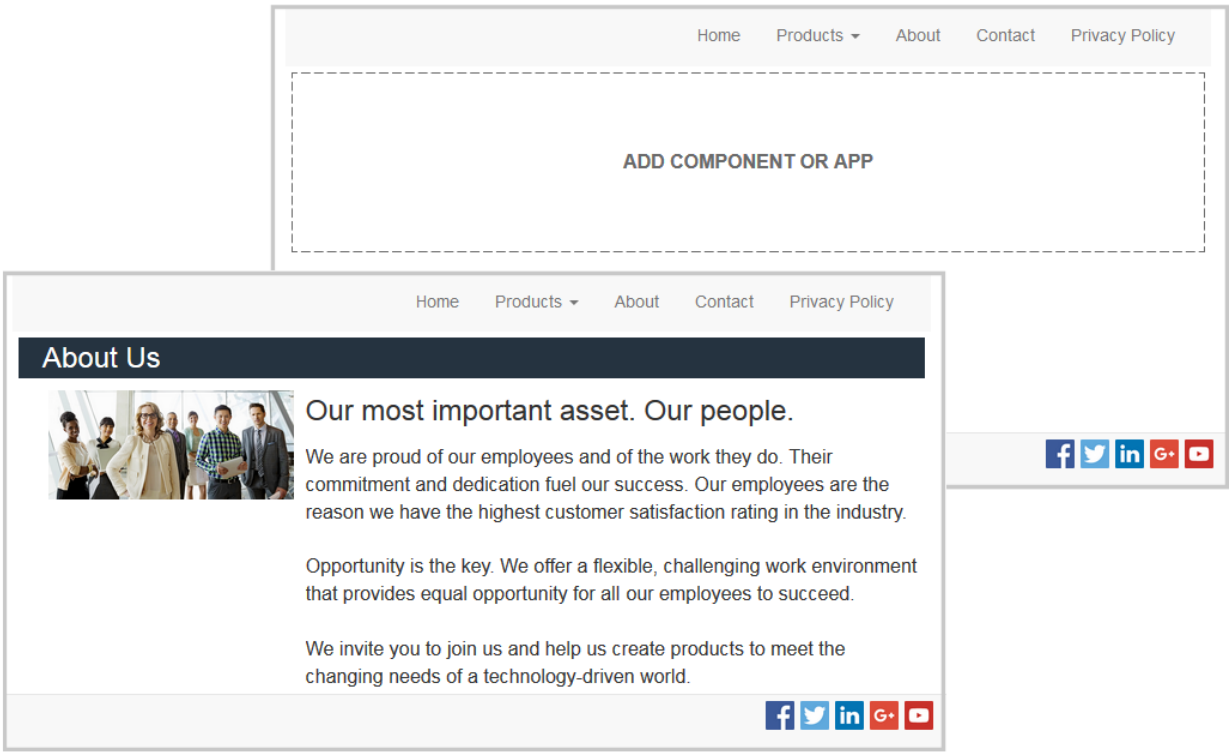
Change the Page Layout

A layout defines how content is arranged on the page. Different layouts can contain a different number of named *slots*, which is a region spanning the width of the page. A slot can contain one or more types of content.

Every theme has several page layouts. When you add a page to a site, you select a layout to use for that page. Each layout has slots where you can drag and drop content. What content goes into these slots is up to you. It can be anything from titles, text, and dividers to multimedia, galleries, and social media. You can arrange the content in a slot, but you can’t change the number or arrangement of slots on the page. To do that, you have to use a new page layout.

> **Note:**
>
> You can swap one layout for another. Be careful though. If you choose a layout with fewer or differently named slots, existing content in other slots won't display in the new layout. The content isn't deleted, it just can't be displayed unless the layout you choose has a slot with the same name.

The following illustration shows a sample layout for a page. You can see the empty slot in the page layout and the completed page with title, image, and text added to the slot.



To select a different layout for a page:

1. Open a site for editing.
2. Navigate to the page and click  .
3. Go to the Page Layout field and select a different layout from the menu. The number and type of page layouts depends on the theme associated with your site.
4. To save all pending changes in the current update, click **Save**.

Change the Background or Theme

You can specify the background color and image for the page, for individual slots on the page, and for section layouts within a slot. You can also change the entire theme for a site.

A slot is a region that spans the width of the page and can contain one or more types of content. The background you specify for a slot applies to the entire slot and all the components in the slot.

Change the Background

Backgrounds layer on top of each other. If you specify a background for a slot, it sits “on top” of the background specified for the page. For most images and colors, the upper layer effectively overrides the lower layer. If you use a degree of transparency either in a background color or in images, the colors in lower layers can show through or blend with the colors used in upper layers.

Some components, like paragraphs and titles, can provide a background color as part of a predefined style or as a customized style. If you specify a background color for a component, it also layers on top of any section layout, slot, or page background. See Use Styles and Formatting.

The background options are similar for pages, slots, and section layouts, though you access them in slightly differently ways:

1. Open a site for editing.
2. Navigate to the page you want to edit and make sure that  is set to **Edit**.
3. To change the background for a page or slot:
   - Click  in the sidebar and then click  .
   - To specify the background for the entire page, click **Complete Page** and click  . To specify the background for a slot on the page, select the slot and click  .
4. To change the background for a section layout:
   - Click the menu icon  for the section layout and choose **Settings**.
   - Click **Background**. The background options are the same for pages, slots, and section layouts.
5. Click  to open the **File Picker** panel and navigate to the content item you want to use. You can use images from any location you can access. You can also use images that were shared with you or that you upload from a local or network file location. The window displays all available files. Choose the file type appropriate for the context. For example, if you’re choosing an image file, select a file with a valid image format (GIF, JPG, JPEG, PNG, or SVG).
6. Select an image and click **OK**.
7. To adjust the image settings:
   - Use **Position** to place the image on the page or in the slot.
   - Select a **Scale** option to adjust the presentation of the image:
     – **Fit**: The image is scaled so the entire image fits in the available space without distorting the image.

  - **Stretch**: The larger of the two dimensions (width or height) is scaled to fit the available space and the smaller dimension is stretched to fill the available space.
  - **None**: The image is used at its full pixel resolution and is cropped uniformly if it doesn't fit in the available space.
- Select a **Repeat** option to tile an image that's too small to fill the available space. This option doesn't apply if you select **Stretch** as the **Scale** option.
- Select **Do not scroll with page** to keep the image stationary while the user scrolls the page.

8. To select a color for the background, choose a color from the **Color** menu or click **More** to select a color from the complete range of colors.
   - Click in the spectrum bar on the right to choose a color and to display variations of the color in the color range display. The display shows the selected color in the upper right corner and shades of the selected color with increasing amounts of white toward the left and increasing amounts of black toward the bottom.
   - To select a variation of the color, click within the color range display or click and drag the selection point to a new color position. The current color box and the 6-digit color code show the selected color.
   - To adjust the transparency (alpha channel), click and drag the transparency slider to the left to increase transparency. An alpha value of 0% makes the color transparent, allowing color and content from lower layers to show through completely. An alpha value of 100% makes the color completely opaque, blocking out all color and content from lower layers. Other values allow images and colors from lower layers to show through and their colors to blend with those of upper layers to different degrees.
   - To apply the current color selection, click **Choose**.

Change the Theme

The change to the theme is recorded as part of an update when you click **Save**. The theme change isn't applied to the site until you publish the update.

> **Note:**
>
> Not all themes are compatible with each other. Different themes can contain different navigation, style name, or page layout information. For example, if you choose a theme with different layout names, existing pages may use layouts that aren't available in the theme. Those pages appear blank. The page content isn't deleted, it just can't be displayed unless the theme you choose has a layout and slots with the same names. If you choose an incompatible theme, you can change the theme back to the original theme to restore the page content.
> Be certain you switch to a compatible theme. If you aren't sure, check with the theme developer.

To change the theme used for the site:

1. Open a site for editing.
2. Click  in the sidebar and then click  .

3. Select a theme from the list of available themes and click **OK**. The theme is applied in the current update. To save all pending changes in the current update, click **Save**.

9

Arrange Page Content

Components and content items provide the features and content your users want and section layouts let you arrange them automatically.

- Add Components and Section Layouts
- Work with Assets and Content Items
- Use Triggers and Actions
- Use Horizontal Section Layouts
- Use Vertical Section Layouts
- Use Two and Three Column Layouts
- Use Tabbed Section Layouts
- Use Slider Section Layouts

For information about using individual components, see Use Built-In Components.

To learn how to manage components and layouts, see Manage Custom Components and Layouts.

For details about how to create your own components, see Develop Components .

Add Components and Section Layouts

Components are the individual parts of a web page. A *section layout* automatically organizes content added to it, making it easy for a contributor to add content without spending time formatting it on the page.

Add Components

To add a component to a page:

1. Navigate to the page you want to edit and make sure that  is set to **Edit**.
2. Click  and then one of the following types of components:
   - Click **Themed** to show the list of components that were chosen to be used in the theme associated with the site.
   - Click **Custom** to show the list of custom components that were shared with you.
   - Click **Seeded** to show the list of default components available with the service.
   - Click **All** to show all components that were shared with you.
3. Click and hold a specific icon and drag it where you want it on the page. When you drag an item to the page, the boundaries of available slots, section layouts, and any existing items are shown. A placement icon  or a vertical placement bar  (put it to the left or right) indicates where the new content will go. A solid border around a section layout or

content item indicates you can drop the item and it will be placed automatically. You can have multiple items in a slot and move items on the page just by dragging them to a new location.

4. When you're in the right location, drop the item onto the page.

5. To adjust the properties for an item, click the item's menu icon and choose **Settings**. Depending on the item, you'll need to add text, specify a link to an image, specify a URL to another site or a map, resize the item, or other actions.

Each component has settings such as size, alignment, spacing, color, and borders. These define how the component looks and acts. For example, paragraph settings include font type, font size and other features that determine how text is presented.

To adjust the properties of a component, select the component, and then click **Menu** and choose **Settings**. Different components have different kinds of settings:

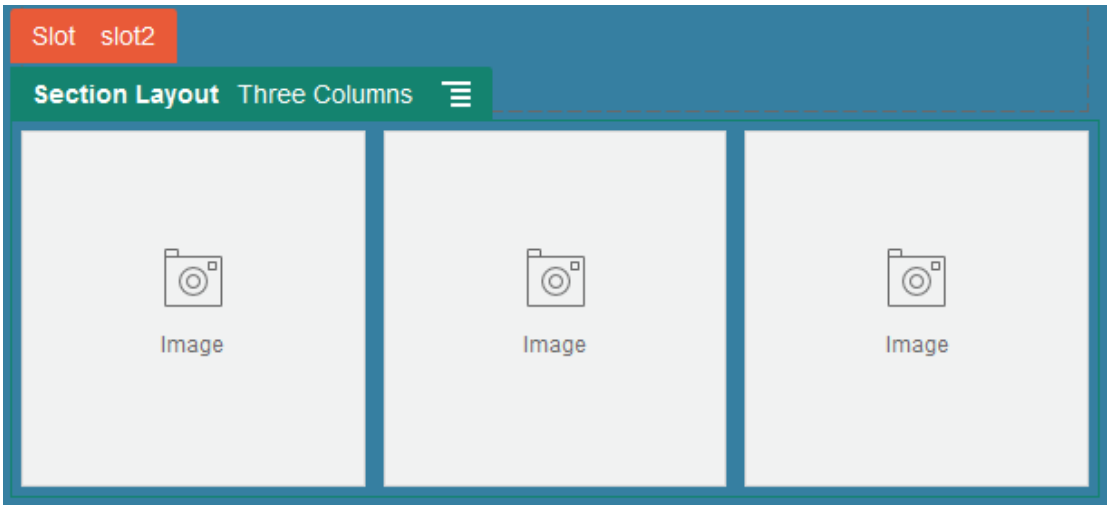
Settings	Description
General	General settings include spacing, alignment, and settings that are common among components.
Style	A style is a named set of default values that govern appearance. Styles are defined in the theme. Different themes can have different styles, and within a theme, different components can have different styles. You can also manually specify style settings for a specific instance of a component.
Link	Link settings include the locations of images, documents or other resources used by the component. For buttons and other components that perform actions, link settings also include the triggers and actions supported by the component.
Components	Components that include other standard components, such as the article component, provide a list of the individual components and give you access to the settings for each of those components.
Custom	Custom settings are unique to the component and are presented separately from the standard setting groups. Remote components, for example, may store preferences at a unique URL and present them as custom settings.

Using Section Layouts

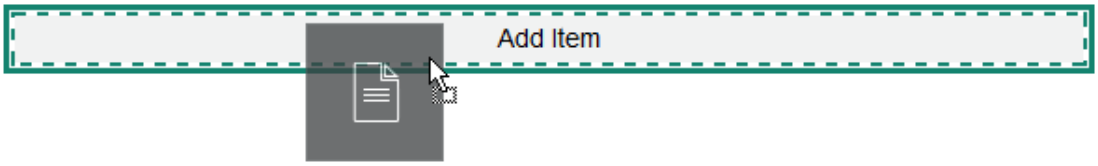
A section layout can automatically organize content into multiple columns, a vertical list, or a set of tabbed areas. A site contributor can add one or more section layouts to a slot to organize content. To add a layout to a page:

1. Navigate to the page you want to edit and make sure that is set to **Edit**.

2. Click to insert a section layout on your page. Choose the type of layout you want to use and follow the same guidance for placing the item as you use for placing a component.



A placement icon indicates where the section layout will go (above, below, left, right). A solid border around a section layout or content item indicates you can drop the item and it will be placed automatically:



If you are an enterprise user working with structured content items, you can assign a section layout when you add a list of content items to a page and the section layout will automatically format the items on the page.

In addition to the layouts provided, a developer can create additional section layouts to solve particular layout problems or to simplify authoring for contributors. A theme designer can even build section layouts into a slot in a page layout. See Develop Layouts.

Work with Assets and Content Items

If you are an enterprise user, your site can include digital assets and content items stored in a *site collection* or the associated repository. The collection is a subset of assets in a repository that can be used on the site.

To add a digital asset or content item to a page, make sure that is set to **Edit**, then click . If your site uses multiple repositories, select the repository to use. For information about using multiple repositories in a site, see Give a Site Access to Multiple Repositories in *Managing Assets with Oracle Content Management*.

You can filter your assets to find exactly what you need. Click and choose how you want to narrow your choices. If you don't see any assets, the assets might not be part of the site collection. Change the filter to show all collections in the repository instead of just the site collection to see if that helps. See Search, Filter, and Sort Assets for complete details.

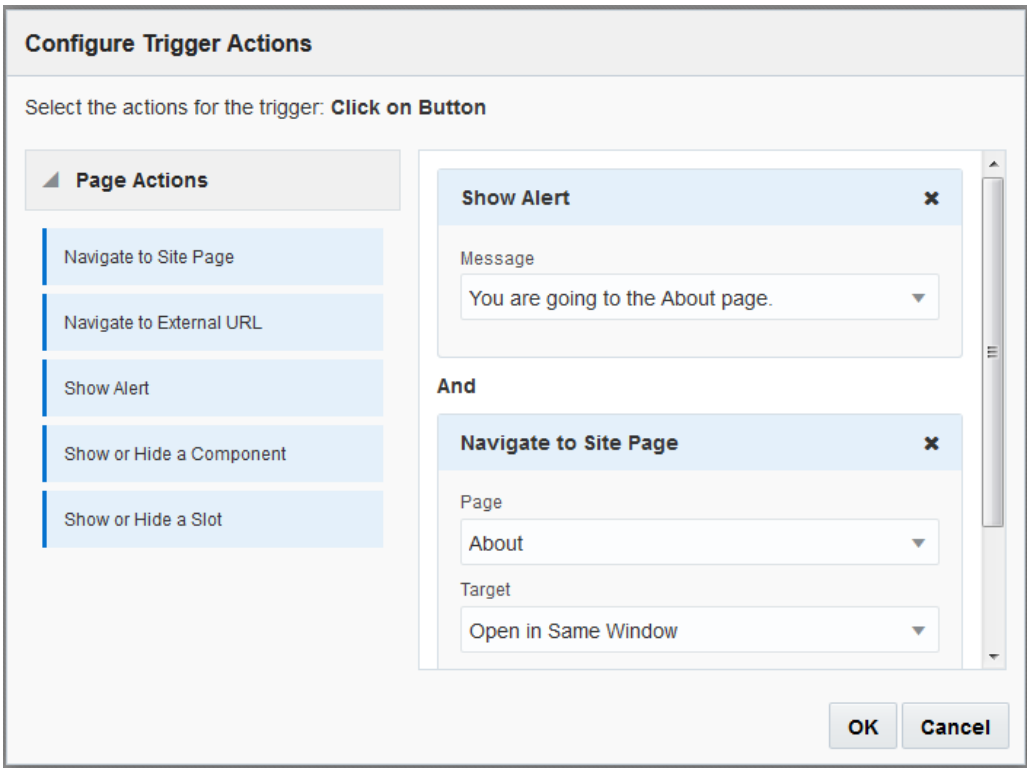
Drag the digital asset or content item from the panel and drop it into a slot on the page. You can embed images directly into a Paragraph component at a cursor location, with options to enter alternative text, set the image height and width, and set the alignment.

If you add an item from the site collection to a page, it is automatically placed in a component of the appropriate type. For example, if you add an image digital asset, it is automatically placed in an image component. If you add a content item, it is automatically placed in a content item component. Alternatively, you can add the image or content item component first and then drag the content item from the Content panel onto the component at a later time. Or you can select **Settings** for the component, then click **Select** to choose an image from your assets or from your documents list.

Use Triggers and Actions

Button components can initiate one or more actions such as showing or hiding page components and showing messages. Certain components, such as folder and file lists, can initiate actions in the companion component based on the selection a user makes.

For example, you can configure a button so that when the button is pushed (the trigger), the user is directed to another page or external URL and an alert is shown that notifies them of the change (the actions). The display you see will change depending on what kinds of components are also used.



If you use more than one action, consider the order of operation and put the actions in the order you want them performed. In the example above, list the alert action first. This will give the user time to read and dismiss the message before they are redirected to the page. If you list the redirect action first, the message may be replaced with the new page before the user has a chance to read it.

Page Actions

All components that support triggers and actions support page actions:

- **Navigate to Site Page**: Select a page on the current site.
- **Navigate to External URL**: Specify a full URL to an external page or site.
- **Show Alert**: Show a specified message in a window.
- **Show or Hide a Component**: Select a component from the list of components on the current page to show, hide, or toggle.
- **Show or Hide a Slot**: Select a slot from the list of slots on the current page to show, hide, or toggle.

Component-specific actions

In addition to page actions, components can define their own actions. These actions can allow a component to communicate with other components and initiate actions in a companion component. For example, when a user selects a folder in the folder list component, the file list component can display the files in the selected folder. In this case, the folder list component supports the **Folder Selected** trigger and the file list component supports the **Display Files** action.

When you add components that support actions to a page, the component and any actions they support are added to the list of available actions. Some components support only actions or certain actions. Some component provide triggers, but do not themselves support any actions.

To learn how to create your own components, see Develop Components.

For example, to specify one or more actions for a button component:

1. Click the button's menu icon and choose **Settings**.
2. In the Settings panel, click **Link**.
3. Click **Select Link Type** and choose **Trigger Actions**.
4. Under Available Triggers, click **Click on Button**.
5. In the Configure Trigger Action window, click and drag an action from the column on the left and drop it on the slot labeled **Do something**.

Use Horizontal Section Layouts

You can use a section layout to automatically determine the spacing and arrangement of components you add to the layout.

A horizontal layout arranges the items added to it one after the other in a horizontal line. The layout changes proportionally as the page width is increased or decreased. By default, items are fit into one line and are allocated equal horizontal space.

To add a layout to a page:

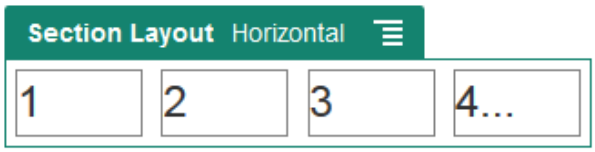
1. Navigate to the page you want to edit and make sure that is set to **Edit**.
2. Add the layout to the page.

3. To place other content in the section layout, drag and drop the content onto the layout.

   The layout highlights with a solid border and a banner that shows Add Item.

   You can continue adding items to the layout and the layout will format them accordingly. You can even add other section layouts to create sophisticated layouts.

   The following is a horizontal layout with numbered text components to show the sequence of items in the layout:



4. To edit the component and its appearance, click its menu icon, and choose **Settings**. You can set the width for individual areas if you don't want to use the default proportional sizing. You can also set the alignment (left, center, or right).
5. Use the General tab to modify settings for the individual components in the layout.

   Click a component name to see the settings for that component.
6. Use the Background tab to modify background settings for the layout.

   See Change the Background or Theme.

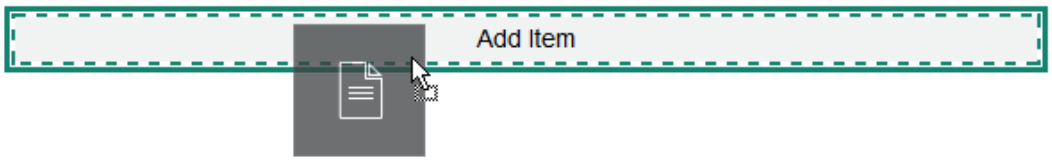
Use Two and Three Column Layouts

You can use a multi-column section layout to automatically determine the spacing and arrangement of components you add to the layout.

To add a layout to a page:

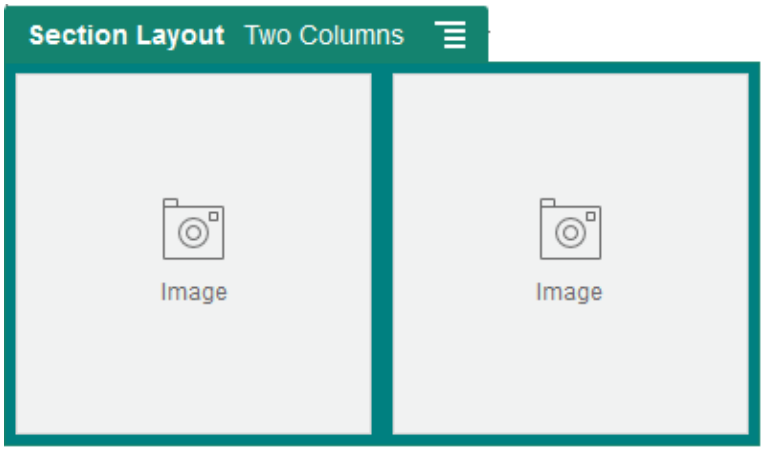
1. Navigate to the page you want to edit and make sure that is set to **Edit**.
2. Add the layout to the page.
3. To place other content in the section layout, drag and drop the content onto the layout.

   The layout highlights with a solid border and a banner that shows Add Item.



You can continue adding items to the layout and the layout will format them accordingly. You can even add other section layouts to create sophisticated layouts or add component groups.

The following is a two-column layout with image components as placeholders:



If you add more items, they are shown in additional rows, each with a maximum of two items.

4. To edit the component and its appearance, click its menu icon , and choose **Settings**.
5. Use the General tab to modify settings for the individual components in the layout.

   Click a component name to see the settings for that component.
6. Use the Background tab to modify background settings for the layout.
7. Choose **Custom Settings** to set additional defaults for the content that's displayed.
   - **First Column Width (%)**: Specify the column width as a percentage of the space available to the layout.
   - **Second Column Width (%)**: Specify the column width as a percentage of the space available to the layout.
   - **Third Column Width (%)**: Specify the column width as a percentage of the space available to the layout.
   - **Responsive Breakpoint (in pixels)**: For responsive page designs that automatically reformat the content when the available display size varies, specify the width in pixels where the section layout switches between the standard two-column layout and the **Responsive Behavior** options you specify below.
   - **Responsive Behavior**: Select how the layout changes when the available display size is smaller than the **Responsive Breakpoint** value.
     - **No Action**: Do not adjust the layout behavior.

- **Stack the columns**: Arrange the items from top to bottom in a single column with all items from column one, followed by items from column 2, and so on.
- **Hide the first column**: Hide the content in the first column to provide more space for the remaining columns.
- **Hide the second column**: Hide the content in the second column to provide more space for the remaining columns.
- **Hide the third column**: Hide the content in the third column to provide more space for the remaining columns.
- **Hide both columns**: Hide all the content in the layout.
- **Move the second column under the first column**: Arrange items in a single column with all items from column one followed by all items from column two.
- **Move the first column under the second column**: Arrange items in a single column with all items from column two followed by all items from column one.

Use Vertical Section Layouts

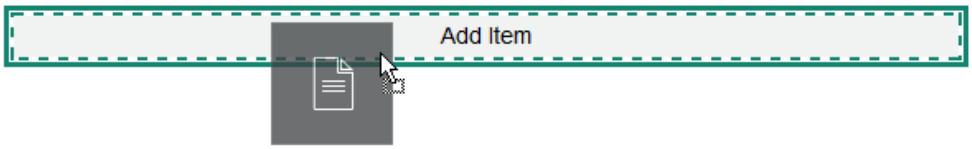
You can use a section layout to automatically determine the spacing and arrangement of components you add to the layout.

A vertical layout arranges the items added to it one after the other in a vertical line.

To add a layout to a page:

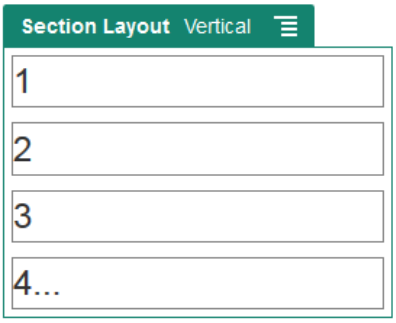
1. Navigate to the page you want to edit and make sure that  is set to **Edit**.
2. Add the layout to the page.
3. To place other content in the section layout, drag and drop the content onto the layout.

   The layout highlights with a solid border and a banner that shows Add Item.



   You can continue adding items to the layout and the layout will format them accordingly. You can even add other section layouts to create sophisticated layouts.

   The following is a vertical layout with numbered text components to show the sequence of items in the layout:



4. To edit the component and its appearance, click its menu icon, and choose **Settings**.
5. Use the General tab to modify settings for the individual components in the layout.

   Click a component name to see the settings for that component.
6. Use the Background tab to modify background settings for the layout.

Use Tabbed Section Layouts

You can use a tabbed section layout to create spacing and arrangement of the components you add into the layout.

To add a layout to a page:

1. Navigate to the page you want to edit and make sure that is set to **Edit**.
2. Add the layout to the page. By default, a single tab is added with the layout. Click **New Tab** to add additional tabs.
3. To place other content in a tab, drag and drop the content onto the tab.

   You can continue adding items to the layout and the layout will format them accordingly. You can even add other section layouts to create sophisticated layouts or add component groups.
4. To edit a tab and its appearance, click its menu icon, and choose **Settings**.
5. Use the Background section to modify background settings for the tab.

   You can use an image for the tab, change its position, and so on. See Change the Background or Theme.
6. Use the Style section to modify other appearance settings for the tab. Choose a style, such as hairline, frame, and so on, or customize the border and the corners for the tab.

Collapse Tabs in Section Layout Settings

When using tabs in a section layout component, you can specify the pixel width that triggers the tabbed content to stack and allows a site visitor to collapse and expand each tab. This is useful when a site visitor is viewing on a mobile device with a limited screen size for content.

To specify the pixel breakpoint that triggers the collapse response, use the custom settings in the general settings options of the section layout component that contains the tabs.

1. On the site page you are editing, add a section layout component.

2. Create the number of tabs you need and add the content you want in each tab.
3. Open the settings for the section layout component that contains the tabs.
4. Click **Custom Settings**.
5. Enter the pixel width at which you want the tabbed content to stack and collapse in the **Responsive Breakpoint** field.
6. Select **Show in Accordion** from the **Responsive Behavior** selector.
7. Close the settings and click **Save** in the action bar to save the changes to the site update.
8. Test the settings by toggling to **View** mode and selecting a viewport with a width below the breakpoint.

Use Slider Section Layouts

You can use a slider section layout to create content that remains in place until it is “slid” to one side, making room for new content. Users navigate through the slides by clicking the navigation dots under the slide section.

To add a slider layout to a page:

1. Navigate to the page you want to edit and make sure that  is set to **Edit**.
2. Add the layout to the page. By default, a single slide is added with the layout. Click **New Slide** to add additional slides.
3. To place content on a particular slide, display the slide in the slider, then drag and drop the component or content item onto the slide.

   You can continue adding items to the slide and positioning them as needed. You can add other section layouts to create sophisticated layouts or add component groups, letting you position content where you want.
4. To edit a slide layout and its appearance, click its menu icon  , and choose **Settings**.
5. Use the Background section to modify background settings for the entire slide area.

   See Change the Background or Theme. You can use an image, change its position, and so on.
6. Use the Style section to modify other appearance settings. Choose a style, such as hairline, frame, and so on, or customize the border and the corners for the slide.
7. To modify settings for each individual slide, click the name of the slide to open its settings. Change the background, style, and so on for each slide as needed.

10

Use Built-In Components

Components, section layouts, digital assets, and structured content give you the flexibility to provide the content and features your users want.

This section provides information about using individual components. For other information about components, see the following:

- To learn how to use components in your site, see Arrange Page Content.
- To learn how to manage components, see Manage Custom Components and Layouts.
- To learn how to create your own components, see Develop Components.

Basic Components

- Titles
- Paragraphs
- Plain Text
- Buttons

Structure Components

- Dividers
- Spacers

Media Components

- Images
- Galleries
- Gallery Grid
- YouTube Videos
- Videos

Document Components

- Documents
- Folder Lists
- File Lists
- Documents Manager
- Project Library

Social Components

- Social Bar
- Facebook Like and Recommend

- Twitter Share and Follow
- Conversation Component
- Conversation List

Process Components

- Process Start Form
- Process Task List
- Task Detail Form

Content Items

- Content Item Component
- Content Placeholder
- Content List
- Dynamic List
- Content Search
- Recommendation

Other Components

- Maps
- Headlines
- Articles
- Images with Text
- Component Groups
- Using Cobrowse on a Page
- Oracle Intelligent Advisor
- Oracle Visual Builder

Basic Components

Basic components provide the building blocks for text content.

- Titles
- Paragraphs
- Plain Text
- Buttons

Titles

To add and format a title component:

1. Navigate to the page you want to edit and make sure that  is set to **Edit**.
2. Add the component to the page.

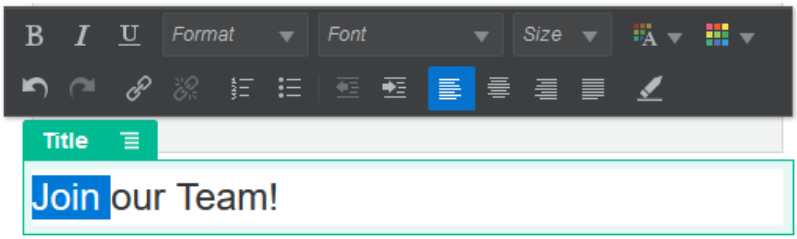
3. Click in the title component to enter the title text. The text takes on the formatting of the default style for the component.
4. To add a link within the title text:
   a. Enter and select the text you want to use as the link text, then click  .
   b. Click **Select Link Type** and choose one of the following options:
      - **Web Page**: Specify a full URL to an external page or site, and select where to open the link.
      - **Site Page**: Use the page picker to select a page on the current site, and select where to open the link. You can specify additional URL parameters in the format `key1=value1&key2=value2`. Empty values are supported; for example, `key1=&key2=value2`. You can also specify a URL anchor, but need to add a special Anchor section layout to the place on the target site page where you want the anchor link to resolve, and specify the same anchor name in layout settings that you used when defining the trigger action.

        > **Note:**
        >
        > The Anchor section layout required to use URL anchors is distributed in the Oracle Content Management Toolkit. For information on how to get the toolkit, see Develop with Oracle Content Management Toolkit.

      - **File Download**: Download a selected file from the repository. Select a file.

        If you select a digital asset, you can select a specific rendition. If you don't select a rendition, the original size will be used. If you want the latest version of the asset to be published when the site is published, select **Use latest version of asset**. If you don't select **Use latest version of asset**, then the most recent *published* version is used rather than a more recent draft version if there is one.
      - **File Preview**: Preview and optionally download a selected file in an overlay over a dimmed and inactive version of the page.
      - **Content Item**: Click  to select a content item from an associated asset repository. Choose the detail page you want displayed and target the page to open in the same or new window.
      - **Email**: Specify a valid email address and, optionally, a subject. The resulting message is opened in and sent through the default email client.
      - **Map**: Enter a valid address or coordinates, and select where you want the map to open in desktop and mobile browsers.
      - **Phone call**: Enter a valid phone number.

      To remove a link, click anywhere in the link text and click  .
5. If you want to change the default formatting, select the text that you want to format, then select any of the options in the formatting tool bar, such as font, color, or alignment. Depending on your theme, you may not see all these options.



Changes you make are applied immediately. These formatting changes are applied over the base style. If you change the base style, the overrides remain in place.

6. To remove the formatting applied with these options, select the text and click .

Paragraphs

To add and format a paragraph component:

1. Navigate to the page you want to edit and make sure that is set to **Edit**.
2. Add the component to the page.
3. Click in the paragraph component to enter the text. The text takes on the formatting of the default style for the component.
4. To add a video (), image (), or content item () to a paragraph, click the appropriate icon.
5. Click to open the **Select Asset** panel and navigate to the content item you want to use. Select a different asset repository if necessary, or select **Add** then **Add from Documents** or **Add from this computer** if the item is not in a repository.
6. To add an additional paragraph within a paragraph component, just press Enter.
7. To add a link within the paragraph:
   a. Enter and select the text you want to use as the link text, then click .
   b. Click **Select Link Type** and choose one of the following options:
      - **Web Page**: Specify a full URL to an external page or site, and select where to open the link.
      - **Site Page**: Use the page picker to select a page on the current site, and select where to open the link. You can specify additional URL parameters in the format `key1=value1&key2=value2`. Empty values are supported; for example, `key1=&key2=value2`. You can also specify a URL anchor, but need to add a special Anchor section layout to the place on the target site page where you want the anchor link to resolve, and specify the same anchor name in layout settings that you used when defining the trigger action.

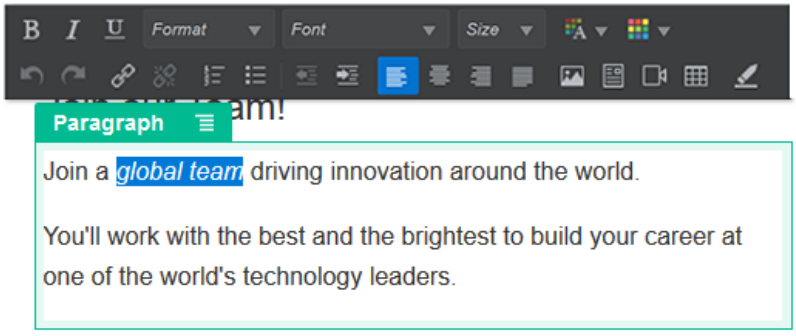
> **Note:**
>
> The Anchor section layout required to use URL anchors is distributed in the Oracle Content Management Toolkit. For information on how to get the toolkit, see Develop with Oracle Content Management Toolkit.

- **File Download**: Download a selected file from the repository. Select a file.

  If you select a digital asset, you can select a specific rendition. If you don't select a rendition, the original size will be used. If you want the latest version of the asset to be published when the site is published, select **Use latest version of asset**. If you don't select **Use latest version of asset**, then the most recent *published* version is used rather than a more recent draft version if there is one.
- **File Preview**: Preview and optionally download a selected file in an overlay over a dimmed and inactive version of the page.
- **Content Item**: Click  to select a content item from an associated asset repository. Choose the detail page you want displayed and target the page to open in the same or new window.
- **Email**: Specify a valid email address and, optionally, a subject. The resulting message is opened in and sent through the default email client.
- **Map**: Enter a valid address or coordinates, and select where you want the map to open in desktop and mobile browsers.
- **Phone call**: Enter a valid phone number.

To remove a link, click anywhere in the link text and click  .

8. If you want to change the default formatting for any portion of text, select the text that you want to format, then select any of the options in the formatting tool bar, such as font, color, or alignment. Depending on your theme, you may not see all these options.



Changes you make are applied immediately. These formatting changes are applied over the base style. If you change the base style, the overrides remain in place.

9. To remove the formatting applied with these options, select the text and click  .

Plain Text

The text component lets you add text to the page and format it exclusively using styles defined in the theme. The formatting tool bar is not available when editing the content, so you can't override the selected style with custom formatting.

This can be useful if you use standardized formatting for elements such as headings. You can update the styles defined in the theme and automatically update the associated text throughout the site without format overrides that would prevent it.

To add a text component:

1. Navigate to the page you want to edit and make sure that  is set to **Edit**.
2. Add the component to the page.
3. Click in the paragraph component to enter the text. The text takes on the formatting of the default style for the component (<p> or paragraph element).

   To add an additional paragraph, just press Enter.

   > **Note:**
   >
   > You can copy and paste content from other sources into the text component, but images and underlying HTML tags are removed.

4. To change the base style for the text component:

   a. Click the component's menu icon  and choose **Settings**.

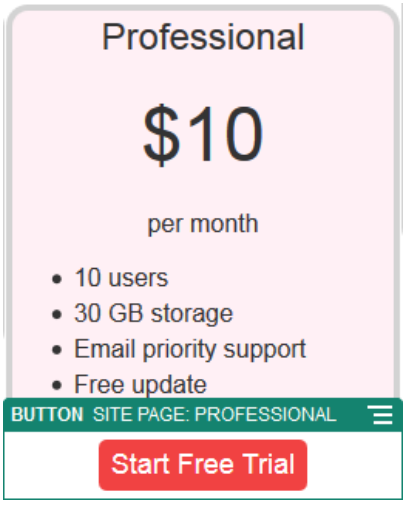
   b. Click **Choose Style** and choose the style from the menu.

      The list includes standard HTML tags for paragraph and headings. The style you select assigns the associated tag to the content.

   The component reflects the selected base style.

Buttons

Use a button to make a link or other functionality more apparent on the page.



To add a button to the page:

1. Navigate to the page you want to edit and make sure that  is set to **Edit**.
2. Add the component to the page.
3. To adjust the properties for the button, click its menu icon  and choose **Settings**.

   You can specify the text on the button (label), size, alignment and other display options for the button.

   > **Note:**
   >
   > If you set the **Width** and **Height** fields to 0 (zero), the button will automatically size to the fit the text you specify for the label.

   To specify the background color, font, border, and other settings, click the **Style** tab. You can choose from the predefined styles in the current theme. Depending on your theme, you may also be able to click **Customize** to specify your own values.
4. To associate a link or other actions with a button:
   a. In the Settings panel, click **Link**.
   b. Click **Select Link Type** and choose one of the following options:
      - **No Link**: The button performs no action when the user clicks it.
      - **Web Page**: Specify a full URL to an external page or site, and select where to open the link.
      - **Site Page**: Use the page picker to select a page on the current site, and select where to open the link. You can specify additional URL parameters in the format `key1=value1&key2=value2`. Empty values are supported; for example, `key1=&key2=value2`. You can also specify a URL anchor, but need to add a special Anchor section layout to the place on the target site page where you want the anchor link to resolve, and specify the same anchor name in layout settings that you used when defining the trigger action.

> **Note:**
>
> The Anchor section layout required to use URL anchors is distributed in the Oracle Content Management Toolkit. For information on how to get the toolkit, see Develop with Oracle Content Management Toolkit.

- **File Download**: Download a selected file from the repository. Select a file.

  If you select a digital asset, you can select a specific rendition. If you don't select a rendition, the original size will be used. If you want the latest version of the asset to be published when the site is published, select **Use latest version of asset**. If you don't select **Use latest version of asset**, then the most recent *published* version is used rather than a more recent draft version if there is one.
- **File Preview**: Preview and optionally download a selected file in an overlay over a dimmed and inactive version of the page.
- **Content Item**: Click  to select a content item from an associated asset repository. Choose the detail page you want displayed and target the page to open in the same or new window.
- **Email**: Specify a valid email address and, optionally, a subject. The resulting message is opened in and sent through the default email client.
- **Trigger Actions**: Select one or more page actions to perform when the button is pushed. Page actions include:
  - **Navigate to Site Page**: Select a page on the current site.
  - **Navigate to External URL**: Specify a full URL to an external page or site.
  - **Show Alert**: Show a specified message in a window.
  - **Show or Hide a Component**: Select a component on the current page to show, hide, or toggle.
  - **Show or Hide a Slot**: Select a slot on the current page to show, hide, or toggle.
- **Map**: Enter a valid address or coordinates, and select where you want the map to open in desktop and mobile browsers.
- **Phone call**: Enter a valid phone number.

See Use Triggers and Actions.

Structure Components

Structure components help to separate content on the page.

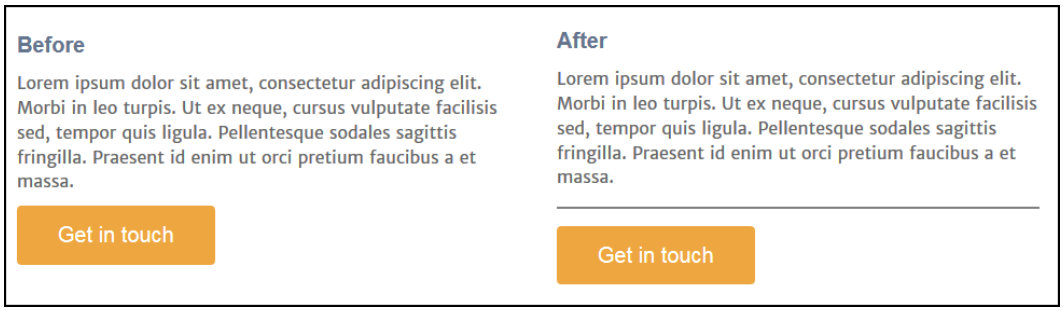
- Dividers
- Spacers

Dividers

Use a divider (horizontal line) to create a visual break in a column or across a page.

To add a divider to the page:

1. Navigate to the page you want to edit and make sure that  is set to **Edit**.
2. Add the component to the page.



3. To adjust the properties for the divider, click its menu icon  and choose **Settings**.

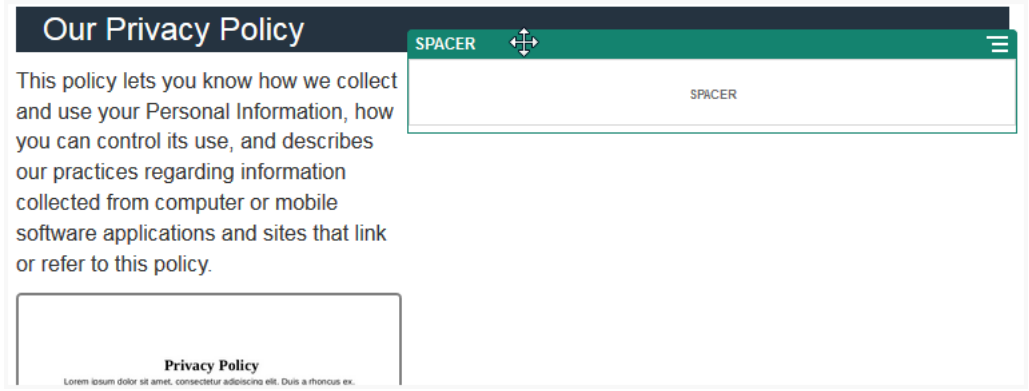
   You can specify the horizontal placement and length of the divider by adjusting the left and right spacing options.
   Line color, thickness, and other settings are defined by the component style. You can choose from the predefined styles in the current theme or click **Customize** to specify your own values.

Spacers

Managing the “white space” on pages can make the page more visually appealing and more readable. Use the spacer component to add white space without having to override the spacing defined in styles or in other components.

To add vertical blank space between components in the page:

1. Navigate to the page you want to edit and make sure that  is set to **Edit**.
2. Add the component to the page.



3. To adjust the height of the spacer, click its menu icon and choose **Settings**.

Media Components

Media components bring image and video to your pages.

- Images
- Galleries
- Gallery Grid
- YouTube Videos
- Videos

Images

Images can direct a viewer's attention and invite the viewer to explore different content areas on the page.

To add an image to a page:

1. Navigate to the page you want to edit and make sure that is set to **Edit**.
2. Add the component to the page. The image component shows a placeholder image until you select the image you want to use.

3. To select an image, double-click the component on the page to open the image selection panel directly, or click the menu icon, choose **Settings**, and click **Select** next to the Image field to open the selection panel.
4. Select an image from the site repository, from a documents folder that's been shared with you, or upload an image to a documents folder. You can also add an image to the repository first from the image selection panel by clicking **Add** and selecting **Add from Documents** or **Add from this computer**. Once the image is added, you must then find and select it.

> **Note:**
>
> The window displays all available files. You must choose the file type that's appropriate for the context. For example, if you're choosing an image file, you must select a file with a valid image format (GIF, JPG, JPEG, PNG, or SVG).

   a. Locate and select the image you want to use.

If you don't see any digital assets, click , and change the collection filter to **All**.

b. If you selected an image from a documents folder, you can link to the file rather than copying it to the site. To link to the file, select **Use a reference to the original file instead of copying the file to the site**. If you don't select this option, a copy of the file is stored with the site and is referenced from the site. Linking to the original file avoids duplicating the content. The link allows site visitors to see the content even if the permissions on the file change or would otherwise restrict viewing.

c. Once you've selected an image, click **OK** to close the image selection panel.

d. If you selected a digital asset, you can select a specific rendition. If you don't select a rendition, the original size will be used. If you want the latest version of the asset to be published when the site is published, select **Use latest version of asset**.

5. To edit the image, click  , and edit the image with any of the following actions:

- To crop the image, click  **Crop**. Select one of the predefined image ratios in the cropping toolbar, or drag the cropping handles on the image as desired. When you're satisfied, in the cropping toolbar, click **Crop**.
- To rotate or flip the image, click  **Rotate**. In the rotation toolbar, enter a custom rotation degree, use the buttons to rotate the image left or right, or select whether to flip the image horizontally or vertically.
- To add a watermark to the image, click  **Watermark**. Add text to the image, changing the text size, style, color, and opacity as desired with the watermark tools.
- To change the image format, click  **Options**, then select a new format from the **Format** drop-down list.
- To change the background color, click  **Options**, then select an option from the **Background Color** drop-down menu.
- If you're editing a .jpg or a .webp (available on Google Chrome browsers), you can change the image quality to create a smaller file size. Click  **Options**, then enter the new percentage in the **Quality** box.
- To undo or redo your change, click  or  . To remove all changes you made, click **Reset**.
- To change the magnification of the image, use the zoom controls (   ).

6. Use the **General** panel of the **Settings** menu  to add a caption or to modify spacing, alignment, style, and other options.

7. You can associate a link or other actions with an image:

a. In the Settings panel, click **Link**.

b. Select one of the following options:

- **No Link**: The image performs no action when the user clicks it.
- **Web Page**: Specify a full URL to an external page or site, and select where to open the link.

- **Site Page**: Use the page picker to select a page on the current site, and select where to open the link. You can specify additional URL parameters in the format `key1=value1&key2=value2`. Empty values are supported; for example, `key1=&key2=value2`. You can also specify a URL anchor, but need to add a special Anchor section layout to the place on the target site page where you want the anchor link to resolve, and specify the same anchor name in layout settings that you used when defining the trigger action.

  > **Note:**
  >
  > The Anchor section layout required to use URL anchors is distributed in the Oracle Content Management Toolkit. For information on how to get the toolkit, see Develop with Oracle Content Management Toolkit.

- **File Download**: Download a selected file from the repository. Select a file.

  If you select a digital asset, you can select a specific rendition. If you don't select a rendition, the original size will be used. If you want the latest version of the asset to be published when the site is published, select **Use latest version of asset**. If you don't select **Use latest version of asset**, then the most recent *published* version is used rather than a more recent draft version if there is one.

- **File Preview**: The selected file will appear as an overlay on the page.
- **Content Item**: Select a content item from an associated asset repository, choose the detail page you want displayed and target the page to open in the same or new window.
- **Email**: Specify a valid email address and, optionally, a subject. The resulting message is opened in and sent through the default email client.
- **Image Preview**: The selected image will appear as an overlay on the page.
- **Map**: Enter a valid address or coordinates, and select where you want the map to open in desktop and mobile browsers.
- **Phone call**: Enter a valid phone number.

8. When you've finished, close the image settings dialog.

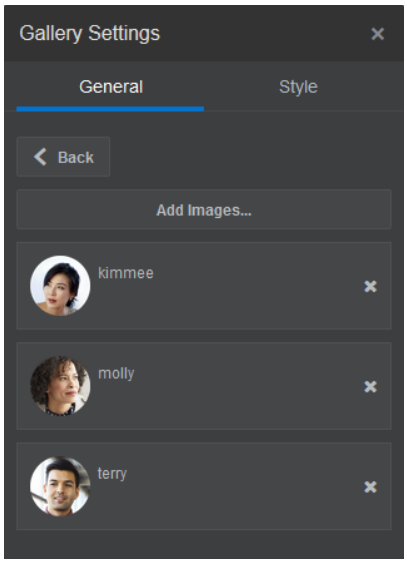
Galleries

Use an image gallery to present a series of images. You can choose to let the images automatically cycle or let the user advance through the images manually.

To add an image gallery to the page:

1. Navigate to the page you want to edit and make sure that  is set to **Edit**.
2. Add the component to the page. The gallery component shows a placeholder image until you select the images you want to use.
3. To add one or more images to the gallery, click its menu icon  , choose **Settings**, and click **Images** on the **General** tab.

4. Click **Add Images**.



5. Select one or more images.

> **Note:**
>
> The window displays all available files. You must choose the file type that's appropriate for the context. For example, if you're choosing an image file, you must select a file with a valid image format (GIF, JPG, JPEG, PNG, or SVG).

   a. Locate and select the images you want to use.

      If you don't see any digital assets, click , and change the collection filter to **All**.

   b. If you selected an image from a documents folder, you can link to the file rather than copying it to the site. To link to the file, select **Use a reference to the original file instead of copying the file to the site**. If you don't select this option, a copy of the file is stored with the site and is referenced from the site. Linking to the original file avoids duplicating the content. The link allows site visitors to see the content even if the permissions on the file change or would otherwise restrict viewing.

   c. Click **OK**.

   d. If you selected a digital asset, you can select a specific rendition. If you don't select a rendition, the original size will be used. If you want the latest version of the asset to be published when the site is published, select **Use latest version of asset**.

   The selected images are added to the list of images. Drag and drop the images to reorder them in the list. The default title of each image is the file name without the extension.

6. To change the title, description, or other options for a particular image, click the image in the list and make the change.

   You can also associate a link or other actions with an image in the gallery:

   a. In the Settings panel for a particular image, click the Link field.

   b. Select one of the following options:

- **No Link**: The image performs no action when the user clicks it.
- **Web Page**: Specify a full URL to an external page or site, and select where to open the link.
- **Site Page**: Use the page picker to select a page on the current site, and select where to open the link. You can specify additional URL parameters in the format `key1=value1&key2=value2`. Empty values are supported; for example, `key1=&key2=value2`. You can also specify a URL anchor, but need to add a special Anchor section layout to the place on the target site page where you want the anchor link to resolve, and specify the same anchor name in layout settings that you used when defining the trigger action.

> **Note:**
>
> The Anchor section layout required to use URL anchors is distributed in the Oracle Content Management Toolkit. For information on how to get the toolkit, see Develop with Oracle Content Management Toolkit.

- **File Download**: Download a selected file from the repository. Select a file.

  If you select a digital asset, you can select a specific rendition. If you don't select a rendition, the original size will be used. If you want the latest version of the asset to be published when the site is published, select **Use latest version of asset**. If you don't select **Use latest version of asset**, then the most recent *published* version is used rather than a more recent draft version if there is one.
- **Content Item**: Select a content item from an associated asset repository, choose the detail page you want displayed and target the page to open in the same or new window.
- **Email**: Specify a valid email address and, optionally, a subject. The resulting message is opened in and sent through the default email client.
- **Map**: Enter a valid address or coordinates, and select where you want the map to open in desktop and mobile browsers.
- **Phone call**: Enter a valid phone number.

c. Click **Back** to return to the image settings panel. Click **Back** again to return to the image list to select another image to update.

d. When you're done updating individual images, click **Back** to specify gallery options.

7. To size and scale images in the gallery:

   a. Use **Width** to specify the width, in pixels, of the gallery within the slot. Click an alignment option other than **Fill** to specify the width. After you set the width, you can use **Fill** to extend the image to the specified width.

   b. Select a **Scaling** option to adjust the presentation of images in the gallery:

      - **Crop**: The smaller of the two dimensions (width or height) is scaled to fit the available space and the larger dimension is cropped to prevent stretching the image.

- **Fit**: Each image is scaled so the entire image fits in the available space without distorting the image.
- **Stretch**: The larger of the two dimensions (width or height) is scaled to fit the available space and the smaller dimension is stretched to fill the available space.
- **None**: The image is used at its full pixel resolution and is cropped uniformly if it doesn't fit in the available space.

8. To help the user move through the gallery:
   a. Select a **Navigation** method:
      - **Thumbnails**: Show a list of the images in the gallery in sequence below the gallery. The user clicks a thumbnail image to go to the associated image in the gallery.
      - **Indexer**: Show a series of buttons  below the gallery to represent each image in the gallery. The user clicks a button to go to the associated image in the gallery.
      - **None**: Provide no visual navigation. The user can swipe right or left to display the adjacent image. This manual method is always available.
   b. Enable **Show Previous/Next** to include arrow icons on each image to advance to the previous or next image in the gallery.
   c. Enable or disable **Use arrow keys to navigate** to determine whether or not the arrow keys on the keyboard can be used to advance to the previous or next image in the gallery. If more than one gallery component is used on a page, all galleries with this setting enabled will move when an arrow is depressed. To avoid confusion when more than one gallery component is used on a page, it's best to enable this setting on one primary gallery and disable it on all the rest, instead enabling **Show Previous/Next** icons to allow someone to navigate images in a specific gallery.
9. You can elect to show or hide a caption for the image by selecting the **Show Caption** box.
10. To have the gallery cycle through the images automatically, select **Autoplay** and specify **Transition Time** and **Display Time**.

    The user can still use all the manual navigation features you set up.

Gallery Grid

Use a gallery grid to present a set of images at one time in rows and columns.

To add an image gallery to the page:

1. Navigate to the page you want to edit and make sure that  is set to **Edit**.
2. Add the component to the page. The gallery grid component shows a placeholder image until you select the images you want to use.
3. To add one or more images to the gallery, double-click the component on the page to open the image selection panel directly, or click its menu icon  , choose **Settings**, and click **Images** on the **General** tab.
4. Click **Add Images**.

5. Select one or more images from the site repository, from a documents folder that's been shared with you, or upload images to a documents folder. You can also add images to the repository first from the image selection panel by clicking **Add** and selecting **Add from Documents** or **Add from this computer**. Once the images are added, you must then find and select the ones you want to use.

   > **Note:**
   >
   > The window displays all available files. You must choose the file type that's appropriate for the context. For example, if you're choosing an image file, you must select a file with a valid image format (GIF, JPG, JPEG, PNG, or SVG).

   a. Locate and select the images you want to use.

      If you don't see any digital assets, click , and change the collection filter to **All**.

   b. If you selected an image from a documents folder, you can link to the file rather than copying it to the site. To link to the file, select **Use a reference to the original file instead of copying the file to the site**. If you don't select this option, a copy of the file is stored with the site and is referenced from the site. Linking to the original file avoids duplicating the content. The link allows site visitors to see the content even if the permissions on the file change or would otherwise restrict viewing.

   c. Once you selected the images, click **OK** to close the image selection panel.

   d. If you selected a digital asset, you can select a specific rendition. If you don't select a rendition, the original size will be used. If you want the latest version of the asset to be published when the site is published, select **Use latest version of asset**.

   The selected images are added to the list of images. Drag and drop the images to reorder them in the list (and in the grid). The default title of each image is the file name without the extension.

6. To change the title, description, or other options for a particular image, click the image to open additional settings and make the change.

   You can also associate a link or other actions with an image in the gallery:

   a. In the Settings panel for a particular image, click the Link field.

   b. Select one of the following options:

      - **No Link**: The image performs no action when the user clicks it.
      - **Web Page**: Specify a full URL to an external page or site, and select where to open the link.
      - **Site Page**: Use the page picker to select a page on the current site, and select where to open the link. You can specify additional URL parameters in the format `key1=value1&key2=value2`. Empty values are supported; for example, `key1=&key2=value2`. You can also specify a URL anchor, but need to add a special Anchor section layout to the place on the target site page where you want the anchor link to resolve, and specify the same

anchor name in layout settings that you used when defining the trigger action.

> **Note:**
>
> The Anchor section layout required to use URL anchors is distributed in the Oracle Content Management Toolkit. For information on how to get the toolkit, see Develop with Oracle Content Management Toolkit.

- **File Download**: Download a selected file from the repository. Select a file.

  If you select a digital asset, you can select a specific rendition. If you don't select a rendition, the original size will be used. If you want the latest version of the asset to be published when the site is published, select **Use latest version of asset**. If you don't select **Use latest version of asset**, then the most recent *published* version is used rather than a more recent draft version if there is one.

- **Content Item**: Select a content item from an associated asset repository, choose the detail page you want displayed and target the page to open in the same or new window.
- **Email**: Specify a valid email address and, optionally, a subject. The resulting message is opened in and sent through the default email client.
- **Image Preview**: The selected image will appear as an overlay on the page.
- **Map**: Enter a valid address or coordinates, and select where you want the map to open in desktop and mobile browsers.
- **Phone call**: Enter a valid phone number.

c. Click **Back** to return to the image settings panel. Click **Back** again to return to the image list to select another image to update.

d. When you're done updating individual images, click **Back** to specify gallery options.

7. Use the **Layout** option to arrange the images in a grid.

   - Masonry
   - Columns
   - Custom

   Each of these options is described in the steps that follow.

8. Choose the **Masonry** layout to automatically arrange the images in rows within the available space.

   The resulting rows have a uniform height, but no defined columns.

a. Specify **Height** to proportionately scale all images to the specified height in pixels.

b. Specify **Image Spacing** to increase or decrease the space between images in the row.

9. Choose the **Column** layout to arrange the images in rows and columns.

a. Select a **Scaling** option to adjust the presentation of images in the grid:

- **Crop**: The smaller of the two dimensions (width or height) is scaled to fit the available space and the larger dimension is cropped to prevent stretching the image.
- **Fit**: Each image is scaled so the entire image fits in the available space without distorting the image.

For example, the following grid uses four columns and scales the six images to fit:

Here's the same grid with the images cropped:

b. Specify an **Aspect Ratio** to determine the shape of the cells in the grid.
   - **Square**: An aspect ratio of 1:1.
   - **Landscape**: An aspect ratio of 16:9.
   - **Portrait**: An aspect ratio of 9:16.
   - **Custom**: Specify your own numeric values for the aspect ratio.

c. Specify the number of **Columns**.

   The grid adjusts automatically to create columns of equal width.

d. Specify **Image Spacing** to increase or decrease the space between images in both rows and columns.

10. Choose the **Custom** layout to arrange the images in rows and columns based on an image size and width that you specify.

    a. Select a **Scaling** option to adjust the presentation of images in the grid:
       - **Crop**: The smaller of the two dimensions (width or height) is scaled to fit the available space and the larger dimension is cropped to prevent stretching the image.
       - **Fit**: Each image is scaled so the entire image fits in the available space without distorting the image.

    b. Specify an **Image Height** and **Image Width** to determine the shape of the cells in the grid.

       The grid adjusts automatically to create cells with the dimensions you specify.

    c. Specify **Image Spacing** to increase or decrease the space between images in both rows and columns.

11. Specify **Alignment**, **Width**, and **Spacing** options to position the grid within the slot.

    Use **Width** to specify the width, in pixels, of the gallery within the slot. Click an alignment option other than **Fill** to specify the width. After you set the width, you can use **Fill** to extend the image to the specified width.

12. When you've finished, close the Gallery Grid settings dialog.

YouTube Videos

Include streaming YouTube videos to add motion and visual engagement to the page.

To use videos other than those hosted by YouTube, see Videos.

To add a YouTube video to the page:

1. Navigate to the page you want to edit and make sure that  is set to **Edit**.
2. Add the component to the page. The component shows a placeholder image until you select the video you want to use.
3. To specify the YouTube video to use and to adjust its display properties, click its menu icon  and choose **Settings**.
4. Locate the video you want to use, then copy the URL and paste it into the **YouTube URL** field.
5. Specify any display options:
   - **Show Controls**: Enable on-screen and device options to allow the user to manually control video playback.
   - **Show info**: Temporarily include the video description in the upper left corner of the video.
   - **Autoplay**: Automatically start video playback.
   - **Loop**: Automatically repeat the video after it completes.
6. Use the **Aspect Ratio** defined for the video (**Auto**) or choose another aspect ration to determine the shape of the video display.
7. Specify any alignment or spacing options to position the video.

Videos

Include videos from Oracle Content Management to add motion and visual engagement to the page.

Oracle Content Management offers several options to add video to a page. You can embed YouTube Videos hosted on their site, or you can add a video stored, managed, and delivered from Oracle Content Management to take advantage of the automatic transcoding and optimized streaming Video Plus offers. This is useful when sites are served to multiple devices with different capabilities. If your system administrator has not enabled Video Plus then standard video is an option.

To use videos hosted by YouTube, see YouTube Videos.

To use a video stored in an Oracle Content Management documents or asset repository:

1. Navigate to the page you want to edit and make sure that the Edit switch () is set to **Edit**.
2. To select a video, double-click the component on the page to open the video selection panel directly, or click the menu icon  , choose **Settings**, and click **Select** next to the Video field to open the selection panel.
3. Select a video from the site repository, from a documents folder that's been shared with you, or upload a video to a documents folder. You can also add a video to the repository first from the video selection panel by clicking **Add** and selecting **Add from Documents** or **Add from this computer**. Once the video is added, you must then find and select it.

4. Once you've selected a video, click **OK** to close the video selection panel.
    a. If Video Plus is enabled, you can select different repositories, search channels, collections and keywords, and filter assets to help you narrow your search when navigating to a video.
    b. When using standard video, choose a video file of type MP4. The MP4 video format is common to all supported browsers. Also, some standard videos have a predefined preview image. Those that don't show a blank screen as the preview image. If you want to use a graphic in place of a blank screen, click **Select** in the video settings dialog and choose a preview image from Oracle Content Management.
5. Once you have selected a video, use the settings dialog to specify any display options:
    - **Show Controls**: Enable on-screen and device options to allow the user to manually control video playback.
    - **Autoplay**: Automatically start video playback.
    - **Loop**: Automatically repeat the video after it completes.
    - **Muted**: Automatically mute the video when the page loads.
6. Specify any width, alignment or spacing options to position the video.
7. Click **Style** to choose or define a custom style for the video border.
8. When you've finished, close the video settings dialog.

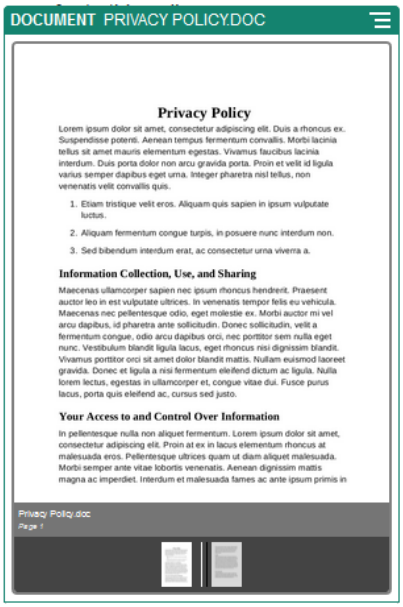
Document Components

Let's look at some components that let you access and display files and folders.

- Documents
- Folder Lists
- File Lists
- Documents Manager
- Project Library

Documents

You can view multi-page documents and slide presentations directly from a page.



To add a document to a page:

1. Navigate to the page you want to edit and make sure that  is set to **Edit**.
2. Add the component to the page. The document component shows a placeholder image until you specify the document to show.
3. To select a document, click its menu icon  , choose **Settings**, and click **Select** next to the Document field.

   The document must be stored in the site repository or another repository to which you have access. You can also use documents that were shared with you or that you uploaded from a local or network file location.
4. Select a document and click **OK**.

> **Note:**
>
> The window displays all available files. You must choose the file type that's appropriate for the context. For example, if you're choosing a document file, you must select a file with a valid document format such as TXT or DOC.

5. Use the Settings panel to add a caption or to modify spacing, alignment, style, and other presentation options.
6. To help the user move through the document:
   a. Click **Show Page Numbers** to show a page number below each page.
   b. Select a **Navigation** method:
      - **Thumbnails**: Show a list of the pages in the document in sequence below the document. The user clicks a thumbnail image to go to the associated page.

- **Indexer**: Show a series of buttons  below the document to represent each page in the document. The user clicks a button to go to the associated page.
- **None**: Provide no visual navigation. The user can swipe right or left to display the adjacent page. This manual method is always available.

c. Click **Show Previous/Next** to include arrow icons on each page to advance to the previous or next page in the document.

d. Enable or disable **Use arrow keys to navigate** to determine whether or not the arrow keys on the keyboard can be used to advance to the previous or next page in the document. If more than one document component is used on a page, all documents with this setting enabled will move when an arrow is depressed. To avoid confusion when more than one document component is used on a page, it's best to enable this setting on one primary document and disable it on all the rest, instead enabling **Show Previous/Next** icons to allow someone to navigate pages in a specific document.

Folder Lists

You can use a folder list to list the folders within a specified folder from your Oracle Content Management account.

If you use this component in conjunction with one or more file list or document manager components on the page, these components can automatically display the contents of a folder selected in the folder list.

To add a folder list component to a page:

1. Navigate to the page you want to edit and make sure that  is set to **Edit**.
2. Add the component to the page.
3. To edit the component and its appearance, click its menu icon  , and choose **Settings**.
4. Choose **Custom Settings** to set the default details about the content that's displayed.
5. Click **Select** next to **Folder Selection** to change the folder to use for display. You must select a folder other than document repository home page. Click **Back** when you are done.

> **Note:**
>
> The folder list grants all users downloader access. Users can view and download files regardless of their role. If a site visitor has privileges that are greater than those specified for the component, their individual privileges override those set on the component.

6. Choose from the following to set additional defaults for the content that's displayed.
   - **Choose Default Selection**: If you selected **Oracle Documents Folder**, select the folder (if any) to show as selected in the list.
   - **Show folder name header**: Select this to display the folder name in the heading of the embedded component.

- **Folder Sorting**: Choose how the items will be initially displayed, either alphabetically by name, or by when the items were last updated.
- **Show Subfolders**: Use the slide bar to limit the number of subfolders displayed.

7. Use the General tab to modify spacing, alignment, and other presentation options.
8. Use the Style tab to format the frame that contains the component with predefined styles or with your own custom choices.

File Lists

You can use a file list to provide a view of files from a specified folder in your Oracle Content Management account.

If you use this component in conjunction with one or more folder list components on the page, the file list component can automatically display the contents of a folder selected in the folder list. You can also configure the component to perform one or more actions when a user clicks a file in the list. For example, you can preview the selected file in a separate browser window, or in a “lightbox” overlay, or even in a Documents Manager component on the page.

To add a file list component to a page:

1. Navigate to the page you want to edit and make sure that  is set to **Edit**.
2. Add a component group to the page.
3. To edit the component and its appearance, click its menu icon  , and choose **Settings**.
4. Choose **Custom Settings** to set the default details about the content that’s displayed.
5. Click **Select** next to **Folder Selection** to change the folder to use for display. You must select a folder other than the document home page. Click **Back** when you are done.

> **Note:**
>
> The file list grants all users downloader access. Users can view and download files regardless of their role. If a site visitor has privileges that are greater than those specified for the component, their individual privileges override those set on the component.

6. Choose from the following to set additional defaults for the content that’s displayed.
   - **Display options**: Choose what details will be shown with the listed files.
     - **Folder name header**: show the folder name in the heading.
     - **File description**: show the file description, if one is given.
     - **File separators**: separate each file by a line.
     - **Download icon**: include a download icon so users can download the file if they want.

  - **Last updated**: show the date when the file was last updated.
  - **File size**: show the size of the file.
  - **Image**: show a thumbnail image of the contents of the file.
- **Triggers & Actions**: Choose if you'd like to auto-refresh the file list with contents of the folder selected in a Folder List component. You can also choose to enable the File Selected trigger when a file is selected. Use the Link tab to associate actions with the **File Selected** trigger.
- **File Sorting**: Choose how the items will be initially displayed, either alphabetically by name, or by when the items were last updated.
- **Show Files**: Choose if all files will be displayed or if you want to limit the number of files using the slider bar.

7. Use the General tab to modify spacing, alignment, and other presentation options.
8. Use the Style tab to format the frame that contains the component with predefined styles or with your own custom choices.
9. Use the Link tab to associate actions with the **File Selected** trigger. For example, to configure the file list component to preview a selected file in a lightbox overlay:
   a. Make sure you select **Activate trigger when file is selected** in the File List Settings window.
   b. In the Link tab of the file list settings, click the **File Selected** trigger.
   c. Click and drag the **Lightbox Preview** page action to the action list.
   d. In the File ID or URL field, choose **File Link**.

      File Link uses a reference link with downloader privileges so all site visitors can preview and optionally download the file. If you use **File ID**, a member link is used. A member link allows only registered users to preview and optionally download the selected file.

   When the user clicks on a file in the file list, the file preview opens over a dimmed and inactive version of the page.

Documents Manager

You can use documents manager to provide a view of your home page or files in Oracle Content Management.

To add a documents manager component to a page:

1. Navigate to the page you want to edit and make sure that  is set to **Edit**.
2. Add the component to the page.
3. To edit the component and its appearance, click its menu icon  , and choose **Settings**.
4. Choose **Custom Settings** to set the default details about the content that's displayed.
5. Click **Select** next to **Folder Selection** to change the folder to use for display.

   To select a folder, check the box next to the folder name. To open a folder, click the folder name. Click a folder name in the path to return to that folder or click **Navigate to Home** to return to the home folder. Click **Back** when you are done.
6. Click **Select Folder Access** and choose the access role to grant visitors.

Visitors will be able to view and work with the folder content based on their role and on the **Browsing Options** you enable in the next step.

- **Member Access**: Visitors will be able to use any features available to members of the folder, such as viewing conversations, annotations, or custom properties of items.
- **Viewer**: Viewers can look at files and folders, but can't change things.
- **Downloader**: Downloaders can also download files and save them to their own computer.
- **Contributor**: Contributors can also modify files, update files, upload new files, and delete files.

Consider the following when setting folder access:

- A site author can’t grant access to a folder that is greater than the access they themselves have. For example, if the author has downloader access to a folder, they can’t give contributor rights to site visitors.
- The privileges set on the folder in the component can augment the visitor’s privileges. For example if the visitor has viewer privileges (or no privileges) for the folder, the component can grant greater privileges based on the selected role. These enhanced privileges are valid only in the component itself.
- If a site visitor has privileges that are greater than those specified for the component, their individual privileges override those set on the component.
- Privileges granted on a folder apply to the folders and files nested in that folder.

7. Choose from the following to set additional defaults for the content that’s displayed.
   - **Layout**: Select an initial grid , list or compact list layout for the folders and files. Users can change the layout when they view the finished embedded component.
   - **Color Scheme**: Choose one of the available color schemes for the embedded folder listing.
   - **Sort Order**: Choose how the items will be initially displayed, either alphabetically by name, or by when the items were last updated.
   - **Browsing Options**: Choose what options will be available to users when they select an item. For example, you can choose to allow users to view files, download files, share or copy files and folders, or delete files. If you want to restrict what users can do with your files and folders, deselect an option in this list. If you chose Member Access in the previous step, you can elect to show a side pane where any conversations, annotations, or custom properties will be shown.
   - **Viewer Options**: Choose how users will view files. You can allow files to be viewed within the embedded frame or in another tab (or window depending on your browser settings). You can also hide or show thumbnails, and customize how videos are viewed.
   - **Show Zoom Controls**: Choose whether to show a slider bar or zoom controls in the embedded view of the folder.
   - **Viewer Fit Mode**: You can choose to have files be shown filling the page, filling the width of the page, or in the original size.

- **Triggers & Actions**: Choose to refresh a file list if it is also used with the folder list component.

8. Use the General tab to modify spacing, alignment, and other presentation options.
9. Use the Style tab to format the frame that contains the component with predefined styles or with your own custom choices.

After the documents manager has been added, viewers will see a view of the selected folder embedded in a frame on your site. Users can use the provided display options to change how folders and files are listed. If a user selects a file or folder, they can choose options provided by the menu bar or with the right-click menu and perform any action allowed by their role and the **Browsing Options** you specify.

If a conversation is associated with an item, you'll see  with the item. Click it to open the conversation pane where you can view annotations and comments. Independent conversations (those not associated with a folder) must be added using the conversation component.

If an item has custom properties, you can view the properties in a pane. Click **More** then select **Custom Properties** to open the property pane.

Project Library

You can use a project library to assemble folders from different locations in the repository without having to change the original folder or its location. You can also use the project library component to assign different permission levels to each folder to accommodate different project team roles.

For example, if you give a folder in the project library one of the visitor roles (viewer, downloader, or contributor), visitors can see and interact with the folder content with the privileges associated with the role. If you specify member access, only members will see it listed in the project library. Members interact with the folder content with the privileges specified for them on the original folder.
If you use this component in conjunction with one or more file list or document manager components on the page, these components can automatically display the contents of a folder selected in the project library.

To add a project library component to a page:

1. Navigate to the page you want to edit and make sure that  is set to **Edit**.
2. Add the component to the page.
3. To edit the component and its appearance, click its menu icon  , and choose **Settings**.
4. Choose **Custom Settings** to select conversations and to select display options.
5. Specify a title. The default title is Project Library. You can choose to display or hide the title by selecting or deselecting the **Title** display option below.
6. Select a **Color Scheme**.

> **Note:**
>
> If you use triggers and actions to associate a project library with a documents manager component, the color scheme you select for the project library also applies to the folder selected to show in the documents manager component. This is not the case if you associate the project library with folder list or file list components because those components do no support color schemes.

7. Click **Add** next to the list of folders to add an existing folder or to create a new folder.
   a. Navigate to and Select one or more folder or click **Create** to create a folder. Each folder you select in this window is added to the project library.

      You can use any of the options listed in the tool bar including view and sort options.

   b. Click **Back** when you are done.
8. To organize the list, select a folder and click **Move Up** or **Move Down** to change its location in the list order or click **Remove** to remove the folder from the list.
9. To specify access to a folder based on the user's role, select the folder from the list of folders, click **Select Folder Permission**, and choose the access role.
   - **Member Access**: Only registered users with permissions on the folder will see the folder listed in the project library. Members interact with the folder content with the privileges specified for them on the original folder.
   - **Viewer**: Viewers can look at files and folders, but can't change things.
   - **Downloader**: Downloaders can also download files and save them to their own computer.
   - **Contributor**: Contributors can also modify files, update files, upload new files, and delete files.

   Consider the following when setting folder access:
   - A site author can't grant access to a folder that is greater than the access they themselves have. For example, if the author has downloader access to a folder, they can't give contributor rights to site visitors.
   - The privileges set on the folder in the component can augment the visitor's privileges. For example if the visitor has viewer privileges (or no privileges) for the folder, the component can grant greater privileges based on the selected role. These enhanced privileges are valid only in the component itself.
   - If a site visitor has privileges that are greater than those specified for the component, their individual privileges override those set on the component.
   - Privileges granted on a folder apply to the folders and files nested in that folder.
10. For folders with **Member Access** permission, you can display conversations associated with folders or content by clicking **Show Conversation pane in Documents Manager**.

    If you select this option and configure a documents manager component on the page to display a selected folder, the user can click the conversation icon to

display any conversations associated with the folder. If you do not select this option, the conversation icon is not shown.

11. When you are done, close the window.
12. Use the General tab to modify spacing, alignment, and other presentation options.
13. Use the Style tab to format the frame that contains the component with predefined styles or with your own custom choices.
14. Use the Link tab to associate actions with the **Folder Selected** trigger. For example, if you also add a documents manager component to the page, you can use the documents manager component to display the contents of a folder selected in the project folder:
    a. In the Link tab of the project library settings, click the **Folder Selected** trigger.
    b. In the Configure Trigger Actions window, click Documents Manager.
    c. Click and drag the **Display documents** action to the action list.
    d. In the Folder ID or URL field, choose **Selected Folder**.

    When the user clicks on a folder in the project library, the folder content is displayed in the documents manager component on the page.

Social Components

Social components help your users stay connected and communicate.

- Social Bar
- Facebook Like and Recommend
- Twitter Share and Follow
- Conversation Component
- Conversation List

Social Bar

Easily add icons and links to popular social media, such as Facebook and Twitter. The social bar includes some targets by default, but you can add and remove items in the social bar.

To add and modify a social bar:

1. Navigate to the page you want to edit and make sure that  is set to **Edit**.
2. Add the component to the page. The social bar shows the icons included by default.
3. To add or change the icons in the social bar, click its menu icon  , choose **Settings**, and click **Icons** at the top of the **General** tab.

   To remove a social icon, click the x next to the name. To add an icon, the icon must be stored in the site repository or another repository to which you have access. You can also

use images that were shared with you or that you uploaded from a local or network file location.

4. To upload one or more icons from a local or network location:
   a. Click **Add Icons** at the top of the panel.
   b. Navigate to the location in the repository where you want to store the image or click  to add a new folder in the current location.
   c. Click  .
   d. Locate and select the image file or files and then click **Open**.

      The image files are uploaded to the current location in the repository.
5. Select one or more images in the repository and click **OK**.
   a. Locate and click the image or images you want to use.
   b. To link to the file in the repository, select **Use a reference to the original file instead of copying the file to the site**. If you do not select this option, a copy of the file is stored with the site and is referenced from the site. Linking to the original file avoids duplicating the content. The link allows site visitors to see the content even if the permissions on the file change or would otherwise restrict viewing.
   c. Click **OK**.

   The selected images are added to the list of images. Drag and drop the images to reorder them in the list. The default title of each image is the file name without the extension.
6. To change the target URL, title, description or other options for a particular image, click the image in the list and make the change.

   When you're done updating individual icons, click **Back** to specify general options.
7. Specify the size, spacing, orientation, and alignment for all of the icons in the social bar.

Facebook Like and Recommend

You can add a Facebook Like button to a page to let viewers easily like your site on Facebook.

1. Navigate to the page you want to edit and make sure that  is set to **Edit**.
2. Add the component to the page.
3. To edit the app and its appearance, click its menu icon  , and choose **Settings**.
4. Choose **Facebook Like Settings** or **Facebook Recommend Settings** to set the URL and to optionally add a **Share** button so you can easily post a link to your site on a Facebook page.
5. Use the General tab to modify spacing, alignment, and other presentation options.
6. Use the Style tab to format the frame that contains the app with predefined styles or with your own custom choices.

Twitter Share and Follow

You can add a Twitter Share button to a page to let viewers quickly share a link on a Twitter account.

1. Navigate to the page you want to edit and make sure that  is set to **Edit**.
2. Add the component to the page.
3. To edit the app and its appearance, click its menu icon  , and choose **Settings**.
4. Choose **Twitter Follow Settings** to set the Twitter user name and to choose whether to show the user name and to choose the size of the **Follow** button. Choose **Twitter Share Settings** to set the following defaults. Users can change the values when they use the button.
   - **Share URL**: The URL of the site a user can share.
   - **Tweet Text**: The text of a tweet about the page.
   - **Via** @: The Twitter user account used for the tweet.
   - **Recommend** @: The Twitter user account used for a Twitter recommendation.
   - **Hashtag** #: A hash tag you'd like used for the post.
   - **Count**: A display of the Share count, either vertical or horizontal.
   - **Large button**: Choose either a large or smaller button for the app.
5. Use the General tab to modify spacing, alignment, and other presentation options.
6. Use the Style tab to format the frame that contains the app with predefined styles or with your own custom choices.

Conversation Component

You can use a conversation to promote discussion about a topic directly from your site.

> **Note:**
>
> For a conversation to work on a site, the site must be a secure site limited to specified users or limited to users with the Oracle Content Management Users role. See Change Site Security.

To add a conversation component to a page:

1. Navigate to the page you want to edit and make sure that  is set to **Edit**.
2. Add the component to the page.
3. To edit the component and its appearance, click its menu icon  , and choose **Settings**.
4. Choose **Custom Settings** to select a conversation and to select a presentation color scheme.

5. Click **Select** next to **Choose Conversation** to select an existing conversation or to create a new one.
    a. Select a conversation from the list of available conversations or click **Create** to create and name a new conversation.

       The list contains all the conversations to which you have access.
       You can filter the list:

       - **All**: Shows all conversations except those that are marked as muted.
       - **Favorites**: Shows conversations that have been marked as favorites.
       - **Muted**: Shows conversations that have been muted and excluded from your list of conversations.
       - **Closed**: Shows all conversations that are marked as closed.

       You can also sort the list:

       - **Last Updated**: List conversations in order from most recently updated to least recently updated.
       - **Name**: Lists conversations in alphanumeric order from lowest to highest.
       - **Unread**: Lists unread conversations first in order of the conversation with the most unread comments to the conversation with the least number of unread comments.

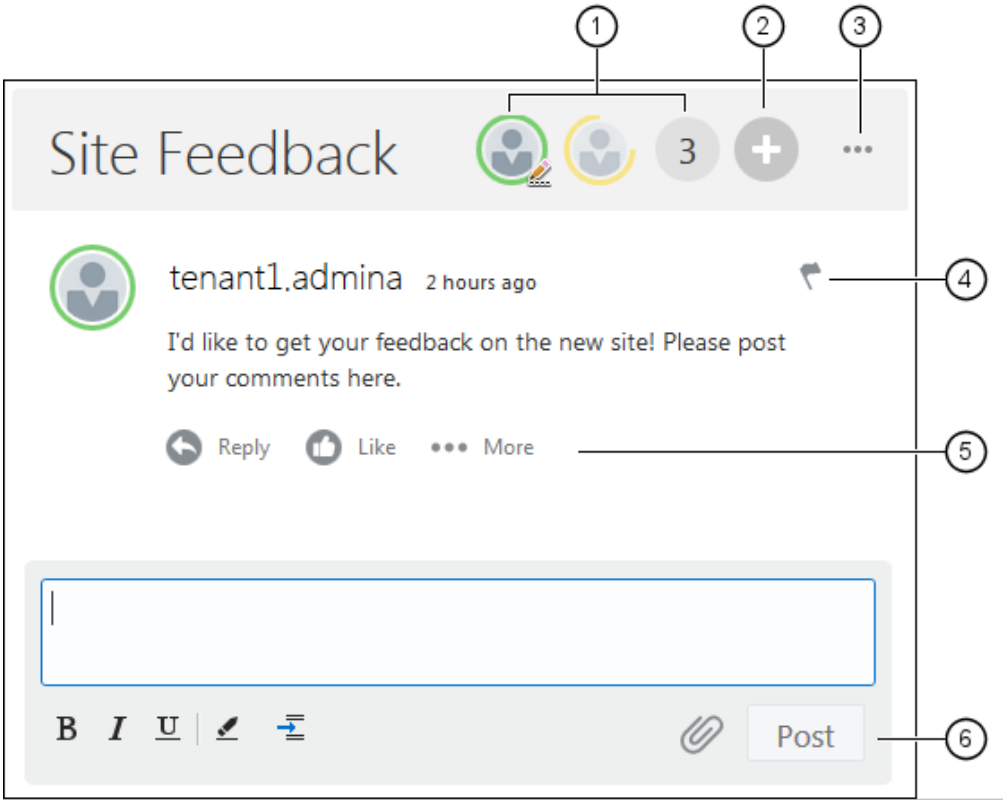
    b. Click **Back** when you are done.
    c. If you selected an independent conversation (one not based on a file or folder), you can set the access permission for people viewing the conversation.
    d. Select a **Color Scheme**.
    e. To use the conversation component in conjunction with a conversation list component on the page, select **Auto-refresh conversation based on selection in the conversation list component**.

       When a user selects a conversation from the list, it displays in the conversation component.

    f. When you are done, close the window
6. Use the General tab to modify spacing, alignment, and other presentation options.
7. Use the Style tab to format the frame that contains the component with predefined styles or with your own custom choices.

On the published site, visitors will see the conversation in a frame on your site. Users can navigate within the conversation to read and respond to comments.

> **Note:**
>
> If the visitor has not been explicitly added as a member of the conversation, they will have the ability to read and reply to comments, but they will not see items 1 through 4 in the image below.



- The **menu bar (1)** has information about individual participants in the conversation. Click a user icon to get status information and to see options for working with the user. The number icon shows the total number of participants. Click the icon to see the complete list.
- Click **Add Users (2)** to add users to the conversation.
- Click **More Options (3)** to see a list of options for working with the conversation. The list of options varies with the role of the user. For conversations associated with folders and files, The list of options also depends on the permissions set on the folder or file.
- Click **Flags (4)** to alert a particular user by assigning them a notification flag. The user is alerted by email as specified in their preferences.
- Use the **comment options (5)** to perform actions on a particular comment in the conversation such as replying to the comment, or liking, editing, or deleting the comment.
- When you add or edit a comment, use the **edit options (6)** to add or remove basic formatting such as bold or underlining, add an attachment, and post the comment to the conversation. Unless the site visitor is a member of the conversation, they won't be able to add attachments.

Conversation List

You can use a conversation to promote discussion about a topic directly from your site.

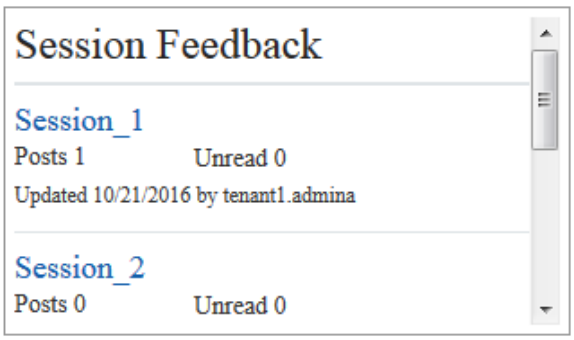
If you use this component in conjunction with one or more conversation list components on the page, the conversation list component can automatically display the contents of a conversation selected in the conversation list.

> **Note:**
>
> For a conversation or conversation list to work on a site, the site must be a secure site limited to specified users or limited to users with the Oracle Content Management Users role. See Change Site Security.

To add a conversation list component to a page:

1. Navigate to the page you want to edit and make sure that  is set to **Edit**.
2. Add the component to the page.
3. To edit the component and its appearance, click its menu icon  , and choose **Settings**.
4. Choose **Custom Settings** to select conversations and to select display options.
5. Specify a title. The default title is Conversation List. You can choose to display or hide the title by selecting or deselecting the **Title** display option below.
6. Click **Add** next to the list of conversations to add an existing conversation or to create a new one.
    a. Select one or more conversations from the list of available conversations or click **Create** to create and name a new conversation.
    b. Click **Back** when you are done.
    c. Choose the permissions allowed for those who view the conversation list.
7. To organize the conversation list, select a conversation and click **Move Up** or **Move Down** to change its location in the list order or click **Remove** to remove the conversation from the list.
8. Select a **Color Scheme** and chose additional **Display Options**. The display options you select are shown below the name of the conversation in the list.
9. When you are done, close the window.
10. Use the General tab to modify spacing, alignment, and other presentation options.
11. Use the Style tab to format the frame that contains the component with predefined styles or with your own custom choices.

The following image shows a conversation list titled *Session Feedback* with all display options selected.



Process Components

Process components let site users initiate and manage tasks for predefined processes.

> **Note:**
>
> To use process components, you must use Oracle Process Cloud Service release 17.1.3 or later.

- Process Start Form
- Process Task List
- Task Detail Form

Process Start Form

You can use a process start form to initiate a process defined with Oracle Process Cloud Service.

For a process start form to work on a site, the following must be true:

- The associated processes and process start forms must be defined with Oracle Process Cloud Service before you can display them with this component. See Developing Structured Processes in *Using Processes in Oracle Integration 3*.
- To use the process start form, the user must be assigned the role associated with the process swimlane that contains the start form.
- An administrator must set up the integration between Oracle Process Cloud Service and Oracle Content Management. See Integrate with Oracle Process Cloud Service in *Administering Oracle Content Management*. The integration between the two services requires SSO sign-ons, so both services must be in the same identity domain.

To add a process start form component to a page:

1. Navigate to the page you want to edit and make sure that  is set to **Edit**.
2. Add the component to the page.
3. To edit the component and its appearance, click its menu icon  , and choose **Settings**.
4. Choose **Custom Settings** to select a process start form and to set form defaults.
    a. If your site will be a public site, select a proxy service.
    b. Select a partition for the start form. You can use the Test partition to verify that the process is working as planned or the Production partition to deploy the process for general use.
    c. If you want to always use the version of the process that is selected as the default, select **Use default process version**. If you don't select this option you'll select a specific version, and if the process is updated, you'll need to update the selection in these settings.
    d. Select a process.

e. Select a start form. Forms have the following syntax: `process type:version:processname:start`. For example, `Basic Approval:1.0:Process:Start Basic Approval`.

   If a message at the top of the window states “No Process Cloud Service connection”, it’s possible that the integration between Oracle Process Cloud Service and Oracle Content Management is not configured. Contact your administrator.

   The process author must add you as an initiator of the process to see it in the list. The process author must add all site visitors as initiators of the process or the visitors will be able to complete the form, but not initiate the process.

f. Choose from the following to set additional defaults for the form.

   - **Form title**: Optionally replace the default form title with a title of your own.
   - **Submit button name**: Optionally rename the **Submit** button with a value you specify.
   - **Show Submit button**: Optionally show or hide the **Submit** button on the component (it’s shown by default). You might hide the component **Submit** button if a similar button is provided on the form itself.
   - **Submit confirmation**: Optionally replace the default confirmation message with a message of your own.
   - **Show submit confirmation**: Optionally show or hide the confirmation message when the **Submit** button is clicked. It’s shown by default and displays within the process start form component.
   - **Show Save button**: Optionally show or hide the **Save** button on the component (it’s hidden by default). You might show the component **Save** button if the associated process allows you to save your work and return to it later.
   - **Show Discard button**: Optionally show or hide the **Discard** button on the component (it’s hidden by default). You might show the **Discard** button if you want to allow the user to discard the form content and start over.
   - **Show Attachments**: Optionally show or hide an **Attachments** area on the form with ability to upload a file or files (it’s hidden by default). Files uploaded are stored as part of the process in Oracle Process Cloud Service.
   - **Customize Default Values**: You can set name and value pairs to prepopulate the start form. Click **Add Field**, then enter the field name and value. The name is one of the form fields, not a label used on the form, and the value is what is allowable for that field. To insert the current logged-in user data, use the special values `%%username%%` and `%%userid%%`.

5. Use the General tab to modify spacing, alignment, and other presentation options.
6. Use the Style tab to format the frame that contains the component with predefined styles or with your own custom choices.
7. Use the Links tab to assign actions to the triggers provided by the component:
   - **Start form submitted**: This trigger occurs when the user clicks the **Submit** button.
   - **Start form saved**: This trigger occurs when the user clicks the **Save** button.

- **Start form discarded**: This trigger occurs when the user clicks the **Discard** button.

Click the trigger to assign an action. For more information about assigning triggers and actions, see Use Triggers and Actions.

Process Task List

You can use a process task list to selectively list processes defined with Oracle Process Cloud Service. You can show detailed information for the tasks in the task list component, or use the process task list component in conjunction with a task detail component to simplify the list and to show detailed information for a selected task only.

For a process task list to work on a site, the following must be true:

- The associated processes must be defined with Oracle Process Cloud Service before you can display them with this component. See Developing Structured Processes in *Using Processes in Oracle Integration 3*.
- The site must be a secure site limited to specified users or limited to users with the Oracle Content Management User role. See Change Site Security.
- An administrator must set up the integration between Oracle Process Cloud Service and Oracle Content Management. See Integrate with Oracle Process Cloud Service in *Administering Oracle Content Management*. The integration between the two services requires SSO sign-ons, so both services must be in the same identity domain.

To add a process task list component to a page:

1. Navigate to the page you want to edit and make sure that  is set to **Edit**.
2. Add the component to the page.
3. To edit the component and its appearance, click its menu icon  , and choose **Settings**.
4. Choose **Custom Settings** to set display options and to filter the available tasks.
5. Choose from the following to set display options for the task list.

> **Note:**
>
> You can set display options for the component, but the presentation of the tasks themselves is determined by the design in Oracle Process Cloud Service.

- **Show Details**: Select to include task detail information in the task list itself. Use the process task list component in conjunction with a task detail component to simplify the list and show detailed information for a selected task only.
- **Show Search**: Select to include the search bar at the top of the task list. If you select **Show Search**, you have the additional option of selecting **Show Filter** to include the filter option in the search bar. Use the filter options below to set the default filter values.
- **Show Select All**: Select to allow the user to select all the displayed task for processing.
- **Page Size**: Adjust to specify the maximum number of tasks to display. If there are fewer tasks than the specified maximum, the component automatically adjusts to the

smaller number. If there are more tasks than the specified maximum, the component shows the maximum number and adds links to the additional page or pages.

6. Choose from the following to filter the available tasks in the list.

   The filters you apply determine the initial list of tasks. If you choose the **Show Search** and **Show Filter** options above, the user can adjust the settings to modify the filter for all settings except **From User**.

   - **Search Keywords**: Optionally specify one or more search terms. Search results include only those tasks with titles that include all of the specified keywords. If you selected **Show Search**, these keywords are displayed in the search bar and can be removed or modified by the user.
   - **Status**: Select one of the available Status values. The default is **Assigned**.
   - **Assignee**: Select one of the available Assignee values. The default is **Me and My Group All** which includes all tasks available to the user and their group, including those that are claimed but not workable.
   - **From User**: Optionally select one or more users from whom the task originates. Begin typing the user name to initiate a search of the available users on the associated Oracle Process Cloud Service. The values you specify are not displayed to the end-user and can't be removed. Users can add additional user names to further expand the list of users.
   - **Due Date**: Optionally select a due date that is On, Before, After, or Between a date selected from the calendar.
   - **Application**: Optionally select the name of a particular application. You can select one or more applications from the list of applications available to the user.

7. Use the General tab to modify spacing, alignment, and other presentation options.
8. Use the Style tab to format the frame that contains the component with predefined styles or with your own custom choices.

> **Note:**
>
> If you have a process task list and a task detail form on the page, the task detail component automatically displays the detail for a task selected from the list. For more information about triggers and actions, see Use Triggers and Actions.

Task Detail Form

You can use a task detail form in conjunction with a process task list to display details for a selected task. If you have a process task list and a process detail form on the page, the process task detail component automatically displays the detail for a task selected from the list.

For a task detail form to work on a site, the following must be true:

- The associated processes must be defined with Oracle Process Cloud Service before you can display them with this component. See Developing Structured Processes in *Using Processes in Oracle Integration 3*.

- The site must be a secure site limited to specified users or limited to users with the Oracle Content Management User role. See Change Site Security.
- An administrator must set up the integration between Oracle Process Cloud Service and Oracle Content Management. See Integrate with Oracle Process Cloud Service in *Administering Oracle Content Management*. The integration between the two services requires SSO sign-ons, so both services must be in the same identity domain.

To add a task detail component to a page:

1. Navigate to the page you want to edit and make sure that  is set to **Edit**.
2. Add the component to the page.
3. To edit the component and its appearance, click its menu icon  , and choose **Settings**.
4. Choose **Custom Settings** to set display options.
5. Choose from the following to set display options for the task detail form.

> **Note:**
>
> You can set display options for the component, but the presentation of the task detail is determined by the design in Oracle Process Cloud Service.

   - **Show Actions**: Select to display the actions available to the user, such as Approve, Reject, and so on.
   - **Show Save**: Select to display the **Save** button.
   - **Show Close**: Select to display the **Close** button.
   - **Show Attachment**: Select to display the **Attachment** section in the detail form.
   - **Show Comments**: Select to display the **Comments** section in the detail form.
   - **Show History**: Select to display the **History** section in the detail form.
   - **Show More Information**: Select to display the **More Information** section in the detail form.
   - **Show Links**: Select to display the **Links** section in the detail form.
6. Use the General tab to modify spacing, alignment, and other presentation options.

   By default, the task detail component expands to display all the specified detail. Click **Set Height** and adjust the height value to set a specific height.
7. Use the Style tab to format the frame that contains the component with predefined styles or with your own custom choices.
8. Use the Links tab to assign actions to the triggers provided by the component:
   - **Task details submitted**: This trigger occurs when the user clicks the **Submit** button.
   - **Task approved**: This trigger occurs when the user clicks the **Approve** button.
   - **Task rejected**: This trigger occurs when the user clicks the **Reject** button.
   - **Task closed**: This trigger occurs when the user clicks the **Close** button.
   - **Task saved**: This trigger occurs when the user clicks the **Save** button.

- **Task comment added**: This trigger occurs when the user adds comment text and clicks the **Post Comment** button.

Click the trigger to assign an action.

> **Note:**
>
> The process task list component does not support manual triggers or actions, however if you add a task detail component to the page, it automatically displays the detail for a task selected from the list. For more information about triggers and actions, see Use Triggers and Actions.

For information about using Oracle Process Cloud Service, see Getting Started with Process.

Content Items

For enterprise users, every site has a collection that contains digital assets and content items associated with the site. Content item components make it easy to add items to your site.

You can drag and drop digital assets and content items directly from the Content panel in the editor and the appropriate component is automatically used, whether for an digital asset image or for a structured content item.

Additional components selected from the Component panel let you dynamically display content items based on the content type.

- Content Item Component
- Content Placeholder
- Content List
- Dynamic List
- Content Search
- Recommendation

Content Item Component

As a enterprise user, you can use a content item component to help lay out a page and set up page interactions until you are ready to add the content items themselves.

When you drag and drop a content item from the Content panel onto a page, it automatically inserts a content item component to hold the item, unless the content item is a custom digital asset type. Dragging a custom digital asset onto a page causes the asset to be inserted as an image or video component. You need to remove it and add the content item component first, then drag a custom digital asset or standard content item from the Content panel onto the component for proper display.

To add a content item component to a page:

1. Navigate to the page you want to edit and make sure that  is set to **Edit**.
2. Add the component to the page.

3. To edit the component appearance, click its menu icon  , and choose **Settings**.
4. Use the General tab to modify spacing, alignment, and other presentation options.

   If a content item is assigned to the component, a thumbnail view of the content items is shown. If no content item is assigned yet, a placeholder image is displayed.
5. Choose from the following to set additional defaults for the content that's displayed.
   - **Version to use**: If you select **Use latest version of asset** and there is a newer, unpublished version of the content item, the newer version will be published automatically when the current site update is published. Unless specifically requested, all items will be the draft or latest versions.
   - **Item View**: Select the layout used to display the content item. The supplied **Default** layout shows all fields in the content item. If the content item has other, custom layouts designed for it, you can choose any available layout. If you select a custom layout that has **Add support for custom settings when used in Sites** enabled, then you can add custom data to the layout by clicking  entering data into the **Custom Data** field, and then click **Back** to return to the General tab.
   - **Page to display individual item**: If you have designated one or more pages as a detail pages, they are listed here. Choose a page to display detailed information when a user clicks the link on a the content item to view detailed information.

     > **Note:**
     >
     > If you do not create a detail page, the link to display details is not shown for the content item in the default layout.
6. Use the Style tab to format the frame that contains the component with predefined styles or with your own custom choices.
7. Click  to open the Select Asset panel and navigate to the content item you want to use. If you want to create a new content item and add it to the repository, click  and select **Create a New Content Item**.
8. If you are creating a new content item or editing one you've selected, make any desired changes to the content item and click **Save** to save the changes or if you are ready to publish, click **Publish**. The content item is saved as a new revision in the content repository, and if you chose to publish, then after the item is saved, the validation results are displayed. If all results are valid, click **Publish**.
9. Close the settings panel and click **Save** to save the changes to the update.

Content Placeholder

As a enterprise user, you can use a content placeholder component to dynamically display content items of one or more types.

For example, you can use a content item placeholder on a designated detail page and when a user clicks a link to get more detailed information for a particular content item, it will automatically load the detail view for the associated content item. For example, if there is a page with multiple articles, each with a headline and image, and the user clicks a particular article, the detail page displays the full article regardless of which article is chosen.

To add a content placeholder component to a page:

1. Navigate to the page you want to edit and make sure that  is set to **Edit**.
2. Add the component to the page.
3. To edit the component and its appearance, click its menu icon  , and choose **Settings**.
4. Use the General tab to modify spacing, alignment, and other presentation options.
5. Choose from the following to set additional defaults for the content that's displayed.
   - **Content Type**: Select one or more of the available content types. The content types are those of the content items in the site collection and include custom digital asset types.
   - **Item View**: Select the layout used to display the content item. The supplied **Default** layout shows all fields in the content item. If the content item has other, custom layouts designed for it, you can choose any available layout.
   - **Page to display individual item**: If you have designated one or more pages as a detail pages, they are listed here. Choose a page to display detailed information when a user clicks the link on a the content item to view detailed information.

   > **Note:**
   >
   > If you do not create a detail page, the link to display details is not shown for the content item in the default layout.
6. Use the Style tab to format the frame that contains the component with predefined styles or with your own custom choices.

Content List

As a enterprise user, you can use a content list component to dynamically display content items of a particular type.

For example, you can use a content item list on a designated detail page and when a user clicks a link to get more detailed information for a particular content item, it will automatically load the detail view for the associated content item. For example, if there is a page with multiple articles, each with a headline and image, and the user clicks a particular article, the detail page displays the full article regardless of which article is chosen.

To add a content list component to a page:

1. Navigate to the page you want to edit and make sure that  is set to **Edit**.
2. Add the component to the page.
3. To edit the component and its appearance, click its menu icon  , and choose **Settings**.
4. Use the General tab to modify spacing, alignment, and other presentation options.

5. Choose from the following to set additional defaults for the content that's displayed.
   - **Content Type**: Select one of the available content types. The content types are those of the content items in the site collection and includes custom digital asset types.
   - **Maximum Items** and **Start at Item**: Specify the maximum number of items to display and where the display will begin. Additional items are not displayed.
   - **Pagination**: Specify if you want to include pagination options with the list. If selected, you can then tailor the way the pagination is displayed with buttons or page numbers and different labels if needed.
   - **Date**: Use the options provided to select content items to display based the item creation date before, after, between, or within selected dates or date ranges.
   - **Categories**: Select categories to filter the list of content items to show only those items in a particular category or categories. Click **Select Categories** to open a slide-out panel you can use to select categories from the site repository. If you select more than one category in the same taxonomy, displayed items just have to have one of the selected categories assigned to them. If you select more than one category from different taxonomies, displayed items must have all of the selected categories assigned to them. For example, if you select categories for **Cities** and **Parks** from the **Destination** taxonomy, then items that are cities or parks are displayed. If you select the category **Cities** from the **Destination** taxonomy and the category **Europe** from the **Regions** taxonomy, then items that are cities in Europe are displayed.

     By default all children of the selected category will be available. If you prefer to limit your list to items in a specific category node, select the node and deselect **Include Child Categories**.
   - **Language**: Choose a language from those associated with the repository.
   - **Additional Query String** (optional): Specify additional query parameters to further refine the list of items displayed using a syntax similar to: `field.dept eq "Finance"`. See the table below for the list of available operators.
   - **Order By**: Sort the items by name or date in ascending or descending order. If the content type includes other date, number, or decimal fields, you can also sort by those fields.

     You can also select **Custom**, and then enter a custom expression. For example, to sort by department number in ascending order, you might enter `fields.deptno:asc`. You could also define a URL parameter for sorting. For example, you might enter `{{URLParams.sortBy}}`, then you could add the following parameter to the end of your site URL: `?sortBy=fields.deptno:asc`.
   - **Item View**: Select the layout used to display the content item. The supplied **Default** layout shows all fields in the content item. If the content item has other, custom layouts designed for it, you can choose any available layout. If you select a custom layout that has **Add support for custom settings when used in Sites** enabled, then you can add custom data to the layout by clicking  entering data into the **Custom Data** field, and then click **Back** to return to the General tab.
   - **Page to display individual item**: If you have designated one or more pages as a detail pages, they are listed here. Choose a page to display detailed information when a user clicks the link on a the content item to view detailed information.

> **Note:**
>
> If you do not create a detail page, the link to display details is not shown for the content item in the default layout.

- **List View**: Select a section layout to arrange the items. This list can include custom layouts. The following layouts are provided:
  - **Horizontal**: Arranges the items one after the other in a horizontal line.
  - **Vertical**: Arranges the items one after the other in a vertical line. This is the default if no layout is selected.
  - **Two Columns**: Arranges the items by twos in multiple rows.
  - **Three Columns**: Arranges the items by threes in multiple rows.

  Click the right arrow next to the selected layout to modify settings for the layout.
- **Empty List View**: Select how an empty list should be displayed.
- **Options**: choose if there should be an auto-refresh on the search query specified in the Content Search component.
- **Spacing**: Choose how the items will be displayed.

6. Use the Style tab to format the frame that contains the component with predefined styles or with your own custom choices.

Table 10-1 Query Operators

Operator	Example	Supported Data Types	Description

eq	`?q=name eq "John"`
	`?q=type eq "DigitalAsset"`
	`?q=type eq "Employee" and fields.DOB eq "1994/09/26T16:23:45.208"`
	`?q=type eq "Employee" and fields.DOB eq "1994/09/26T16:23:45.208"`
		text, reference, number, decimal, boolean, datetime	Equals operator (eq) matches the exact value supplied in the query. This operator is not applicable to multivalued data types. The value provided with this operator is not case-sensitive except for standard fields. This operator considers even special characters in the value.

Table 10-1 (Cont.) Query Operators

Operator	Example	Supported Data Types	Description
co	`?q=(type eq "Employee" AND name co "john alex")` `?q=(type eq "Car" AND fields.features co "manual")`	text, reference, number, decimal, datetime, largetext	Contains operator (co) matches every word given in the criteria. The words are formed by splitting the value by special characters. It gives the results that have at least one of the words (in this example, john or alex or both). This operator does not consider special characters in the value while searching. This operator does not perform a search on stop words. Refer to Apache Lucene documentation to know more about stop words. This operator is applicable to *text, largetext* in case of single-valued attributes, whereas for multivalued attributes, it is applicable to *text, reference, number, decimal, datetime, largetext*. To understand the possible datetime formats, refer to the *Supported date/datetime formats (24Hrs)* table below. The value provided with this operator is not case-sensitive.
sw	`?q=type eq "Employee" AND name sw "Joh"` `?q=type eq "Employee" AND fields.city sw "Los"`	text	Starts with operator (sw) matches only the initial character values given in the field condition. This operator is not applicable to multivalued data types. The value provided with this operator is not case-sensitive.
ge	`?q=(type eq "Employee" AND fields.age ge "40")` `?q=type eq "DigitalAsset" AND updatedDate ge "20171026"`	number, decimal, datetime	Greater than or equal to operator (ge) matches only numeric and datetime values. To understand the possible datetime formats, refer to the *Supported date/datetime formats (24Hrs)* table below. This operator is not applicable to multivalued data types.
le	`?q=(type eq "Employee" AND fields.weight le "60.6")`	number, decimal, datetime	Less than or equal to operator (le) matches only numeric and datetime values. To understand the possible datetime formats, refer to the *Supported date/datetime formats (24Hrs)* table below. This operator is not applicable to multivalued data types.

Table 10-1 (Cont.) Query Operators

Operator	Example	Supported Data Types	Description
gt	`?q=(type eq "Employee" AND fields.age gt "20")`	number, decimal, datetime	Greater than operator (gt) matches only numeric and datetime values. To understand the possible datetime formats, refer to the *Supported date/ datetime formats (24Hrs)* table below. To understand the possible datetime formats, refer to the *Supported date/datetime formats (24Hrs)* table below. This operator is not applicable to multivalued data types.
lt	`?q=(type eq "Employee" AND fields.age lt "20")` `?q=type eq "Employee" AND createdDate lt "1994/09/26T16:23:45.208"`	number, decimal, datetime	Less than operator (lt) matches only numeric and datetime values. To understand the possible datetime formats, please refer to the section: **Supported date/datetime formats (24Hrs)**. This operator is not applicable to multivalued data types.

Table 10-1 (Cont.) Query Operators

Operator	Example	Supported Data Types	Description

mt	`?q=(type eq "Car" AND fields.review mt "petrol 20KMPL")`
	`?q=(type eq "Employee" AND name mt "Jo?n")`
	`?q=(type eq "Employee" AND name mt "Jo*")`
	`?q=(type eq "Employee" AND fields.role mt "senior*")`
		text, largetext	Phrase query or proximity search (matches) operator (mt) enables you to find words that are within a specific distance to one another. Results are sorted by best match. It is useful for searching content items when values given in the criteria "petrol 20kmpl" need to discover actual content that may contain "*petrol* fuel mileage runs *20KMPL* in the speed way".
			Matches operator also can use a wildcard within the given value and supports both single-character and multiple-character wildcard searches within a single value. Use ? for a single-character wildcard and * for multiple characters. Both "John" and "Joan" can be searched by "Jo?n" for a single character and "Jo*" for multiple characters.
			This operator is applicable to both single-valued and multivalued data types. This operator does not perform a search on stop words. Refer to Apache Lucene documentation to know more about stop words. The value provided with this operator is not case-sensitive.

sm	`?q=(type eq "Employee" And fields.city sm "Rome")`	text, largetext	Similarity query operator. This operator allows searching for values that sound like specified criteria - also called fuzzy search, which uses by default a maximum of two edits to match the result. "Rome" is similar to "Dome". This operator is applicable to both single-valued and multivalued data types. The value provided with this operator is not case-sensitive.
AND	`?q=(type eq "Employee" AND name eq "John" AND fields.age ge "40")`	N/A	AND operator, can be used to put an AND condition between multiple query conditions. This takes precedence over OR.

Table 10-1 (Cont.) Query Operators

Operator	Example	Supported Data Types	Description
OR	`type eq "Employee" AND name eq "John" OR fields.age ge "40"`	N/A	OR operator can be used to put an OR condition between multiple query conditions.
( )	`?q=type eq "Employee" AND (name eq "John" AND fields.age ge "40")` `?q=type eq "Employee" AND ( (name eq "John" AND fields.age ge "40") OR fields.weight ge 60)`	N/A	Parentheses, enclosing operator to group the conditions in the criteria. This takes highest precedence, followed by AND, and then by OR.

Dynamic List

As a enterprise user, you can use a dynamic list component to dynamically display content items of multiple content types.

1. Navigate to the page you want to edit and make sure that  is set to **Edit**.
2. Add the dynamic list component to the page.
3. To edit the component and its appearance, click its menu icon  , and choose **Settings**.
4. Use the General tab to modify spacing, alignment, and other presentation options.
5. Enter the query expression to query across multiple types. A multi-type search is specified by using curly braces {} to enclose type predicates in the query 'q' parameter, signifying a type scope. For example: `{type eq "Article" and fields.author eq "Smith"} or {type eq "Event" and fields.sponsor eq "Oracle"}`. For a list of query operators, examples, supported data types, and descriptions, see the topic on Content List components.
6. Enter any optional parameters:
   - The 'orderBy' parameter can also have a typed section and takes the form of:

     ```
     {<type-name-1>:fields.<field-name-1>[:asc|:desc]};
     {<type-name-2>:fields.<field-name-2>[:asc|:desc]};
     ```

     This parameter only supports one orderBy field per type scope.
   - The 'fields' parameter can be 'typed' as well and takes the form of:

     ```
     {<type-name-1>:fields.<field-name-1>[,fields.<field-name-2>]};
     {<type-name-2>:fields.<field-name-1>[,fields.<field-name-2>]};
     ```
7. Choose from the following to set additional defaults for the content that's displayed.

- **Maximum Items** and **Start at Item**: Specify the maximum number of items to display and where the display will begin. Additional items are not displayed.
- **Pagination**: Specify if you want to include pagination options with the list. If selected, you can then tailor the way the pagination is displayed with buttons or page numbers and different labels if needed.
- **Item View**: Select the layout used to display the content item. The supplied **Default** layout shows all fields in the content item. If the content item has other, custom layouts designed for it, you can choose any available layout. If you select a custom layout that has **Add support for custom settings when used in Sites** enabled, then you can add custom data to the layout by clicking  entering data into the **Custom Data** field, and then click **Back** to return to the General tab.
- **Page to display individual item**: If you have designated one or more pages as a detail pages, they are listed here. Choose a page to display detailed information when a user clicks the link on a the content item to view detailed information.

  > **Note:**
  >
  > If you do not create a detail page, the link to display details is not shown for the content item in the default layout.

- **List View**: Select a section layout to arrange the items. This list can include custom layouts. The following layouts are provided:
  - **Horizontal**: Arranges the items one after the other in a horizontal line.
  - **Vertical**: Arranges the items one after the other in a vertical line. This is the default if no layout is selected.
  - **Two Columns**: Arranges the items by twos in multiple rows.
  - **Three Columns**: Arranges the items by threes in multiple rows.

  Click the right arrow next to the selected layout to modify settings for the layout.
- **Empty List View**: Select how an empty list should be displayed.
- **Spacing**: Choose how the items will be displayed.

8. Use the Style tab to format the frame that contains the component with predefined styles or with your own custom choices.

Content Search

As an enterprise user, you can use a content search component and specify the actions returned by the search.

You can insert a customized search bar to change or refresh the content that's displayed on the page or choose another action, such as opening a search results page or displaying an alert.

To add a content search component to a page:

1. Navigate to the page you want to edit and make sure that  is set to **Edit**.
2. Add the content search component to the page.

3. To edit the component and its appearance, click its menu icon  , and choose **Settings**.
4. Use the General tab to modify placeholder text, spacing, alignment, and other presentation options.
5. Use the Style tab to format the frame that contains the component with predefined styles or with your own custom choices for the font, border, background color, and so on.
6. If you want to use the search component to refresh data on the current page, add a Content List component to the page. Go to **Settings** and pick the content type and any other query information, such as number of items to display, whether to paginate the results, or lazy load on scroll (loading content as the page is scrolled). You can now go to View mode and try out the search.
7. If you want to use the search component to refresh data on the current page and you have more than one Content List component on the page, you'll need to turn off the auto-query on all content lists except for the one that will display results. Go to the Content List settings and uncheck **Auto-refresh on search query in Content Search component.**
8. If you want to have more than one content search component on a page (with each search component driving a certain content list for results), you must uncheck all auto-refresh options in the Content List components and use Trigger/Actions to associate the content search component to its respective content list.
9. For the search component, use the Link tab to associate actions with the component. Choose an option from the Select Link Type drop-down list. Select **Trigger Actions** then click **On search query** to see the available triggers or to create a new trigger. Find the content list you want to search, expand it, and drag **Search Content** to the actions list. Under **Search** select **Search String**. The content list you chose is selected in the **Do this action in** list.
10. You can also use the content search component to pass the query to a search results page. On the Link tab of the search component, select **Search Page** to display a search result page. You can use the default page or link to a search result page that you created. Select the page from the Page drop-down list and choose the display actions for the page. You can tailor the results to a specific content type, open the results in a new window, and so on.

The following pointers can help you design an effective search results page:

- Create a page and designate it as a search page. By default, it's marked as hidden, but you can change the setting in the page properties.
- Edit the page and add a content list to the page. You can edit the settings for the content list, specifying a content type. Or the content type can be specified from the search component link settings (above). If you use the link settings, that lets you use a search results page that can show results from different content types, depending on which component is used to initiate the search.
- To edit the display, change the settings, such as choosing lazy load pagination (because the page will probably be used exclusively for search results).
- On the search results page, you can also put in a content search component. This will echo the search string used to launch the page, letting a user refine the search if needed.

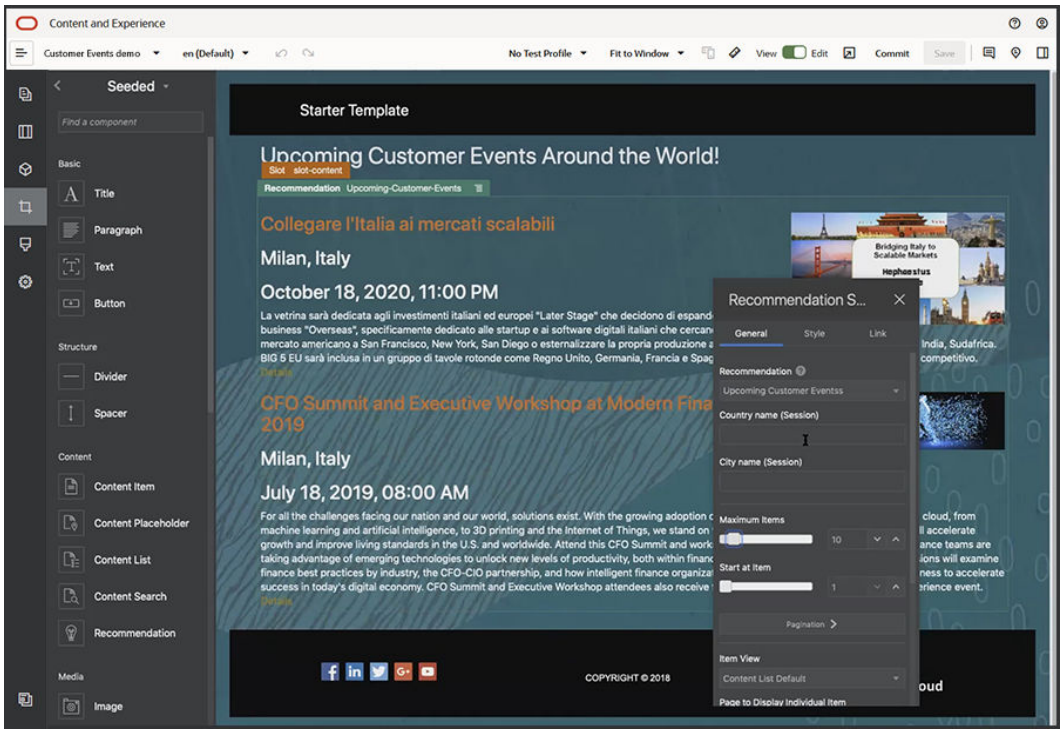
Recommendation

As a enterprise user, you can use a **Recommendation** component to provide personalized experiences for website visitors by showing assets based on location or areas of interest. When a repository contributor creates a recommendation they define a set of rules that find assets matching audience attributes such as the geolocation information of a site visitor. For example, site visitors with European IP addresses may first see event announcements for Europe on the site home page, while site visitors from North America will see events in the United States and Canada.

Like digital assets, recommendations are associated with a repository and can be edited and moved through a workflow for review and publishing by anyone who is a content contributor to the repository. However, even if you don't have contributor rights, you can still view and test recommendations to see how they work before using them in a site or headless experience.

To add a recommendation to a page:

1. Click **Sites** in the side menu, select the site you want to add the recommendation to, then choose **Open** in the right-click menu or click  in the actions bar.
2. Toggle the site to **Edit** and select the update to use or create a new one.
3. Select **Components** from the side menu.
4. Open **Seeded**.
5. Click-and-drag **Recommendation** under the Content section to place it onto the page.
6. Select **Settings** from the Recommendation menu ().



7. In the Recommendation settings **General** tab, select the recommendation to use. The page will refresh and display the recommended assets.

8. Optionally, enter default values of the audience attributes used by this recommendation by clicking the arrow next to the selected recommendation. When finished adding values, click **Back**.
9. Change any additional properties you want, such as which version to use, maximum items to display or default content layout. When done, close the recommendation settings and click **Save**.

> **Note:**
>
> If you have created test profiles with predefined audience attribute values, select a profile from the test profile menu in the menu bar to preview how the recommendation responds to those inputs. Test profile values would override the default values you set in the component settings.

10. If your recommendation uses the Current date (System) in its rules and you want to test the recommendation as if it were a different date, select **System Date** from the test profile menu and select the date to test with.

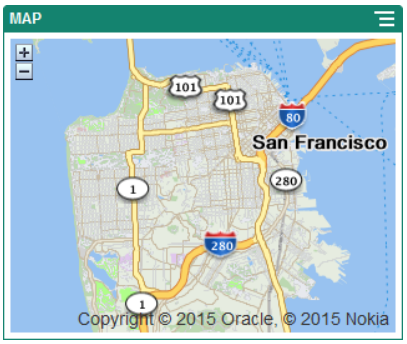
Other Components

Let's look at some components that combine different type of content.

- Maps
- Headlines
- Articles
- Images with Text
- Component Groups
- Using Cobrowse on a Page
- Oracle Intelligent Advisor
- Oracle Visual Builder

Maps

Add a map to your site to let users interactively explore the area around a location.



To add a map to the page:

1. Navigate to the page you want to edit and make sure that  is set to **Edit**.
2. Add the component to the page. The Oracle map component shows a default location.
3. To change the location for the map and to adjust its display properties, click its menu icon  and choose **Settings**.
4. Enter the starting **Location** for the map. You can use an address, zip code, or comma-separated latitude and longitude (for example, 40.5,-57.6)

   If you use an address, make sure you provide enough information to match a single location. If the address matches more than one location, the map remains blank.
5. Specify an initial **Zoom** level for the map. You can optionally allow the user to adjust the zoom level using a mouse, track pad, or on-screen controls.
6. Specify any style, alignment, and display options:
   - **Zoom**: Enable on-screen and device options to allow the user to adjust the zoom level of the map.
   - **Pan**: Enable on-screen and device options to allow the user to move the area of focus of the map.
   - **Display Marker**: Mark the starting location with a pin icon.
   - **Scale**: Include an indicator that shows the scale of the current zoom level.
   - **Overview**: Include an inset map that shows the current view in a larger context.

Headlines

You can add a headline to call attention to a certain spot on the site with title text and a supporting image and paragraph.

1. Navigate to the page you want to edit and make sure that  is set to **Edit**.
2. Add the component to the page.
3. Click in the different areas of the component to add text. You add content to the main headline and the paragraph below the main headline. The text takes on the formatting of the default style for the component. Press Enter to add additional lines of text.
4. If you want to change the default formatting for any portion of text, select the text that you want to format, then select any of the options in the formatting tool bar, such as font, color, or alignment.
5. To remove the formatting applied with these options, select the text and click  .
6. To edit the component and its appearance, click its menu icon  , and choose **Settings**.
7. Choose **Components** to set the details about the content that's displayed.
   - **Image**:
     - Click **Select** to use an image in the headline. Select an image from the site repository, an image from a documents folder that's been shared with you, or upload an image to a documents folder. Locate and select the image you want to

use. If you don't see any digital assets, click , and change the collection filter to **All**. If you selected an image from a documents folder, you can link to the file rather than copying it to the site. To link to the file, select **Use a reference to the original file instead of copying the file to the site**. If you don't select this option, a copy of the file is stored with the site and is referenced from the site. Linking to the original file avoids duplicating the content. The link allows site visitors to see the content even if the permissions on the file change or would otherwise restrict viewing. Click **OK**. If you selected a digital asset, you can select a specific rendition. If you don't select a rendition, the original size will be used. If you want the latest version of the asset to be published when the site is published, select **Use latest version of asset**.

To edit the image, click  , and edit the image with any of the following actions:

* To crop the image, click  **Crop**. Select one of the predefined image ratios in the cropping toolbar, or drag the cropping handles on the image as desired. When you're satisfied, in the cropping toolbar, click **Crop**.
* To rotate or flip the image, click  **Rotate**. In the rotation toolbar, enter a custom rotation degree, use the buttons to rotate the image left or right, or select whether to flip the image horizontally or vertically.
* To add a watermark to the image, click  **Watermark**. Add text to the image, changing the text size, style, color, and opacity as desired with the watermark tools.
* To change the image format, click  **Options**, then select a new format from the **Format** drop-down list.
* To change the background color, click  **Options**, then select an option from the **Background Color** drop-down menu.
* If you're editing a .jpg or a .webp (available on Google Chrome browsers), you can change the image quality to create a smaller file size. Click  **Options**, then enter the new percentage in the **Quality** box.
* To undo or redo your change, click  or  . To remove all changes you made, click **Reset**.
* To change the magnification of the image, use the zoom controls ().

– **Title**: Enter text you'd like to have displayed in a tooltip.
– **Alternate Text**: Enter the alternate text that will be displayed for accessibility purposes.
– **Caption**: Enter a caption that appears under the image.

  - **Alignment, Width, and Spacing**: Change the layout of the image as needed.
- **Title**: Change the spacing for the headline title.
- **Paragraph**: Change the spacing for the text that appears under the headline.

8. Use the General tab to modify spacing, alignment, and other presentation options.
9. Use the Style tab to add formatting around the text and to customize the background color, fonts, and borders.

Articles

You can add an article to your site, which combines the components of a headline, paragraph, and image in one easy-to-use component.

1. Navigate to the page you want to edit and make sure that  is set to **Edit**.
2. Add the component to the page.
3. Click in the different areas of the component to add text. You add content to the article headline and a sub-head below that. Then you can add the text of your article below the sub-heading. All text takes on the formatting of the default style for the component. Press Enter to add additional lines of text.
4. If you want to change the default formatting for any portion of text, select the text that you want to format, then select any of the options in the formatting tool bar, such as font, color, or alignment.
5. To remove the formatting applied with these options, select the text and click  .
6. To edit the component and its appearance, click its menu icon  , and choose **Settings**.
7. Choose **Components** to set the details about the content that's displayed.
   - **Image**:
     - Click **Select** to use an image in the article. Select an image from the site repository, an image from a documents folder that's been shared with you, or upload an image to a documents folder. Locate and select the image you want to use. If you don't see any digital assets, click , and change the collection filter to **All**. If you selected an image from a documents folder, you can link to the file rather than copying it to the site. To link to the file, select **Use a reference to the original file instead of copying the file to the site**. If you don't select this option, a copy of the file is stored with the site and is referenced from the site. Linking to the original file avoids duplicating the content. The link allows site visitors to see the content even if the permissions on the file change or would otherwise restrict viewing. Click **OK**. If you selected a digital asset, you can select a specific rendition. If you don't select a rendition, the original size will be used. If you want the latest version of the asset to be published when the site is published, select **Use latest version of asset**.

       To edit the image, click  , and edit the image with any of the following actions:

       * To crop the image, click  **Crop**. Select one of the predefined image ratios in the cropping toolbar, or drag the cropping handles on the image as desired. When you're satisfied, in the cropping toolbar, click **Crop**.

* To rotate or flip the image, click **Rotate**. In the rotation toolbar, enter a custom rotation degree, use the buttons to rotate the image left or right, or select whether to flip the image horizontally or vertically.
* To add a watermark to the image, click **Watermark**. Add text to the image, changing the text size, style, color, and opacity as desired with the watermark tools.
* To change the image format, click **Options**, then select a new format from the **Format** drop-down list.
* To change the background color, click **Options**, then select an option from the **Background Color** drop-down menu.
* If you're editing a .jpg or a .webp (available on Google Chrome browsers), you can change the image quality to create a smaller file size. Click **Options**, then enter the new percentage in the **Quality** box.
* To undo or redo your change, click or . To remove all changes you made, click **Reset**.
* To change the magnification of the image, use the zoom controls ( ).

– **Title**: Enter text you'd like to have displayed in a tooltip.
– **Alternate Text**: Enter the alternate text that will be displayed for accessibility purposes.
– **Caption**: Enter a caption that appears under the image.
– **Alignment, Width, and Spacing**: Change the layout of the image as needed.

- **Title**: Change the spacing for the article title.
- **Paragraph** (sub heading) and **Paragraph**: Change the spacing for the text that appears under the article title.

8. Use the General tab to modify spacing, alignment, and other presentation options for the component.
9. Use the Style tab to add formatting around the text and to customize the background color, fonts, and borders.

Images with Text

You can add use the image and text component which combines the components of a paragraph and an image in one easy-to-use component.

1. Navigate to the page you want to edit and make sure that is set to **Edit**.
2. Add the component to the page.

3. Click in the paragraph area of the component to add text. All text takes on the formatting of the default style for the component. Press Enter to add additional lines of text.
4. If you want to change the default formatting for any portion of text, select the text that you want to format, then select any of the options in the formatting tool bar, such as font, color, or alignment.
5. To remove the formatting applied with these options, select the text and click  .
6. To edit the component and its appearance, click its menu icon  , and choose **Settings**.
7. Choose **Components** to set the details about the content that's displayed.
   - **Image**:
     – Click **Select** to use an image. Select an image from the site repository, an image from a documents folder that's been shared with you, or upload an image to a documents folder. Locate and select the image you want to use. If you don't see any digital assets, click , and change the collection filter to **All**. If you selected an image from a documents folder, you can link to the file rather than copying it to the site. To link to the file, select **Use a reference to the original file instead of copying the file to the site**. If you don't select this option, a copy of the file is stored with the site and is referenced from the site. Linking to the original file avoids duplicating the content. The link allows site visitors to see the content even if the permissions on the file change or would otherwise restrict viewing. Click **OK**. If you selected a digital asset, you can select a specific rendition. If you don't select a rendition, the original size will be used. If you want the latest version of the asset to be published when the site is published, select **Use latest version of asset**.

       To edit the image, click  , and edit the image with any of the following actions:

       * To crop the image, click  **Crop**. Select one of the predefined image ratios in the cropping toolbar, or drag the cropping handles on the image as desired. When you're satisfied, in the cropping toolbar, click **Crop**.
       * To rotate or flip the image, click  **Rotate**. In the rotation toolbar, enter a custom rotation degree, use the buttons to rotate the image left or right, or select whether to flip the image horizontally or vertically.
       * To add a watermark to the image, click  **Watermark**. Add text to the image, changing the text size, style, color, and opacity as desired with the watermark tools.
       * To change the image format, click  **Options**, then select a new format from the **Format** drop-down list.
       * To change the background color, click  **Options**, then select an option from the **Background Color** drop-down menu.
       * If you're editing a .jpg or a .webp (available on Google Chrome browsers), you can change the image quality to create a smaller file size. Click  **Options**, then enter the new percentage in the **Quality** box.

    * To undo or redo your change, click  or  . To remove all changes you made, click **Reset**.
    * To change the magnification of the image, use the zoom controls ().
  - **Title**: Enter text you'd like to have displayed in a tooltip.
  - **Alternate Text**: Enter the alternate text that will be displayed for accessibility purposes.
  - **Caption**: Enter a caption that appears under the image.
  - **Alignment, Width, and Spacing**: Change the layout of the image as needed.
- **Paragraph**: Change the spacing of the text.

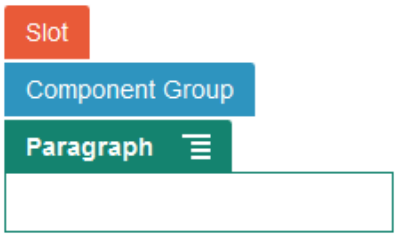
8. Use the General tab to modify spacing, alignment, and other presentation options for the component.
9. Use the Style tab to add formatting around the text and to customize the background color, fonts, and borders.

Component Groups

You can combine one or more components to create a component group that you can name and reuse.

When you save a component group, the component group is saved as a custom component with the name you give it and it then appears in the list of custom components in the editor.

1. Navigate to the page you want to edit and make sure that  is set to **Edit**.
2. Add a component group to the page. The component group is identified by  .
3. Drag and drop one or more components into the frame of the custom component.
4. Position and size the components within the component group in the same way you do for components in a slot.
5. To edit a component and its appearance, click its menu icon  , and choose **Settings**. If you click the component name instead of the menu icon, you can see and select the menu icon for the component group (or slot):



The Settings tab for the component group lets you specify the position of the component group, a background image, and other settings that apply to the entire component group.

6. When you are ready to save your changes to the component group, click the component group, click its menu icon, and choose **Save**.

   a. In the dialog, enter a name for the component group. You can use letters, numbers, underscores (_), and hyphens (-). If you enter a space, it's automatically replaced with a hyphen.

      If this is a new component group, you cannot use the name of an existing custom component.
      If you added an existing component group to the page, modified the component group, and then tried to save your changes, you are given the option to provide a name to create a new component group to select **Overwrite the existing component group** to update the existing component group with your changes.

   b. Click **Save**.

      The component group is saved with the specified name as a custom component. It appears in its own folder in the component manager and in the list of custom components in the editor if you are the owner or someone has shared the component with you. You can share the component group as you would any custom component.

Copying Component Groups to the Clipboard

For easy access to component groups used while editing a site update, you can copy component groups to the component clipboard.

1. Find and select the component group you want to copy.
2. Click the component menu of the component group, and select **Copy to Clipboard**. The component group is now listed in the clipboard section of the component panel.

> **Note:**
>
> Each time you copy a component group to the clipboard, it is listed by a new name, whether you have edited the component group or not.

3. Drag-and-drop the component group from the clipboard section of the component panel onto any page.

> **Note:**
>
> The component clipboard is associated with an update. Items in the clipboard of one update are not available in different update. The clipboard is emptied when closing Site Builder.

Using Cobrowse on a Page

The Oracle Cobrowse Cloud Service is a collaboration tool that lets you share screens or initiate a cobrowsing session with another person. For example, you may want to include this on an order form so a representative can view a customer's screen while the customer is placing an order.

To use this feature, it must first be enabled for a site. There are two types of launcher scripts that can be enabled: one that uses a customized button (Launch Point 2) and one that uses the default Cobrowse button (Launch Point 1). You determine which kind of launcher will be used when you enable the feature for your site and add the necessary script. See Enable Cobrowse Integration for details.

After Cobrowse is enabled for a site, any page can be configured to allow cobrowsing.

1. Navigate to the page you want to edit and make sure that  is set to **Edit**.
2. Click  . Select the checkbox in the Cobrowse section.
3. Click **Close**.

If a Launch Point 1 script is enabled, when you next view the page, you'll see the default Cobrowse button appear or after pressing a hot key, if configured.

If a Launch Point 2 script is enabled, you need to add the custom button to the page.

1. Make sure that  is set to **Edit**.
2. Add the component to the page. The Cobrowse Launcher component is listed in the Integration section of the components.
3. To edit the component and its appearance, click its menu icon  , and choose **Settings**. You can adjust the label for the button, its appearance, size, and alignment. Use the Style tab to add formatting around the text and to customize the background color, fonts, and borders. For more advanced styling, edit or add style classes in the `design.json` and `design.css` files in the theme designs folder of current site template. The style class prefix is `scs-cobrowse`.

After your site is published, your site visitors can use the Cobrowse button to initiate a session with a representative from your organization. The visitor clicks the Cobrowse button and is given a secure session ID. The visitor relays the ID via a phone call to a representative from your organization who has access to the Cobrowse Agent Console. The agent uses the console to initiate a session, which continues until either the visitor or the agent terminates it. When using a Launch Point 2 setting, the Launcher component has a fixed ID of cec-start-cobrowse. Use that ID in the Cobrowse console.

Notes on Usage

When using Cobrowse Instant Mode (ICB), video or embedded iFrames are not viewable on a page unless the iFrame content is enabled with the same Cobrowse site ID. As a result, some Oracle Content Management components are not rendered in the Cobrowse agent console in ICB mode. Use Cobrowse Advanced Mode (ACB) to render the following components:

- Video
- Youtube
- Document Manager
- Facebook Like
- Twitter Follow
- Twitter Share

- Facebook Recommend
- Conversation

A custom component that uses an iFrame to get content also does not render in Instant Mode.

See Cobrowse Overview in the *Cobrowse Deployment and Use Guide* for more information about Oracle Cobrowse Cloud Service. See Enable Cobrowse Integration for details about using Cobrowse with a secure site or one under development.

Oracle Intelligent Advisor

Oracle Intelligent Advisor (formerly Oracle Policy Automation) is used to implement online interview scenarios, such as feedback for troubleshooting or eligibility assessments for services. It delivers advice across channels by capturing rules in natural language Microsoft Word and Excel documents, then building interactive customer service experiences called interviews around those rules.

Before you can use the Intelligent Advisor feature, it must be configured and enabled. Your service administrator enables the feature for your service, including adding the host name, URL, user name, and password for the Intelligent Advisor Hub in use. The integration between the two services requires SSO sign-ons, so both services must be in the same identity domain. See Integrate with Intelligent Advisor in *Integrating and Extending Oracle Content Management*.

On the Intelligent Advisor side, interviews must be created and stored on the host site. In addition, your Oracle Content Management service must be authorized for use by the Intelligent Advisor host.

Once Intelligent Advisor is configured and enabled, you can add an Intelligent Advisor component to a page on your site.

1. Navigate to the page you want to edit and make sure that  is set to **Edit**.
2. Add the component to the page. The component appears in the Integration section of the Component list.
3. To edit the component and its appearance, click its menu icon  , and choose **Settings**. You can adjust the label for the component, its appearance, size, and alignment. Use the Style tab to use the default style associated with the interview from the Intelligent Advisor host. For more advanced styling, edit or add style classes in the design.css files in the theme designs folder of the current site template. The style class prefix is scs-opainterview-.

After your site is published, your site visitors will see the interactive interview that is chosen in the Intelligent Advisor component. For more details about Intelligent Advisor , see the Intelligent Advisor Documentation Library.

Oracle Visual Builder

Oracle Visual Builder is a hosted environment for your application development infrastructure. It provides an open-source standards-based solution to develop, collaborate on, and deploy applications within Oracle Cloud.

Initial Steps

Before you use Oracle Visual Builder, it must be enabled and configured. Your service administrator enables the feature for your service, including adding the host name for where the apps are created and stored. See Integrate with Oracle Visual Builder in *Integrating and Extending Oracle Content Management*. The integration between the two services requires SSO, so both services must be in the same identity domain.

In Oracle Visual Builder, the following must be done before this feature can be used with Oracle Content Management:

- Cross-Origin Resource Sharing (CORS) must be enabled on the Oracle Visual Cloud Service site.
- Apps must be created, made available for embedding, and must be configured for use with Oracle Content Management.
- Web applications must be created and made available for embedding in an iframe. The Sites SDK must be imported and referenced in the web applications. A page URL parameter called “id” must be added to the web applications.

Create Oracle Visual Builder Components

After the integration is enabled and apps and web applications are created and ready to use, you must create a new component for each app that you want to add to your site pages.

1. In Oracle Visual Builder, get the URL for the published web application. Click the live project that includes the web application, then click the web application. Copy the URL from the address bar.
2. In Oracle Content Management, click **Developer** and then click **View All Components**. Registered remote components and layouts are displayed.
3. Click **Create** and choose **Create Visual Builder Component**.
4. Enter a name for the component. You can’t use a name used by another component or layout.

   You can use letters, numbers, underscores (_), and hyphens (-). If you enter a space, it’s automatically replaced with an underscore.

   Don’t use the following names for site templates, themes, components, sites, or site pages: authsite, content, pages, scstemplate_*, _comps, _components, _compsdelivery, _idcservice , _sitescloud, _sitescloudddelivery, _themes, _themesdelivery. Although you can use the following names for site pages, don’t use them for site templates, themes, components, or sites: documents, sites.
5. Optionally, enter a description for the component.
6. Paste the URL to the live web application or live classic application.

7. Click **Create**. After the component is created, the name appears in the list of components. You can explore the folders and files that make up the component or layout by clicking the component name in the list .
8. To select an icon other than the default icon assigned to the component:
    a. Select the component from the list.
    b. Click **Properties**
    c. Click the **Component Logo** tab.
    d. Click a logo from the gallery of logos and then click **Done**.

Adding the Component to a Site Page

Now you can add the component for the app to one of your site pages. You can add the component to either a public or a secure page.

1. Navigate to the page you want to edit and make sure that  is set to **Edit**.
2. Add the component to the page. The component appears in the Custom section.
3. To edit the component and its appearance, click its menu icon  , and choose **Settings**. You can adjust the label for the component, its appearance, size, and alignment. Use the Style tab to use the default style associated with the component from the Oracle Visual Builder host. For more advanced styling, edit or add style classes in the design.css files in the theme designs folder of current site template. The style class prefix is scs-component.

You can view the component in preview mode while editing your site. After your site is published, your site visitors see the Oracle Visual Builder app chosen for use with that component, running in an iFrame on the page.

Part III
Publishing and Managing Sites

This part details how to work with, secure, improve, and publish sites. It includes the following chapters:

11

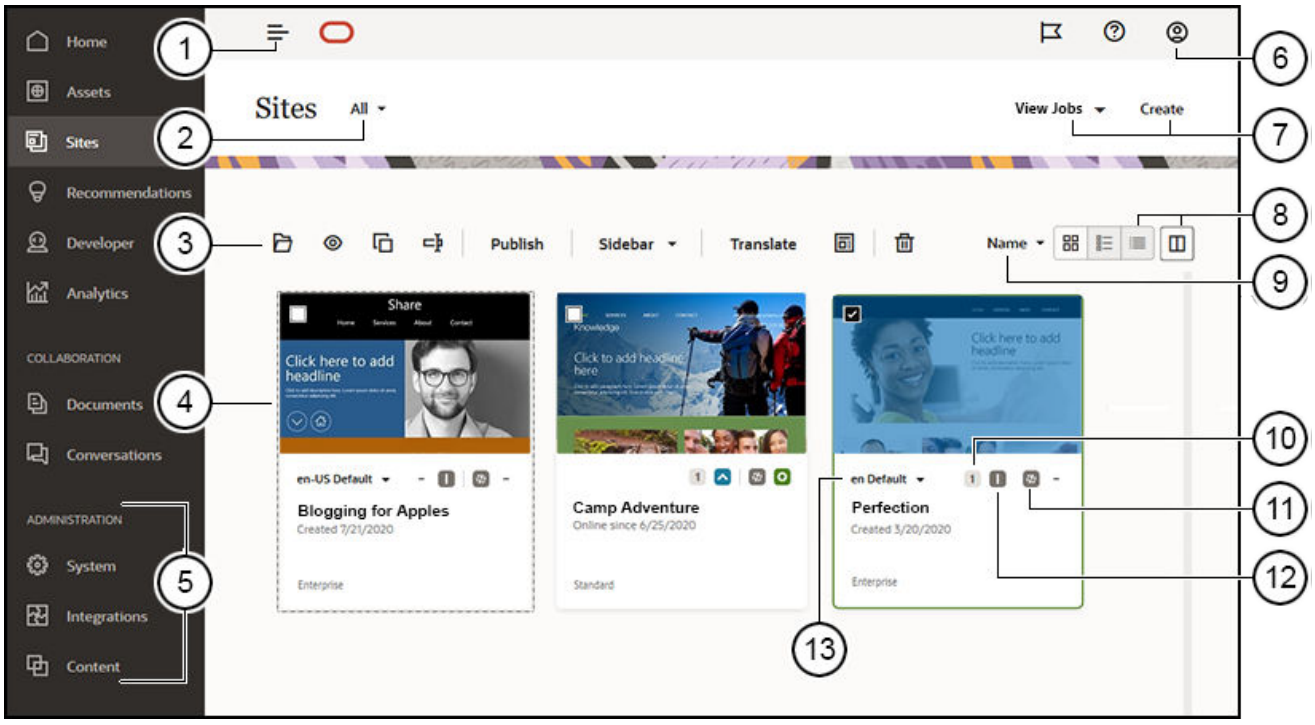
Manage Sites

To change a site layout or content, create and open an update in the editor. To create and manage the site itself and the properties of the site, use the options in the site manager.

- Get to Know the Sites Page
- Manage Sites and Site Settings
- Bring a Site Online or Take It Offline
- Change the Site Description, Logo, or Properties
- Set Search Engine Properties
- Customize Site Settings
- Enable Cobrowse Integration
- Add Analytics Tracking

Get to Know the Sites Page

The Sites page is your gateway to working with web sites.



Callout	Description
1	The **panel toggle** hides and shows the navigation menu.
2	The **filter menu** lets you filter the sites list to see all sites you have access to or subset of them. If site governance is enabled, you can view pending site requests.
3	After you select a site in the list, the **actions bar** is available. Use the options on the actions bar to open, view, change the status of, or rename your sites, and perform other tasks. The options that are shown depend on your role for the selected site. For example, if you created the site, then you have the Manager role for that site and can perform all the tasks listed. But if someone has shared a site with you and assigned you the Viewer role, you can view site properties, but you can't change anything. The actions bar options also depend on the current status of the site. For example, **Compile** is only available for sites that are published and the **Compile site after publish** option is enabled in the **Static Content** tab of the site properties panel. Also, the **Rename** and **Delete** options are available only if the site is offline. **Tip:** Looking for a shortcut? Right-click a site to open the context menu and choose an option.
4	The **sites list** shows all the sites you own or that were shared with you. The list includes the site name, the number of updates, if the site is secure, and the status of the site (online, offline, or pending updates).
5	The **Administration menu** is shown if you're logged in as a content administrator. Options here let you change service settings, set up integrations, and set up repositories and other asset-related features.
6	The **user menu** has options for setting your preferences, giving feedback, accessing help, and signing out. Click the user picture to display menu options.
7	Click **Create** to start the process of creating a website. Before you can create a site, your administrator must enable site creation and provide one or more templates. If you don't see the **Create** option on the sites page or templates on the template page, contact your service administrator. Click **View Jobs** and select an option to see a listing of translation, publishing, site export or site import jobs.
8	Click the **view icon** to display items in a list view, a grid view, or a table view. In table view, click the **table preferences icon** () to organize columns and choose which columns are displayed in the table. Click the **column sort icon** (- shown in table view) in a column header to sort the table display.
9	Use the **sort options** to change the display order for the sites.
10	Click the **update number** to edit an existing update for the site or to create a new update for the site. If the site doesn't have any updates, you can create a new update by opening the site and changing Site Builder to edit mode.
11	Click the **Offline**, **Online**, or **Republish** icons to change the status of the site or publish changes that have been committed but not published to the live site.
12	The **security** icons show whether a site requires log in or not.
13	Use the **language list** to see the languages specified in the site's associated localization policy. To preview a localized version, select the language, and open the site.

Manage Sites and Site Settings

Site management includes creating and editing the site and managing the content used in the site. You use the *Site Builder* to create, copy, and delete site pages and page content. Use the *folder and file manager* to create, copy, share, and delete entire sites.

Use options on the action bar or right-click menu on the Sites page and in Site Builder to perform the following tasks.

Task	Description
Filter sites view	To filter the sites list, select one of the following options in the filter menu: • **All** - Shows all sites you have access to. If you're a site administrator and site governance is enabled, you have access to all sites in your environment. • **Owned by You**—Shows all sites you own. • **Shared with You**—Shows all sites that have been shared with you. • **Online**—Shows sites that are online (live). • **Offline**—Shows sites that are offline. • **Requests** (available only when site governance is enabled)—Shows pending site requests. If you're a site administrator, you see all site requests in your system, otherwise you see only those site requests you've submitted. • **Trash**—Shows deleted sites that are owned by you. If you are a site administrator and governance has been enabled, then it also shows all deleted sites.
Create a site	To create a site, click **Create**. When you create a site, you begin with a template. A template has everything you need to get started with your site, including the site code framework, a default site with sample pages and content, a theme with styling, resources such as images, and even custom components. See Create Sites. You can see whether site governance is enabled by looking at the filter menu on the **Sites** page. If you see the **Requests** option, site governance is enabled. See Understand Site Governance.
View and manage site requests	To view pending site requests (available only when site governance is enabled), in the filter menu, choose **Requests**. If you're a site administrator, you see all site requests in your system, otherwise you see only those site requests you've submitted. See Manage Site Requests.
Edit a site	To edit the site, choose **Open** in the right-click menu or click  in the actions bar. See Edit Sites.
Discuss or annotate a site	You can discuss and annotate sites in Site Builder. Select a site and choose **Open** in the right-click menu or click  in the actions bar. To discuss a site, click  to open the conversation panel. To annotate a site, click  . See Get to Know the Site Builder Page.
View a site	To see what the site will look like when it's live (online), select a site and choose **View** in the right-click menu or click  in the actions bar.
Publish a site	To publish a site, select it and choose **Publish** in the right-click menu or actions bar. See Publish Site Changes.
Republish	If you've previously published a site, but an associated item or policy has change (for example, the associated localization policy has been updated to include more languages), you can **Republish** the site to refresh those changes. Select a site and choose **Republish** in the right-click menu or in the actions bar.

Task	Description
Unpublish	If you've previously published a site and you want to remove the site files from the hosting location in Oracle Cloud, select a site and choose **Unpublish** in the right-click menu or in the actions bar. A site must be offline in order to unpublish it.
Compile	If your service administrator has enabeld a compilation service, compiling a site can speed up site delivery by generating and using static HTML files to render your site rather than rendering your site dynamically. You automate site compilation by enabling **Compile site after publish** on the **Static Delivery** tab of the site properties. If enabled, your site will render dynamically until compilation is finished and the static files will be served. If you select a site that you've previously published and compiled and your site is set to **Compile site after publish**, a **Compile** option is available on the action bar. This allows you to compile a site without having to republish it. This is useful if you've added or republished a few assets that you would like to update in the compiled HTML without having to republish an entire site.
Bring a site online or take it offline	The site must have been published before you can bring it online. To bring a site online or take it offline, select a site and choose **Bring Online** or **Take Offline** in the right-click menu. You can also click  in the site tile to bring a site online, or click  in the site tile to bring a site offline. When a site is *online*, users can view it with a web browser at a designated address (URL). When a site is *offline*, the site isn't available for public viewing; you can view the site only in Oracle Content Management. See Bring a Site Online or Take It Offline.
Rename a site	To rename a site, you must be the site owner or have the manager role for the site. If so, right-click it, and choose **Rename** in the right-click menu or click  in the actions bar. Enter a name for the site that's different from any existing site name on the same server. You can't rename a site if it's online. To take the site offline, you must be the site owner or have the manager role for the site.
Copy a site	You can copy a site to get a head start in building a site. Everything from the original site, including the theme, all outstanding updates, the pages, the page content, and all other assets such as images, are copied to the new site under the new name you provide. Your new site is offline and ready for editing. **Note:** If site governance is enabled, the site might need to be approved before the copy is created. If you are copying a site that uses content from multiple repositories, you need to do so using the Oracle Content Management Toolkit. See Develop with Oracle Content Management Toolkit and Use the cec Command-Line Utility. Select the site you want to copy, and choose **Copy** in the right-click menu or click  in the actions bar. See Copy Sites.

Task	Description
Create a template from a site	If you have a site that you want to use as a starting point for other sites, you can create a template from that site. Select the site, and choose **Create Template** in the right-click menu or click in the actions bar. See Create a Site Template from a Site. **Note:** If you are creating a template from a site that uses content from multiple repositories, you need to do so using the Oracle Content Management Toolkit. See Develop with Oracle Content Management Toolkit and Use the cec Command-Line Utility.
Delete or restore a site	If you have the appropriate permissions, you can delete a site and its contents. When you delete a site, everything in the site folder, including all outstanding updates, site pages, page content, and assets such as images that you added to pages are put in the trash. You can delete or restore a site if you created the site (you're the site owner) or if someone shared a site with you and gave you the Contributor or Manager role. You can't delete a site if it's online. To take the site offline, you must be the site owner or have the manager role for the site. You also can't delete a site that has been published. You must first unpublish the content. If the site has content from multiple repositories, you need to unpublish associated content in each repository. To delete a site, select it, and choose **Delete** from the right-click menu or click in the actions bar. You're prompted to move the site and all site updates to the trash. A deleted site stays in the trash until: • You restore the site. • You permanently delete the site. • Your trash quota is reached. • The trash is automatically emptied based on the interval set by your service administrator. The default value is 90 days. To restore a site, in the Sites menu, choose **Trash**. Select the site from the list, and choose **Restore**.

Task	Description
Add members to a site	If your administrator enabled sharing, you can share your site with other Oracle Content Management users and allow them to view, modify, or manage the site in Oracle Content Management based on the permission you give them. Anyone who can access your service is considered a member. When you share a site, you assign a role that defines what the member can or can't do with your site. Members can only use your site according to the role you assign to them. Select the site you want to share and choose **Members** from the right-click menu or on the actions bar to open the members panel. Once the members panel is open, click **Add Members**. Enter one or more user names or email addresses, If your service administrator has enabled inclusion of external users, then you can choose to allow them access to the folder or not. When adding an external user, enter the complete email address of the external user you are adding. Their full email address must be used each time you add them as a member. If there isn't already a user with that email address, Oracle Content Management will automatically provision a new external user. Once an external user has been invited and successfully provisioned, you can add them to other folders just as you would internal users. Assign one of these roles: • **Viewer**: Viewers can view the site in the editor, but can't change anything. • **Downloader**: For a site, the downloader role provides the same privileges as the viewer role and the user can create a new template from the site. • **Contributor**: Same as viewer, and can also edit the site, delete site pages, and delete the site if it's offline. • **Manager**: Same as contributor, and can also add users and assign their roles, publish changes to an online site, and switch the site online and offline. The creator of a site (the owner) is automatically assigned the manager role. Note that external users can collaborate on objects to which they're given access, but they cannot be assigned the manager role. If they are a member of a group that is assigned a manager role, they will have only contributor rights. This safely limits their ability to create and remove content. **Note:** If you are sharing a site that has access to multiple repositories, only the default repository is shared. Any additional repository will need to be shared separately. External users may not be a part of your organization, but can be added to a site, provided your service administrator allows it, their email address is registered with your service, and **Allow external users folder access and membership** is not disabled when adding members to the site. External users can collaborate on objects to which they're given access, but they cannot be assigned the manager role. This safely limits their ability to create and remove content. If they are a member of a group added to a site as a manager, any external users in the group will have only contributor rights.
Change site properties (description, logo, etc.)	To quickly access files in a site's document folder, or to change the site description, add a logo, allow the site to be embedded, add a vanity URL, and view valuable information about a site, including the site URL, site owner, and other details, select the site and choose **Properties** from the right-click menu or on the actions bar. See Change the Site Description, Logo, or Properties.

Task	Description
Review site activity	Select the site and choose **Properties** from the right-click menu or on the actions bar. then select the **Activity** tab to access logs of recent site activity.
Translate a site	You can translate an enterprise site into multiple languages, if the associated repository allows for it, by exporting the site files, translating them, and then importing the translated files. See Translate a Site.
Set search engine properties	In Site Builder, you can provide keywords and text to help search engines identify the content of the site. See Set Search Engine Properties
Customize site settings (favorite icons, controller files, etc.)	In Site Builder, you can specify the site icons used with different browsers and platforms, or add a controller file to handle link behavior. See Customize Site Settings
Enable cobrowsing	In Site Builder, you can enable the Oracle Cobrowse Cloud Service, a collaboration tool that lets you share screens or initiate a cobrowsing session with another person. For example, you may want to include this on an order form so a representative can view a customer's screen while the customer is placing an order. See Enable Cobrowse Integration
Add analytics tracking to a site	In Site Builder, you can add a snippet of JavaScript tracking code to a site for web analytics tracking, making it easier to integrate with external analytics providers like Google, Adobe, or Oracle Infinity. See Add Analytics Tracking.
Secure a live site	When you publish a site and make it available online, it's publicly available to anyone. However, if you're the site owner or have the manager role, you can restrict live sites to be available to registered users or a subset of users. To change the status of a site, you must be the site owner or have the manager role. Select the site you want to secure and choose **Properties** from the right-click menu or on the actions bar, then click the **Site Security** tab. See Change Site Security.
Add a site redirect	If the site URL changes, a redirect forwards one URL (source) to another URL (target). This helps to preserve user bookmarks and search engine rankings. See Add Site Redirects.

Change the Site Description, Logo, or Properties

Most site properties are set when you create or update a site. When you view properties, you can get valuable information about a site, including the site URL, site owner, and other details.

To change the site description, its logo, add a vanity site, change the properties, or access files in the site's associated Documents folder:

1. Select the site and choose **Properties** from the right-click menu or click **Properties** on the actions bar.
2. To change the optional description of the site, click the **Properties** tab and enter or change the description.
3. You can't change the site URL directly, but you can select and copy the site URL and then paste it into documents, presentations, and email to provide access to the site.

   If the site is online, click  to go directly to the online site.
4. To allow the site to be used as an embedded site, toggle **Yes** next to **Embeddable Site**.
5. You can optionally enable the **Optimize Dynamic Page Delivery** switch. When enabled, the initial HTML markup of dynamic site pages is generated on the server instead of the

client. This improves performance by reducing the number of network calls and work required by browsers to display site pages.

> **Note:**
>
> When this switch is enabled, any custom controller.html associated with the site is not used.

6. To access files in the site's documents folder, click **View files** next to **Site Folder**. As a site manager, you may need access to modify site files. For example, you may want to modify how a site renders by editing the controller.html file.
7. To change the site image, click the **Site Logo** tab, then click **Change**. Find and select the image to use. It must be a .png, .jpeg, or .jpg file and have a 4:3 (rectangular) ratio. The best size is 300x225 pixels because smaller images might be distorted and bigger ones might affect performance.
8. When finished, click **Close**.

Disable Default Page Extension

When new pages are created in a site, the default page URL is constructed with an .html extension. You can disable this so that new pages are constructed without any extension.

1. Click **Sites** in the side navigation.
2. Select the site and choose **Properties** from the right-click menu or click **Properties** on the actions bar.
3. Click **SEO** in the properties sidebar.
4. Click **Include '.html' extension** to toggle the default extension off.
5. Click **Save** to close the panel.

Changing this value will only affect new pages. URLs of existing pages within the site are not changed.

Enable Prerender Service for Search Engine Optimization

You can enable sites built with Oracle Content Management to prerender static HTML pages to respond to search crawler requests.

If the prerender service is enabled and a request comes in from a search crawler, the page is searched for in cache.

- If the page is found in cache and is less than 15 days old, the request is served the cached page.
- If the page in cache is older than 15 days or is not found in cache, a new version is fetched from Oracle Content Management, the cache is updated, and the new prerendered page is served.
- If the page is not found in cache or in Oracle Content Management, an error page is returned.

Expired pages on public sites or new sites that are not yet cached are rendered once a day. If a page is already cached, it does not update the cache. When the prerender service is enabled, the prerendered cache is updated hourly for sites that have been updated and republished. If you are making changes to site pages and want to test how those changes affect optimization, you can update the prerendered cache manually in site properties.

1. Click **Sites** in the side navigation.
2. Select the site and choose **Properties** from the right-click menu or click **Properties** on the actions bar.
3. Click **SEO** in the properties sidebar.
4. The date and time of the last cache refresh is listed. To update the cache, click **Refresh Now**. A progress bar displays the caching status.

> **Note:**
>
> If a page is set to be hidden from search engines in the site SEO settings, then that page is not prerendered or cached.

To enable the prerender service in Oracle Content Management:

1. Click **System** in the Administration area of the side navigation.
2. Select **SEO for Sites** in the system setting menu.
3. Click **Enabled**.
4. Define any additional user-agents required.

Table 11-1 Hardcoded User-agents in Oracle Content Management

User-agent	User-agent	User-agent	User-agent
baiduspider	facebookexternalhit	twitterbot	rogerbot
linkedinbot	embedly	quora link preview	showyoubot
outbrain	pinterest/0.	developers.google.com/+/web/snippet	slackbot
vkShare	W3C_Validator	redditbot	Applebot
WhatsApp	flipboard	tumblr	bitlybot
SkypeUriPreview	nuzzel	Discordbot	Google Page Speed
Qwantify	pinterestbot		

Table 11-2 Additional Preconfigured User-agents in the PrerenderUserAgents property of the config.cfg file

User-agent	User-agent	User-agent	User-agent
AddSearchBot	AdIdxBot	AdsBot-Google	AdsBot-Google-Mobile-Apps
AppEngine-Google	Baidu-YunGuanCe	Bingbot	BingPreview
DuckDuckBot	DuckDuckGo-Favicons-Bot	endeca webcrawler	Exabot

Table 11-2 (Cont.) Additional Preconfigured User-agents in the PrerenderUserAgents property of the config.cfg file

User-agent	User-agent	User-agent	User-agent
Facebot	Feedfetcher-Google	FeedValidator	Fetch
FlipboardProxy	Google Favicon	Google Web Preview	Google-Adwords-Instant
Googlebot	Googlebot-Image	Googlebot-Mobile	Googlebot-News
Googlebot-Video	Google-PhysicalWeb	Google-Structured-Data-Testing-Tool	HubSpot,ia_archiver
Mediapartners-Google	MSNBot	NetcraftSurveyAgent	nutch
Oracle Secure Enterprise Search	pinterest.com	PIs-Google	SEOkicks
seoscanners	Siteimprove.com	Slurp	Sogou web spider
VSE/1.0	W3C_CSS_Validator	W3C_I18n-Checker	W3C_Unicorn
W3C-checklink	W3C-mobileOK	Y!J	Yahoo Link Preview
Yahoo! Slurp	Yandex	YandexBot	YunGuanCe

Specify and Configure Vanity Domains

Oracle Content Management supports two methods of providing access to sites on vanity domains. Site vanity domains allow specific sites to be accessed on a vanity domain. Instance vanity domains allow all sites in your Oracle Content Management instance to be accessed on a vanity domain.

For example, a standard Oracle Content Management site URL might be `https://myinstance.cec.ocp.oraclecloud.com/site/MyCustomerSite/`, a site vanity domain could be configured as `https://www.mycustomer.com`. This is easier to remember, potentially useful for branding, and generally simpler to use when accessing a site. A custom path may also be used to access a site configured with a site vanity domains, such as `https://www.example.com/store/`.

Instance vanity domains can be used to access all sites in your instance by using the site path in addition to the selected vanity domain. Both site and instance vanity domains require several setup steps. Determine which method makes sense for your need and follow the instructions.

Use a Content Delivery Network

Both site and instance vanity domains require the use of a Content Delivery Network (CDN). A CDN is a platform of globally distributed servers meant to improve the performance and security of web sites. A CDN minimizes the distance between users and servers while optimizing the end-to-end performance of requests for content. While the primary goal of a CDN is to improve user experience, a CDN can also be used to alter requests in transit so that what the visitor sees is clean even if the process behind the scenes is not.

To support the hosting of an Oracle Content Management site on a vanity domain you will need to work with the CDN to configure it to handle all requests from the configured vanity domain, route them back to Oracle Content Management properly, and make alterations to the requests so they are handled properly and securely by Oracle Content Management.

Use Oracle Content Management's Content Delivery Network

Oracle Content Management provides CDN services to enable several vanity domain setups. By using Oracle Content Management's CDN services you can host site level vanity domains, including bare domains and custom paths, as well as instance level vanity domains, both standard and short paths, and friendly management URLs.

> **Note:**
>
> Oracle Content Management's built-in Content Delivery Network isn't supported in private instances.

To set these up, sign in to your Oracle Support account and see knowledge base article How to Use a Custom Hostname with Oracle Content Management. Work with the support teams to complete the process.

Oracle Content Management controls the CDN and associated security policies so access to full CDN capabilities and customizations are not possible. If you require additional control over the CDN delivery layer you must acquire your own CDN services and configure them to your needs.

Manage a Domain with a Domain Name System

Any domain can be used as a vanity domain for an Oracle Content Management site. You must control any domain used as the vanity domain before configuring it for use with an Oracle Content Management site.

Due to the limitations of domain name systems (DNS), using a root domain, such as `example.com`, without a www or another subdomain, such as `store.example.com`, may not be possible. Check with your DNS and CDN providers to determine if using a canonical name (CNAME) record for your root domain is possible.

Because DNS functions at the domain level and not the path level, for Oracle Content Management to host some paths of your domain and another service host other paths, routing will need to be handled by the CDN. DNS can only be used to segregate traffic at the domain and subdomain level.

Deploy Certificates

A certificate protecting a vanity domain needs to be created and hosted by the CDN. A certificate may protect a single domain, multiple domains, subdomains, and wildcarded subdomains such as `*.example.com`. Any combination is acceptable for a vanity domain. All protected domains will be visible in the certificate details, so if sharing these details publicly is unintended, separate certificates should be used.

> **Note:**
>
> The process for creating and hosting certificates is often specific to the CDN and they will need to specify how best to do this.

Set Up a Site Vanity Domain

The following steps must be completed to configure a site vanity domain. This process may be repeated for additional sites on the same domain, at different paths, or on different domains.

- Configure a Site With a Site Vanity Domain
- Configure the CDN to Route Requests to a Public Site
- Configure the CDN to Route Requests to a Secure Site

Configure a Site With a Site Vanity Domain

For an Oracle Content Management site to load properly when using a vanity domain, you must configure the site to do so. This is done in the site's properties.

1. In Oracle Content Management, click **Sites** in the side navigation.
2. Select the site you want to use a vanity domain with and choose **Properties** from the right-click menu or in the actions bar.
3. Enter the vanity domain in the vanity domain field and click **Save**.

It can take up to an hour for Oracle Content Management to be ready to accept requests on the vanity domain. During this time, you can access the site on the original domain. You can monitor progress at any time in the site properties panel.

> **Note:**
>
> If you are using the Oracle Content Management CDN you do not need to perform any additional actions. If you are using a different 3rd-party CDN, review Configure the CDN to Route Requests to a Public Site and Configure the CDN to Route Requests to a Secure Site. If necessary, consult your CDN for specific instructions.

Configure the CDN to Route Requests to a Public Site

Once Oracle Content Management is properly configured and ready to accept them, requests made using the vanity domain will be routed according to the DNS entries to the CDN and the CDN will forward the requests to Oracle Content Management. This routing is usually done with a CNAME entry in your DNS records. Consult your CDN for specific instructions.

For example, if an Oracle Content Management site with a site URL of `https://myinstance.cec.ocp.oraclecloud.com/site/MyCustomerSite/` is configured with a vanity domain of `https://www.example.com/store/`, then the CDN must be configured to:

- recognize the vanity domain and custom path: `https://www.example.com/store/`
- specify the origin Oracle Content Management instance using the vanity domain: `https://myinstance.cec.ocp.oraclecloud.com/`
- append the site path for the specific site, in this case: `/site/MyCustomerSite/`

- send the full site URL to the origin Oracle Content Management instance: `https://myinstance.cec.ocp.oraclecloud.com/site/MyCustomerSite/`

Oracle Content Management will then receive the request and respond to the CDN, which satisfies the request to the visitor's browser, showing only the vanity domain and custom path to the visitor: `https://www.example.com/store/`

CDN configuration steps are often specific to the CDN, so work with your CDN provider to properly configure the desired behaviors.

> **Note:**
>
> The CDN configuration altering the path must not apply to any requests containing the following strings. The trailing wildcard is required for proper matching.
>
> - /documents*
> - /system*
> - /content/published*
> - /osn*
> - /pxysvc*
> - /_compdelivery/*
> - /_themes/*
> - /site*
> - /_sitesclouddelivery/*
> - /favicon.ico*

Requests to these paths are not meant to include the site path and so should be excluded from the path modification behavior. They should resolve to the root of the Oracle Content Management instance to be handled properly.

Routing requests from a single vanity domain to multiple Oracle Content Management instances is not supported. Many required requests have shared paths that do not include a site identifier so it is not possible to properly route requests to the correct instance. It is recommended that you use different domains or subdomains if you are working with multiple Oracle Content Management instances.

Configure the CDN to Route Requests to a Secure Site

A secure site requires visitors to authenticate so Oracle Content Management can confirm they are authorized to view the site before accessing it. This authentication is handled by routing the visitor to an Oracle identity manager such as Oracle Cloud Infrastructure (OCI) Identity and Access Manager (IAM), and then back to the site once a proper session has been established. This means the CDN configuration for a secure site requires a few more behaviors than for a public site. Consult your CDN for specific instructions.

For example, if a secure Oracle Content Management site with a site URL of `https://myinstance.cec.ocp.oraclecloud.com/site/authsite/MySecureSite/` is configured with a vanity domain of `https://www.example.com/secure/` then the CDN must be configured to:

- recognize the vanity domain and custom path: `https://www.example.com/secure/`

- specify the origin Oracle Content Management instance using the vanity domain: `https://myinstance.cec.ocp.oraclecloud.com/`
- append the site path for this specific site: `/site/authsite/MySecureSite/`
- send the full site URL to the origin Oracle Content Management instance: `https://myinstance.cec.ocp.oraclecloud.com/site/authsite/MySecureSite/`
- ensure the Forward Host Header matches the vanity domain using a Custom Value or Incoming Host Header option.
- ensure all calls to the server function by enabling the HTTP DELETE, POST, PUT, and PATCH methods, which are often not enabled by default in CDN configurations.
- create a separate rule that will update the location header of the /cloudgate/v1/oauth2/callback response. This will ensure the visitor ends up at the correct domain and path.

  By default, the authenticated user will be returned to a combination of the vanity domain and original site path, such as `https://www.example.com/site/authsite/MySecureSite/`. You want the visitor returned to `https://www.example.com/secure/`. To do this, this rule must execute on the /cloudgate/v1/oauth2/callback request when the response's location header includes the name of your site. In this case, *MySecureSite*.

  This rule should then execute a find and replace of the location header's value, replacing `/site/authsite/MySecureSite/` with `/secure/`. A find and replace operation will allow all pages of the site to also redirect properly, where as a simple path replacement would always return the user to the home page.

When implemented correctly, Oracle Content Management will receive the request and respond to the CDN, which satisfies the request to the visitor's browser, showing only the vanity domain and path to the visitor. In this example: `https://www.example.com/secure/`

CDN configuration steps are often specific to the CDN, so work with your CDN provider to properly configure the described behaviors.

Set Up an Instance Vanity Domain

The following steps must be completed to configure an instance vanity domain. While multiple instance vanity domains can be configured, only a single instance vanity domain will be used by the user interface to display site URLs.

- Configure Oracle Content Management With Your Instance Vanity Domain
- Configure the CDN When Using Standard Paths
- Configure the CDN When Using Short Paths

Configure Oracle Content Management With Your Instance Vanity Domain

For Oracle Content Management sites to load properly on an instance vanity domain, you must configure Oracle Content Management properly.

1. Sign in as a service administrator and click **System** under **Administration** in the side navigation panel.
2. Select **Sites** from the banner menu.

3. Click **Manage Vanity Domains** under the **Vanity Domain Configuration** section and enter your instance level vanity domain and click **Save**. Multiple domains can be added and managed.
4. Select a vanity domain as the default.
5. Enable or disable **Display Short Paths** to toggle on or off the display of `/site/` or `/site/authsite/` in the user interface. This is helpful when most or all of your sites are either public or secure, and your CDN is configured properly.

> **Note:**
>
> Short paths aren't supported in private instances.

It can take up to an hour for Oracle Content Management to be ready to accept requests on the vanity domain. During this time, you can access your sites on the original domain.

> **Note:**
>
> If you are using the Oracle Content Management CDN you do not need to perform any additional actions. If you are using a different 3rd-party CDN, review Configure the CDN When Using Standard Paths and Configure the CDN When Using Short Paths. If necessary, consult your CDN for specific instructions.

Configure the CDN When Using Standard Paths

If **Display Short Paths** is disabled, all site URLs shown in the product will include the full instance vanity domain and site path. Your CDN needs to be configured to route those requests back to the Oracle Content Management origin unaltered.

Once Oracle Content Management is properly configured and ready to accept them, requests made using the instance vanity domain will be routed according to the DNS entries to the CDN and the CDN will forward the requests to Oracle Content Management properly. This is usually done using a CNAME entry in your DNS records. Consult your CDN for specific instructions.

For example, if an Oracle Content Management site has the URL of `https://myinstance.cec.ocp.oraclecloud.com/site/MyFirstProjectSite/` and you want to access that site at `https://www.example.com/site/MyFirstProjectSite/` the CDN must be configured to:

- recognize the vanity domain: `https://www.example.com/`
- identify the origin Oracle Content Management instance using the vanity domain: `https://myinstance.cec.ocp.oraclecloud.com/`
- passthrough the request path: `/site/MyFirstProjectSite/`
- and send the full request path to the origin Oracle Content Management instance: `https://myinstance.cec.ocp.oraclecloud.com/site/MyFirstProjectSite/`
- Oracle Content Management receives the request and responds to the CDN, which satisfies the request to the visitor's browser, showing only the vanity domain and standard path to the visitor: `https://www.example.com/site/MyFirstProjectSite/`

These same steps would apply to all requests made for a secure site. The only difference is those paths include `/site/authsite/` rather than just `/site/`.

CDN configuration steps are often specific to the CDN, so work with your CDN provider to properly configure the desired behaviors.

Configure the CDN When Using Short Paths

If **Display Short Paths** is enabled, site URLs shown in the product will only include the site name rather than including the `/site/` or `/site/authsite/` portion of the path.

For example, if you enable **Display Short Paths** and want to reach your Acme-Store site and you know it's a public site, you could make a request to `https://www.acme.com/Acme-Store/` and the CDN would inject `/site/` when going back to the Oracle Content Management instance with the full path of `https://acmeInstance.cec.ocp.oraclecloud.com/site/Acme-Store/`.

A limitation of this feature is that the CDN must know to inject `/site/` or `/site/authsite/`. This is because the Oracle Content Management instance must receive the full path, including `/site/` or `/site/authsite/`, depending on if the site is a public site or a secure site. This means this option is most useful when the majority of your sites are of the same type, either public or secure.

If you have a large mix of public and secure sites, then short paths may not be worth the effort required to maintain your CDN configuration. Preferably most of your sites would be of one type and each of the few remainders could then be handled with exception rules.

For example, let's say you have 10 sites, 9 of which are public and one is secure called *MyAccountSite*. Your CDN should be configured such that the public site requests coming to your domain, for a path other than `/MyAccountSite/` or one of the excluded paths listed below, have `/site/` injected into the path before going back to the Oracle Content Management instance to load the site resources. But if the request is for the secure site `/MyAccountSite/`, then an exception rule for that site will instead inject `/site/authsite/` into the path and the additional behaviors needed to authenticate users are done. If most of your sites are secure, then the CDN configuration should be reversed so that each public site would need an exception rule.

If you do not set up exception rules for each site not covered by the default path injection behavior in your CDN configuration, those sites will fail to load as your Oracle Content Management instance will not know where to find the site.

> **Note:**
>
> The CDN configuration altering the path must not apply to any requests containing the following strings. The trailing wildcard is required for proper matching.
>
> - /documents*
> - /system*
> - /content*
> - /osn*
> - /pxysvc*
> - /_compdelivery/*
> - /_themes/*
> - /site*
> - /_sitesclouddelivery/*
> - /favicon.ico*

Once Oracle Content Management is properly configured and ready to accept them, requests made using the instance vanity domain will be routed according to the DNS entries to the CDN and the CDN will forward the requests to Oracle Content Management properly.

For example, if an Oracle Content Management has been configured to use short paths, your sites are public, and a request is made to `https://www.example.com/MySecondProjectSite/` the CDN must be configured to:

- recognize the vanity domain: `https://www.example.com/`
- specify the origin Oracle Content Management instance using the vanity domain: `https://myinstance.cec.ocp.oraclecloud.com/`
- prepend `/site/` to the path
- send the full site URL to the origin Oracle Content Management instance: `https://myinstance.cec.ocp.oraclecloud.com/site/MySecondProjectSite/`
- Oracle Content Management receives the request and responds to the CDN, which satisfies the request to the visitor's browser, showing only the vanity domain and site name: `https://www.example.com/MySecondProjectSite/`

If most of your sites are secure sites the same rules apply. Instead of prepending `/site/` you need to prepend `/site/authsite/`.

Exception rules must be defined for all sites that are not the default type. Configure that exception rule to match on the specific site names so those requests can have the proper path appended rather than the default.

CDN configuration steps are often specific to the CDN, so work with your CDN provider to properly configure the desired behaviors.

Enable Cobrowse Integration

The Cobrowse feature is a collaboration tool used with the Oracle Cobrowse Cloud Service.

Integration with Cobrowse Cloud Service must first be added as an accepted integration by the service administrator. See Integrate with Oracle Cobrowse Cloud Service in *Integrating and Extending Oracle Content Management*.

After Oracle Cobrowse Cloud Service integration is enabled, the feature can be configured for the site and then added to specific site pages for use.

To enable cobrowsing on a site:

1. Open a site for editing.
2. Click  in the sidebar and then click  .
3. In the Cobrowse section, select **Enable use of Oracle Cobrowse on this site**.
4. Enter the launcher script for the site. An Oracle Cobrowse Cloud Service administrator can access the Cobrowse administration console to obtain the appropriate launcher Javascript snippet. There are two different types of launchers.
   - Launch Point 1: a cobrowse button is automatically added on a page.
   - Launch Point 2: allows you to customize the button and the interface that is added on a page.
5. Click **Close**.

After Cobrowse is enabled for the site, you can add it to a page or customize how it is used on a page. See Using Cobrowse on a Page for details.

Use Cobrowse with Secure Sites and Site Builder Testing

Cobrowse has two modes: Instant (ICB) and Advanced (ACB). An extra configuration is needed to use Cobrowsing in ICB mode for a secured site or to preview a site still in development. This configuration is done in the Cobrowse Administration Console. See *Log in to the Agent Console* in **Using Standalone Cobrowse**.

In the Custom Functions field, add a function to allow agents to see password-protected resources on a published secure site:

```
function () {
return {
passwordProtectedPatterns: [
"<PROTOCOL>://<DOMAIN>/authsite/*?*",
"<PROTOCOL>://<DOMAIN>/documents/*?*",
"<PROTOCOL>://<DOMAIN>/content/*?*#*"
]
}
}
```

To use this function for both published secure sites and to view/preview a site in development, add additional code:

```
function () {
return {
passwordProtectedPatterns: [
"<PROTOCOL>://<DOMAIN>/authsite/*?*",
"<PROTOCOL>://<DOMAIN>/documents/*?*",
"<PROTOCOL>://<DOMAIN>/content/*?*#*"
```

```
"<PROTOCOL>://<DOMAIN>/sites/*?*#*",
"<PROTOCOL>://<DOMAIN>/_themes/*?*",
"<PROTOCOL>://<DOMAIN>/_sitescloud/*?*",
"<PROTOCOL>://<DOMAIN>/_compdelivery/*?*"
]
}
}
```

Configuring protected resources is a new Cobrowse feature. It uses the same wildcard URL patterns as Cobrowse Page Masking. For more details, see Configure page masking in the *Cobrowse Deployment and Use Guide*.

Add Analytics Tracking

In Site Builder, you can add a snippet of JavaScript tracking code to a site or page for web analytics tracking, making it easier to integrate with external analytics providers like Google, Adobe, or Oracle Infinity.

To add analytics tracking to a site:

1. Open a site for editing.
2. Click  in the sidebar and then click  **Analytics**.
3. Click the switch to enable analytics tracking.
4. In the **JavaScript Tracking Snippet** box, add a new snippet or edit existing script. Your administrator may have supplied a code snippet for your environment. If so, that will show in the box. You can customize the script or add your own. If you edit the administrator supplied snippet, a message says that the script has been modified. To remove your customizations: **Restore to Latest Tenant Script**.

   Here's an example of a Google Analytics tracking snippet:

   ```
   <!-- Global site tag (gtag.js) - Google Analytics -->
   <script async src="https://www.googletagmanager.com/gtag/js?
   id=UA-85172963-3"></script>
   <script>
     window.dataLayer = window.dataLayer || [];
     function gtag(){dataLayer.push(arguments);}
     gtag('js', new Date());

     gtag('config', 'UA-85172963-3');
   </script>
   ```

You must save and publish this change, and, if necessary, bring the site online before analytics are collected for the site.

Viewing Analytics Data

After you publish the site and bring it online, you can view the tracked analytics data on the vendor's site, such as Google Analytics. If you used a snippet for Oracle Infinity analytics tracking, go to the Oracle Infinity home page and click **Analytics** to view the data and select or create reports.

12

Publish Sites

Let's learn about bringing sites online, taking them off, and publishing site changes.

- Bring a Site Online or Take It Offline
- Publish Site Changes

Bring a Site Online or Take It Offline

When a site is *online*, users with proper access can view it with a standard web browser at a designated address (URL). When a site is *offline*, the site isn't available for public viewing. You can view the site only in Oracle Content Management.

Before you can bring a site online, it must have been published. To publish the site, see Publish Site Changes.

The status icon on the right shows whether the site is online or offline:

- If the site has never been published before, you see a dash (-).
- If the site is online, you see  .
- If the site is offline, you see  .

For information on who can access an online or offline site, see Understand Site Security.

To change the status of a site, you must be the site owner or have the manager role, or, if site governance is enabled, site administrators can change the status of any site regardless of whether or not the site is shared with them.

1. On the Sites page, select the site from the list.
2. To bring a site online or take it offline, choose **Bring Online** or **Take Offline** in the right-click menu. You can also click  in the site tile to bring a site online, or click  in the site tile to bring a site offline.

   You're prompted to confirm your choice.

When you bring a site online, a fully rendered HTML version of the site is created and copied to the hosting location in Oracle Cloud. An online site shows its URL below the site name. The format of the default URL is:

`https://service_name.identity_domain.sites.oraclecloud.com/site_name`

When you take a site offline, the site and its folders and files are removed from the hosting location in Oracle Cloud.

Publish Site Changes

To publish site changes, you must be the site owner or have the manager role.

Keep in the mind the following when you publish site changes:

- When you publish changes to a site that's online, any committed changes are immediately visible to anyone with access to the site.
- When you publish changes to a site, you can publish all assets targeted to the site's channel or just the assets that are used on the site pages, including recommendations. For example, if you publish all the assets targeted to the site's channel and the site includes a list that references content items that aren't directly part of the site, the referenced assets will also be published.
- When you republish a site that has already been published, you can choose to republish only auxiliary files, such as a custom controller file or redirects, specific pages you select, or the entire site.
- When you republish a site that has already been published, you can choose to publish site detail pages with assets, so that if site compilation is enabled, static HTML pages will also be generated and published.
- When you publish changes to a multilingual site, only the translations for languages defined in the site’s associated localization policy will be published.
- If translations have previously been published for languages that are no longer defined in the site’s associated localization policy, the translated content will be removed from the published site.
- If you delete a page from a multilingual site, the translated pages will also be deleted when the site is published.

1. To publish site changes, select it and in the right-click menu or actions bar, choose **Publish** if this is the first time the site is being published, or **Republish** if the site has been published before. If republishing, you can select to republish the whole site, just pages you select, or just the auxiliary files. For example, if you configured the site with RSS feeds, redirects, or a custom controller file, you can publish them to the site runtime independently from publishing the whole site.
    a. If republishing auxiliary files, select **Publish Site Auxiliary Files** from the Republish menu, choose what auxiliary files you want to publish, then click **Publish**.
    b. If republishing individual pages, select **Publish Site Pages**, then choose the site pages you want to publish. Enable **Show Updated Pages** to list only those site pages that have been updated. If site compilation is enabled, static HTML pages will also be generated and published.
    c. If republishing detail pages, select **Publish Detail Pages with Assets**, then click **Select** next to the detail pages with assets to publish. Navigate to and select the assets to include then click **OK** to add them. Click **Publish**. If site compilation is enabled, static HTML pages will also be generated and published.
    d. If publishing or republishing the whole site, select whether to publish the site and all assets targeted to the site's publishing channel, or publish the site and all assets added to the site's pages.

2. By default, Oracle Content Management validates that all site strings and assets have the required approvals and translations. You won't be able to publish the site if you don't have the translations that are required by the site's associated localization policy, if the assets aren't marked as translated, or if an asset requires approval, but isn't marked as approved. If **Show site validation screen** is enabled, a validation panel is displayed listing the assets and pages in the site that have any problems with validation. Expand items to see more details. If there are issues, correct them and then try to publish again. If all items are valid, click **Publish**.

   If you want to see the full validation report, including assets that are ready for publication, enable **Show complete validation report**. Requesting a complete validation report can increase validation time. If you select **Show issues only**, the validation screen will show site pages and assets with **Not Ready** or **Informational** status only. This would speed up the time to render the validation screen.

   Whether you accept the default validation or request a full validation report, if your site uses a large number of assets the validation process can take a long time to complete. For example, a site with 30,000 assets may take up to an hour to validate, and you won't be able to click **Publish** until it is done. Disable **Show site validation screen** to allow Oracle Content Management to automatically publish the site when the validation process has finished successfully. If for some reason the site doesn't validate successfully, a notification showing the reason is displayed on the Sites page and you can take the appropriate steps to correct any issues.

3. If your site isn't already online ( ), you need to bring it online to make the site available to users.

View Site Publishing Job Details

You can view a list of publishing jobs by selecting **View Jobs** on the Sites page. Each job entry details the name and status of a job, when it was published and by whom. Click in the **Publish Log** or **Content Log** column to download a log file in JSON format.

13

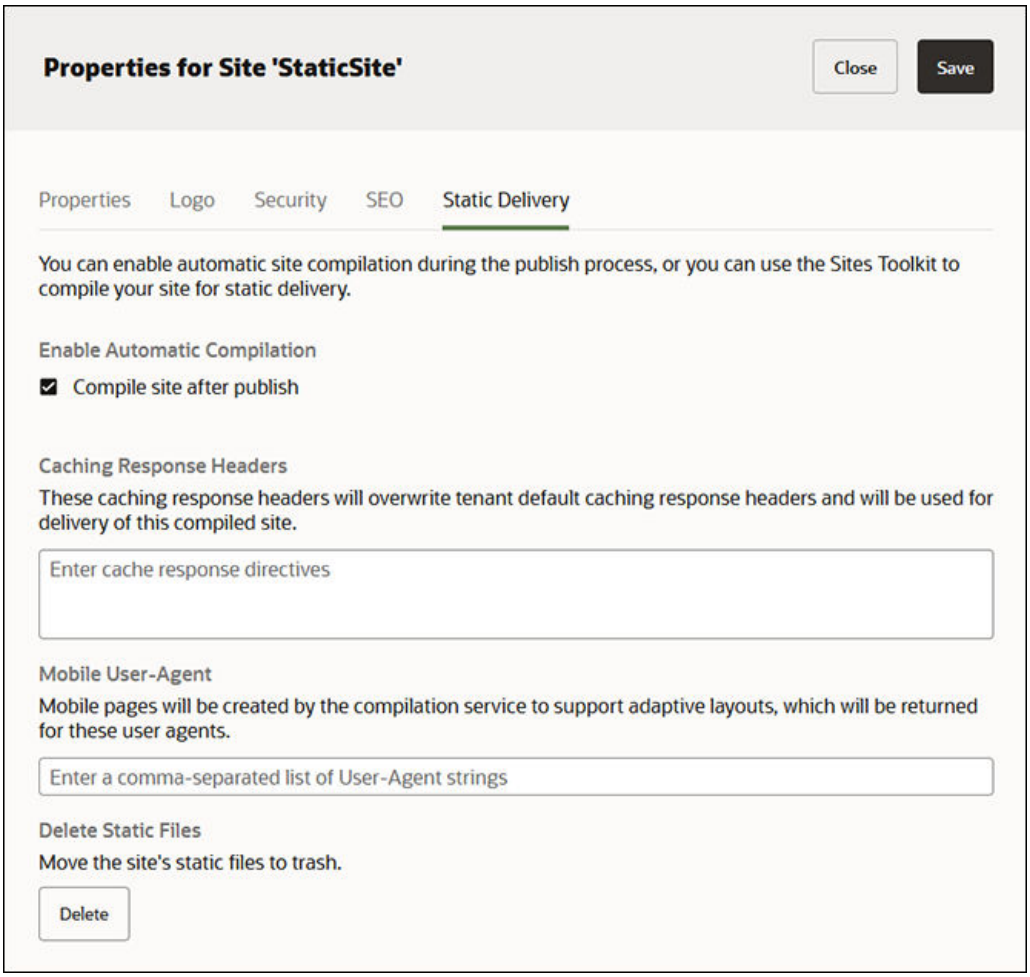
Compile Sites

Site compilation is the process of creating a *static* HTML file for each page in a site. The resulting HTML page will behave exactly the same as the dynamic page that would have been rendered by the JavaScript controller file in the web browser, but will load faster.

Site compilation in Oracle Content Management is an optional process that can be called after the site is published to generate the static HTML pages for the site. Once the site compilation is completed, the generated HTML files are uploaded to a 'static' folder under the site. From there they are published to the site runtime. Subsequent requests to a site URL will return these compiled HTML files.

Through site compilation, you can ideally remove all JavaScript from the page that is not required to support page functionality on the a client-side. This includes removing any Oracle Content Management JavaScript code, so that only HTML and CSS are rendered in the browser, which gives the page the fastest possible runtime performance.

Oracle Content Management has a compilation service built-in to easily allow you to compile a site without any additional configuration. Or you may manually set up a compilation service for testing purposes to validate site compilation, or to use custom libraries.



Site compilation is a Node.js application that executes in the background after the site has been published. It generates a server-side rendered page by combining the page layout with the *page-ID.json* metadata to generate the HTML file that will be returned, instead of the *controller.html* file with JavaScript. To determine how a site page is being rendered, you can view the page source in your web browser. If the page is delivered as compiled, you'll see the page HTML code. If it's delivered as controller page, you'll see the *controller.html* file instead.

See Compile and Publish Sites with Oracle Content Management for detailed information about the compile process, site compile API, and various compiler and delivery options.

Set Static Site Delivery Options

By default, Oracle Content Management uses a controller model to render site pages in a browser. You can enable site compilation on publish that will automatically generate HTML files for each page and deploy the resulting HTML files for static site delivery to improve site performance at runtime.

If your company uses compiled sites, you can control how long static sites are cached and what mobile user-agents call for adaptive mobile layouts supported by site compilation.

- Enable Automatic Compilation on Publish

Enable Automatic Compilation on Publish

Oracle Content Management has a compilation service built-in to easily allow you to compile a site without any additional configuration. Or your service administrator may manually set up a compilation service for testing purposes to validate site compilation, or to use custom libraries. Regardless of the compilation service used, you can choose to compile a site when the site is published or republished.

1. After logging in to Oracle Content Management, click **Sites** in the side navigation menu.
2. Select the site you want to modify and click **Properties**.
3. Click **Static Delivery** in the properties dialog.
4. Enable **Compile site after publish** in the **Enable Automatic Compilation** section.
5. When finished, click **Done**.

When you publish or republish a site with automatic compilation enabled, the publishing status is tracked and displayed on the site tile of the **Sites** page. After publishing is complete, the compiling process is tracked in the **Static Delivery** section of the site properties dialog. Once the compilation is complete, the site properties static delivery section lists the date and time of the last compilation and provides a link to download the compilation log.

Override Default Cache Control Headers for Compiled Sites

Compiled sites are cached on a user's browser for 300 seconds (5 minutes) by default. Your service administrator may change this default, but as a site developer, you can override the default for specific sites in the site properties.

1. After logging in to Oracle Content Management, click **Sites** in the side navigation menu.
2. Select the site you want to modify and click **Properties**.
3. Click **Static Delivery** in the properties dialog.
4. In the **Caching Response Headers** section, enter `Cache-control: max-age=` and then a numeric value for the number of seconds you want the page cached in a user's browser. For example, `Cache-control: max-age=600` would cache the page for 10 minutes.
5. When finished, click **Done**.

If your instance uses Akamai, to keep the existing Akamai settings, leave the **Caching Response Headers** section blank. To override the existing Akamai settings, enter `Edge-Control: !no-store,max-age=1800,downstream-ttl=1800` where the bold items are the default settings in seconds.

- `!no-store` indicates that this setting should override the corresponding Akamai caching configuration for the property.
- `max-age` determines how long Akamai should cache this page. The default is 1800 seconds (30 minutes). During that time, Akamai will fulfill requests for the page without requesting the page from Oracle Content Management.

- `downstream-ttl` tells Akamai to send a "Cache-Control: max-age" header with its response to client browsers, instructing those browsers to cache the page for the alotted time. The default is 1800 seconds (30 minutes).

Specify Mobile User-Agents to Support Compiled Adaptive Layouts

When compiling a site, mobile pages can be created to support adaptive layouts. You can specify the user-agents that will cause the server to deliver the mobile pages instead of the standard compiled pages for a site. The values entered here are treated as case-insensitive substrings when matching against the user-agent headers sent by browsers. Note that the keyword `Mobile` is commonly used in the user-agents strings for browsers on mobile devices.

1. After logging in to Oracle Content Management, click **Sites** in the side navigation menu.
2. Select the site you want to modify and click **Properties**.
3. Click **Static Delivery** in the properties dialog.
4. In the **Mobile User-Agent** section, enter a comma separated list of user-agent substrings for the user agents you want served mobile pages. If any part of the substring matches the browser's user-agent string, then the mobile pages are served.

Delete Static Files to Render a Site Dynamically During Compilation

The first time you compile a site, the site is rendered dynamically until compilation ends. Once compilation completes, static files are used to render the site.

If you publish a site with new data, the previous static files are used until new static files are compiled. This is useful if publishing a large site that takes a long time to compile.

If you would prefer your site to render dynamically during compilation, you need to delete the static files before you recompile.

1. After logging in to Oracle Content Management, click **Sites** in the side navigation menu.
2. Select the site you want to modify and click **Properties**.
3. Click **Static Delivery** in the properties dialog.
4. At the bottom of the tab, click **Delete Static Files**.

Existing static files are deleted and the site is rendered dynamically until the next time a site is compiled for static delivery.

14

Secure Sites

When you publish a site and make it available online, you'll want to control who can access the site.

When you secure a site, you specify which groups of users can access your published (online) site based on an assigned role. These roles are service-wide roles assigned by an administrator of the service instance.

- Understand Site Security
- Change Site Security

If you are an administrator, there are additional actions you can take that relate to site security, such as enabling custom sign-in, allow sharing of sites and themes, limit site, template, and component creation, enable governance, and other actions. See Configure Sites and Assets Settings in *Administering Oracle Content Management*.

Understand Site Security

You can apply security to control who can see the published (online) site, who can see and interact with secure content on the site, and who can see and edit the unpublished (offline) site.

Site Security

When you publish a site and make it available online, you'll want to control who can access the site. Depending on how your system and site administrators configured your environment, you can make the site publicly available to anyone, restrict the site to registered users, or restrict the site to specific users.

You must be the site owner or have the manager role to change site security or any other settings. To change the sign-in requirement, the site must be offline. To change the specified users or user roles, however, the site can be online. When you take a site offline, the site and its folders and files are removed from the hosting location in Oracle Cloud.

The security options available may be limited by the template policy if site governance is enabled or by the tenant policy if site governance is disabled. See Understand Site Governance.

When you secure a site, you specify which groups of users can access your published (online) site based on an assigned role. These roles are service-wide roles assigned by an administrator of the service instance.

- **Cloud Users**: Authenticated users sign in to the service instance with a user name and password. This includes all authenticated users with or without the **Visitors** role or the **Users** role.
- **Visitors**: Only users with this role can access the site. For example, this role might be given to users who can see published sites, but don’t have access to folders and files in this instance of Oracle Content Management.

> **Note:**
>
> This doesn't include users with the **Users** role unless they're the site owner or the site was explicitly shared with them.

- **Service Users**: Only users with this role have access to the site. For example, this role might be given to users who can both see published sites and have access to folders and files in this instance of Oracle Content Management.
- **Specific Users**: Only the users you add as members of the site can see the published site.

Site Sharing

With *site sharing*, you specify individual users who can access your unpublished (offline) site and allow them to view, modify, or manage the site based on the permission you give them. You can share a site if you're the site owner or if the site was shared with you and you were given the Manager role.

> **Note:**
>
> Any sharing role you assign to a user augments their security role. For example, if a user has the **Visitors** role, but you share the site and give them a contributor role, they can modify the offline site while others with the **Visitors** role can only view the online site.

- **Viewer**: Viewers can see sites in the site listing and view the properties of each site, but not change them. They can also opt to remove themselves as a member.

> **Note:**
>
> If governance is enabled and a viewer has the CECSitesAdministrator role, then the viewer can also delete the site.

- **Downloader**: Downloaders can view the site in the editor, but can't change anything. Downloaders can also copy and export a site.
- **Contributor**: Same as downloader, and can also edit the site, delete site pages, and delete the site if it's offline.
- **Manager**: Same as contributor, and can also add users and assign their roles, publish changes to an online site, and switch the site online and offline. The creator of a site (the owner) is automatically assigned the manager role.

When you create a site, a channel is created with that site name. In order to share the channel with others, you must share the site and give someone a minimum of a contributor role in order for that person to use the channel for publishing assets. In order to publish a site, a user must have a Manager role.

Component Sharing

Some components provide access to shared resources such as folders, files, or conversations. *Component sharing* considers both site security (who can view the

published site) and resource sharing (who can view and work with folders, files, and conversations).

For example, when you add a document manager component to your site, all visitors to the site can see the content of the folder, and based on their role and any other permissions, they may be able to add, modify, or delete what's in the folder.

General considerations:

- A site author can't grant access to a folder that's greater than the access they themselves have. For example, if the author has downloader access to a folder, they can't give contributor rights to site visitors.
- The privileges set in the component can augment the visitor's privileges. For example, if the visitor has viewer privileges (or no privileges) for a folder, the documents manager component can grant greater privileges based on the role selected in the component. These enhanced privileges are valid only in the component itself.
- If a site visitor has privileges that are greater than those specified for the component, their individual privileges override those set on the component.
- Privileges granted on a folder apply to the folders and files nested in that folder.

For public sites:

- Conversation components are supported on secure sites only.
- Documents manager components give all visitors downloader privileges for the associated folder by default. You can change the role within the guidelines listed above, and you can restrict the options presented to the user with settings on the component itself.
- Folder list and file list components grant all users downloader access. Users can view and download files regardless of their role.

Secure Sites URL

When you make a site online, a fully rendered HTML version of the site is created and copied to the hosting location in Oracle Cloud. An online site shows its URL below the site name.

The format of the default URL for unsecured sites is:

`https://service_name.identity_domain.sites.oraclecloud.com/site_name`

The format of the default URL for secured sites is:

`https://service_name.identity_domain.sites.oraclecloud.com/authsite/site_name`

Note the addition of `authsite` in the URL.

Change Site Security

When you publish a site and make it available online, you'll want to control who can access the site. Depending on how your system and site administrators configured your environment,

you can make the site publicly available to anyone, restrict the site to registered users, or restrict the site to specific users.

You must be the site owner or have the manager role to change site security or any other settings. To change the sign-in requirement, the site must be offline. To change the specified users or user roles, however, the site can be online. When you take a site offline, the site and its folders and files are removed from the hosting location in Oracle Cloud. To take a site offline, select the site from the Sites page and choose **Take Offline** in the right-click menu or click  in the actions bar. You're prompted to confirm your choice.

To change site security:

1. On the Sites page, select the site and choose **Properties** in the right-click menu or click in the actions bar.
2. Click the **Site Security** tab.

   The available options on the Site Security tab depend on how your system administrator configured site security settings and, if site governance is enabled, how your site administrator configured the security settings in the template this site is based on. See Get Started with Sites and Understand Site Governance.
3. To require registered users to sign in to see the site when it's online, click **Yes** next to Login Required. To remove the requirement and make the site publicly available when it's online, click **No** next to Login Required.
4. Select which groups of registered users can access the online site. To select individual groups, first deselect **Cloud Users**.
   - **Cloud users**: Only authenticated users have access to the site. Authenticated users sign into your domain with a user name and password. This includes users with either the **Oracle Content Management Cloud Visitors** role or the **Oracle Content Management Cloud Users** role.
   - **Visitors**: Only users with this role have access to the site. This doesn't include users with the **Oracle Content Management Cloud Users** role.
   - **Service users**: Only users who can sign into this instance of Oracle Content Management can access the site.
   - **Specific users**: Specify individuals who can access to the site. Click **Add Members**. Enter a user name or a portion of a user name in the search field. Select the user from the displayed list and repeat to add more users. When done, click **Add**. To remove a user, click **Remove** from the menu below the user's name.
5. Click **Save** to save your changes and close the window.

   The site shows that it's offline and that login is required.
6. To bring the site online, choose **Bring Online** in the right-click menu or click  in the actions bar. Click **Confirm to proceed** and then click **OK**.

When you bring a site online, a fully rendered HTML version of the site is created and copied to the hosting location in Oracle Cloud. An online site shows its URL below the site name.

The format of the default URL for unsecured sites is:

`https://service_name-identity_domain.cec.ocp.oraclecloud.com/site/site_name`

The format of the default URL for secured sites is:

`https://service_name-identity_domain.cec.ocp.oraclecloud.com/site/authsite/site_name`

Note the addition of `authsite` in the URL.

You can add a logout URL and implement it as a link, or a button, or a page that shows up in the menu. See Paragraphs, Buttons, and Add Pages.

The format of the logout URL is:

`https://service_name-identity_domain.cec.ocp.oraclecloud.com/cloudgate/logout.html?postlogouturl=%2Fsite%2Fauthsite%2Fsite_name`

> **Note:**
>
> The postlogouturl needs to be in encoded format as above.

15

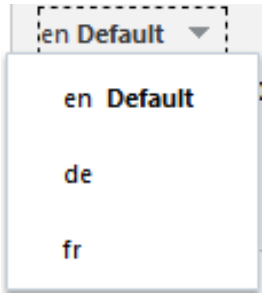
Work with Multilingual Sites

Learn about site translations for targeted, multilingual experiences.

- Overview of Multilingual Sites
- Translate a Site
- Manage Site Translation Jobs

Overview of Multilingual Sites

You can translate a site into any language specified in the localization policy. To see the languages specified in the site's associated localization policy, in the site tile, click the down arrow next to the language. To preview a localized version, select the language, and open the site.



> **Note:**
>
> With Oracle Content Management Starter Edition, you cannot create multilingual sites. For a full feature set and unlimited sites, upgrade to Oracle Content Management Premium Edition.

When you select a site for translation, a .zip file of the site files is created.

> **Note:**
>
> If the site contains assets from multiple repositories, only those assets from the default repository are included in the .zip.

The .zip contains the following files and folders:

- `assets` folder—This folder exists only if you select to translate the complete site or just the targeted assets.

  - `root` folder
    - a `<contentItem_ID>.json` file for each content item—includes name, description, and translatable strings from fields in the content item, as well as additional information about the content item that shouldn't be edited.
  - `job.json`—a file describing the translation job. Don't edit this file.
- `site` folder—This folder exists only if you select to translate the complete site or just the site content.
  - `root` folder
    - a `<page_ID>.json` file for each site page—includes name, title, description, keywords, header, footer, and translatable strings from components on the page, as well as additional information about the page that shouldn't be edited. If you have custom components, they might have been configured to use translatable strings. See Develop Translatable Components for Multilingual Sites.
    - `siteinfo.json`—includes description, keywords, header, and footer.
    - `structure.json`—includes navigation and site structure.
  - `job.json`—a file describing the translation job. Don't edit this file.

Translate a Site

If you have assets on a page in a translated site, either directly or in a content list, and you've translated those assets, the assets will display in the same language as the site. You can also translate content items separate from a site; see Localize Content Items.

When translating a site that contains assets from multiple repositories, only assets from the default repository are included in the translation job

> **Note:**
>
> With Oracle Content Management Starter Edition, you cannot work with translations. For a full feature set and unlimited sites, upgrade to Oracle Content Management Premium Edition.

Create a Translation Job

When exporting a translation package for manual translation or using a translation connector, you need to create a translation job.

1. Select the site you want to translate, and then click **Translate**. You might need to click **More** to see the **Translate** option.
2. Enter information for the site files you're exporting — the translation job, and then click **Create**:
   - Enter a name for the translation job.
   - Select the target languages into which the site will be translated.

- Select whether to export the complete site—site content (pages, structure, and site info) and targeted assets, just the site content, or just the targeted assets.
- Select **Include only published versions of assets** or not. Including only published asssets means any revisions of assets being worked on in the source language that have not yet been published will be excluded from the translation job. This is useful if content providers begin new drafts of assets in the source language as soon as a site is published.
- Select a translation connector or choose to export a translation package for manual translation.
- Click **Create** when done.

Once the translation job is finished, the translated package is imported into Oracle Content Management. If for any reason a translation job fails, select it and choose **Resubmit** in the right-click menu or actions bar. The original job is deleted and a new one submitted.

Manually Translate the Source Language Files

If you're manually translating content, you need to download the .zip file of the site files after the translation job has finished.

1. Click **Translation Jobs** in the banner, select the translation job, then click **Download**.
2. For each language you selected as a target language, create a folder in the .zip file at the same level as the `root` folder, for example, `de`, `es`, and `fr`. You can translate a subset of the selected languages. For example, you could translate into German (`de`) now, and then translate into Spanish (`es`) and French (`fr`) at a later time.
3. Copy *all* the `.json` files from the `root` folder to each language folder.
4. Translate the strings in all of the `.json` files to the appropriate languages. Do not delete any strings from the .json files, and do not rename the files.
5. Zip up the `assets` (if your translation job includes assets) and `site` (if your translation job includes site content) folders, with `job.json`, `root`, and all the language folders with the translated files.

Import a Manually Translated Package

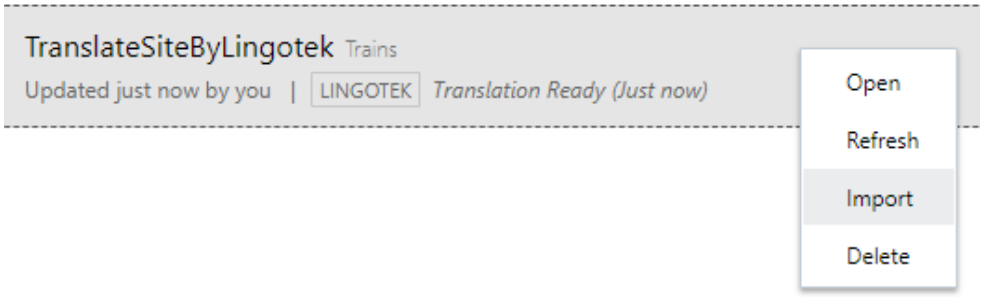
Whether translation was done manually or automatically, you need to import the finished translation package.

1. On the **Translation Jobs** page, click **Import**.
2. Click **Upload**, select the .zip file of translated site files, then click **Open**.
3. After the upload is finished, click **OK**
4. Oracle Content Management validates that the translations that are defined in the job are available in the .zip file. If you want to see which pages and assets are included in the translation job, click the link in the dialog.
5. When you're ready to import the translations, click **Import**.
   The status of the import appears above the banner. You can view the details of the job by clicking **Details**.

Import a Translated Package from a Translation Connector

Whether translation was done manually or automatically, you need to import the finished translation package.

1. Click the ••• and select **Translation Jobs**.
2. Right-click on the finished translation job and select **Import**.



3. Oracle Content Management validates the translations that are defined in the job. If you want to see which pages and assets are included in the translation job, click the link in the dialog.
4. When you're ready to import the translations, click **Import**.
   The status of the import appears above the banner. You can view the details of the job by clicking **Details**.

After importing your translations, you need to publish your site and make sure the site is online to make the translations available on your site.

If any changes are made to translations at the translation service after you've imported the translation package, you can click **Refresh** to update the translations.

If you change the site after translation, you'll need to translate any new or edited strings. When you edit a site in Site Builder, you edit the default language version of the site. Any site structure changes you make, such as adding components or rearranging pages, will be replicated in the localized versions of the site. You can then create a new translation job to translate the updated strings.

View Site Translation Job Details

On the Translation Jobs page, click a translation job to view its details.

The translation job details lists the following information:

- The translation job name and repository.
- The job status.
- When the job was last updated.
- The source and target languages, and the status of each target language.

If your translation uses a translation connector, you'll see additional information:

- The translation connector used for the job.
- The status of the translation. Click **Refresh** to refresh the translation job details.
- If your translation connector provides validation data of the upload to Language Service Provider step, you can download the details in a JSON file by clicking **Download**.
  You can also view validation data for the site by expanding the **List of assets**. The validation data includes whether the fields or attributes and any associated native

files are translatable. If the connector encounters any problem while uploading the site to the Language Service Provider, an error would be shown.

Manage Site Translation Jobs

When you select a site for translation, a *translation job* is created. You can then download the files for translation, translate them, and then import the translated files.

To create a translation job, see Translate a Site.

The Translation Jobs page lists all translation jobs and their status:

-  (Ready) - The .zip file of site files is ready to be downloaded.
-  (In Progress) - The .zip file has been downloaded. The status will remain in progress until all translations for all targeted languages have been imported successfully.
-  (Complete) - The translations for all targeted languages for this job have been imported successfully.
-  (Failed) - The translation job failed. You should have seen a failure message above the banner about why the job failed. If you need to see the message again, you can click **Details** in the translation job listing. To resubmit a failed job, select it and choose **Resubmit** in the right-click menu or actions bar. The original job is deleted and a new one submitted.

You can perform the following actions:

- To search for a translation job, enter the job name in the action bar search box.
- To view the details of a translation job, open it. The details include the source language and all the selected targeted languages, and the status of those translations. If the job has failed, click **Resubmit** to try again.
- To download the .zip file of site files, select the job, and click **Download**.
- To delete a job, select the job and click **Delete**.
- To import translations, click **Import**, then click **Upload**, select the .zip file of translated site files, then click **OK**.

  Oracle Content Management validates that all the translations that are defined in the job are available in the .zip file. If you want to see which site pages and assets are included in the translation job, click the link in the dialog. When you're ready to import the translations, click **Import**.

Locales for Translation

When submitting an item for translation the target language is identified by a code so the language service provider knows what language to translate the item into and return. For example, **fr** represents French and **de** represents German.

These codes can be extended for more regional dialects. For example, **de-LI** is the code for German as it is spoken in Liechtenstein and **de-LU** is the code for German as it is spoken in Luxemburg. But if the language service provider doesn't support a regional dialect, then the code provided is truncated to the two character base language. For **de-LI** and **de-LU** the code would be truncated to **de**, for example.

If the language service provider supports one regional dialect but not all, it may substitute. For example, **ms-BN** is the code for Malay as spoken in Brunei, but if the language service provider doesn't support that dialect, it may switch to a dialect it does support, such as **ms-MY**, which is the code for Malay as it is spoken in Malaysia. If the language service provider doesn't make a distinction between dialects, for example **en-BZ** for English as spoken in Belize and **en-JM** for English as spoken in Jamaica, then it will truncate to the base language, in this case **en** for English.

Custom Locales for Translation

Custom locales may be created by a developer based on your organization's needs. Custom locale codes include the base language, any regional dialect code if applicable, an **x** to designate it is a custom locale, and whatever other identifying customization is required by your organization. For example, a custom locale for English might look like **en-JM-x-custom**.

Because a custom locale is unique to your organization, custom locale codes are truncated when submitted for translation to the base language and the regional dialect if supported by the language service provider. In the example above, **en-JM-x-custom** would be truncated to **en-JM**, eliminating the portion of the code specific to the customization. Or if the language service provider does not support the regional dialect code for Jamaica (JM), it may be truncated to just the base language, **en**.

Set Locale Alias for URL Redirect

You can easily set an alias for locales that is used in a site URL at runtime and the runtime preview. This is especially useful if your organization makes use of custom locales that can be long and add complexity to a URL.

For example, you may have a custom locale defined as en-GB-x-cornish, which in the URL would look like this:

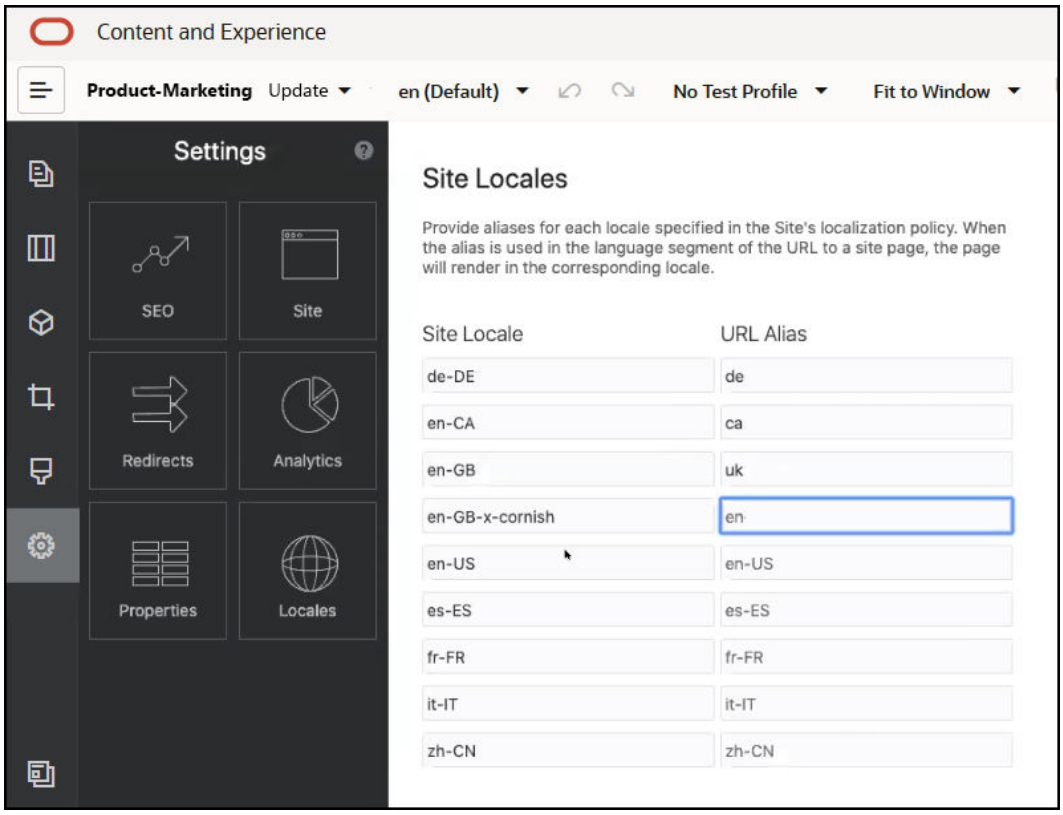
`https://example.com/site/BlogSite/en-GB-x-cornish/home.html`

By adding an alias, you can redirect to a simpler URL:

`https://example.com/site/mysite/en/home.html`

1. Open a site for editing.
2. Click  in the sidebar and then click **Locales**.

   A list of all locales used in your site is displayed next to corresponding URL Alias fields.



3. Enter an alias next to each site locale you want an alias for, click **Close**, and then **Save**.
4. To preview the alias used in the URL at runtime, click 
5. When you publish the update, the changes are published and put into use.

Set Fallback Locales

In some situations, you as a site manager may want to support locales whose translations are not yet available. In these situations you can designate a fallback locale to display in place of the locale that is not complete. If you support a large number of locales but may not have translated them all yet, you can save time when publishing or compiling your site if you set a fallback locale for similar languages. For example, you may want to render a site in generic French (fr) if the locales for French Canadian (fr-CA), Swiss French (fr-CH), Luxembourg French (fr-LU), and Belgian French (fr-BE) have not been translated yet, even though they are available as localization options.

Note:

- By default a locale does not have a fallback, so it displays the language itself.
- After a site locale is used as a fallback for another locale, you cannot define a fallback for it. For example, after (fr) is set as fallback for (fr-CA), fallback for (fr) can only be (fr) itself.
- A site locale can be set as a fallback for more than one locale. For example, (fr) can be set as fallback for (fr-CA) and (fr-CH).

To set a fallback locale:

1. Open a site for editing.
2. Click  in the sidebar and then click **Locales**.

   A list of all locales used in your site is displayed next to corresponding locale fallback fields.
3. Select a fallback locale for every site locale you want a fallback for, then click **Close**.
4. Save your update and when you're ready, commit your changes to the update/
5. To preview the alias used in the URL at runtime, click 
6. When you publish the update, the changes are published and put into use.

16

Use Site Redirects or URL Mapping

When you restructure or move a web site, you can redirect user requests from old URLs to their current ones. Specifying 30x redirects for URLs can maintain bookmarks or published links across site redesigns.

Pages that have high reputation rankings in search engines could move to different URLs when you move to Oracle Content Management hosted sites from other infrastructure technologies. Redirects help reorganize the URL structure of a site and preserve the search engine rankings.

Plan for Redirects

You can specify redirects that send HTTP 30x responses for designated URLs. If a request does not match one of the nominated redirects, then regular processing of the URL happens, and the page is returned in the normal way.

You can create a JSON file specifying redirects and upload that file to the server. The server will use the JSON file as it process incoming request URLs.

Two kinds of redirect rules let you redirect incoming URLs to new locations:

Simple String-to-String Matching

For simple string matching and replacement, you can specify explicit URLs and then redirect each URL by mapping it directly to a target URL.

The following table shows some sample string-to-string matchings.

Source URL	Target Location URL
`/index.html`	`/home.htm`
`/products/widget`	`/items/knickknack`
`/index?page=widgets`	`/items/widgets`

String-to-string mappings are simple to understand and test. The rules evaluate quickly using simple string matchings and map lookups.

However, there is little flexibility regarding URL query parameters. They would need to match exactly. Extra URL parameters or parameters in a different order would cause a rule to not match.

Simplified Wildcard Matching

Simplified wildcard matching lets a rule match many URLs while also limiting the amount of regular expression backtracking required to obtain a result.

Because regular expressions can be complicated to write, and because poorly constructed ones can evaluate for an undetermined amount of time (`ReDoS`), a second type of rule allows a simplified matching mechanism. It uses a wildcard character ("`*`") to match zero (0) or more characters in the incoming URL, and the keyword `'wildcard'` with an index value to copy incoming parts of the URL to the redirected URL.

The following table shows some sample simplified wildcard matchings.

Source URL	Target Location URL
`/old/*`	`/new/<$wildcard(1)$>`
`/dispatch.asp?page=*&facet=Lang*`	`/page<$wildcard(1)$>/<$wildcard(2)$>`

Simplified wildcard matching gives more power to the matching of URLs than simple string-to-string matching, but does so without unbounded regular expression processing. The syntax is simple, and you can use pattern matching on a URL to accommodate a large number of URLs with one pattern.

Because they are based upon regular expressions, wildcard rules would evaluate somewhat slower than simple string matching. A large number of rules could introduce a performance penalty to general page-delivery performance.

Add Site Redirects

If the site URL changes, a redirect forwards one URL (source) to another URL (target). This helps to preserve user bookmarks and search engine rankings.

Two types of redirects can be used:

- A permanent redirect, which uses a 301 HTTP service response code
- A temporary redirect, which uses a 302 HTTP service response code

To upload a `redirect.json` file:

1. Open a site for editing.
2. Click  in the sidebar and then click  **Redirects**.
3. Click **Select file to upload** and navigate to the file you want to use, select it, then click **OK**.
4. When you publish the update, the changes are published and put into use.

Specify Redirect Rules in a JSON File

You can specify redirect rules for URLs in a JSON file.

Use the following format in a JSON file to specify redirect rules for URLs.

```
{
     "redirectRules":
     [
        {
            "type": "string",
            "comment": "this rule is applied first",
            "expression": "/index.htm",
            "location": "/home.html"
        },
        {
            "type": "wildcard",
            "expression": "/items/*?page=*",
            "location": "/<$page$>?item=<$wildcard(1)$>",
            "code": 302
        }
    ]
}
```

The outer containing structure in the JSON file is an array. The array contains rule instances.

The `"string"` rules will be evaluated first, followed by the `"wildcard"` rules, in order. Once one of the rules matches, the evaluation of subsequent rules is abandoned, and the corresponding redirect is generated.

Each rule has the following properties:

- The "type" property is an optional string that indicates the type of redirect rule. The value must be one of the following strings:
  - `"string"` specifies a faster rule whose expression matches the entire input URL exactly.
  - `"wildcard"` specifies a wildcard rule that can match a number of URLs. This is the default value if the property is omitted.
- The `"expression"` property is a required string that matches against the incoming site-relative URL. In a wildcard rule, the asterisk (`*`) token matches zero or more characters.
- The `"location"` property is a required string that indicates the location or destination of the redirect. The redirect can be a full or relative URL.
- The `"comment"` property is an optional string that has no impact upon the evaluation of the rules. It includes notes or commentary.
- The `"code"` property is an optional integer that provides the HTTP response code to use when issuing the redirect. The value must be one of the following integers:
  - 301: Indicates that the resource moved permanently. This is the default value if the `"code"` property is omitted.
  - 302: Indicates that the resource moved temporarily.

- Only "wildcard" rules can have the "flags" property. The two legal flags are "globstar" and "caseinsensitive", separated by comma. The "globstar" flag changes the way that the "*" matches in a wildcard rule "expression". Normally, the "*" matches all possible characters. When "globstar" is set, the "*" matches until the next "/" character in the URL. By default, expression matching is case sensitive unless the flags value includes caseinsensitive.
- The "enabled" property is an optional property used to enable or disable the rule. It can be true or false. The default value is true.

Location Tokens

You can use location tokens to help manufacture a redirect location. Each of the following location tokens can help specify a redirect:

- <$urlPath$>: The path portion of the matching URL.
- <$urlQueryString$>: The entire URL query string from the matching URL.
- <$urlQueryStringExcept(name1,name2)$>: The entire URL query string from the matching URL minus the named parameters.
- <$wildcard(N)$>: The one-based index of the matching wildcard in the matching URL. (This is analogous to \1..\9 in regular expressions.)
- <$name$>: The value of the named query string parameter. For example, if you have the query string msmith: ?page=42 on the input, then you can use <$page$> in the location to put '42' into the location.

Restrictions

The following restrictions apply to the redirects.json file as a whole and to the rules it contains:

- The maximum overall file size accepted by Oracle Content Management is 250 KB.
- The maximum number of rules in the redirects.json file is 1,000.
- The maximum "expression" length for a rule is 1,000 characters.
- The maximum "location" length for a rule is 2,000 characters.
- The maximum number of '*' tokens in a wildcard rule expression is 10.

String Matching Example

Rule:

```
{
      "type": "string",
      "expression": "/old/page.jsp?id=material&type=glass",
      "location": "/new/<$id$>.htm"
}
```

The following URL would match the rule:

```
/old/page.jsp?id=material&type=glass
```

- The resulting location would be: `/new/material.htm`
- The entire URL matches, including the query string.
- Although `<$id$>` is used in the location, it is not necessary for this example because only one possible query string could match. The location could have been written as `/new/material.htm`.

The following URLs would not match the rule:

- `/old/page.jsp`

  (The rule's expression gives a query string that must match.)

- `/old/page.jsp?id=material&type=glass&index=2`

  (The extra `&index=2` in the candidate URL does not exactly match the rule expression.)

- `/old/page.jsp?type=glass&id=material`

  (The ordering of query string parameters must match in a "string" rule.)

Wildcard Matching Example

Rule:

```
{
      "type": "wildcard",
      "expression": "/old/*/pages/*?id=*&item=sheet-*",
      "location": "/new/<$id$>/<$wildcard(4)$>.html"
}
```

The following URLs would match the rule:

- `/old/phones/android/pages/info.asp?id=XT1045&item=sheet-specs`
  - The resulting location would be: `/new/XT1045/specs.html`
  - The path portion of the URL matches, so the query string is also examined for matching conditions.
  - The parameters in this example happen to match the ordering of the parameters in the rule expression, but this is not required.
- `/old/phones/android/pages/info.asp?item=sheet-specs&id=XT1045`
  - The resulting location would be: `/new/XT1045/specs.html`
  - The path portion of the URL matches the rule expression before the question mark (`?`), so the parameters are also checked for a match.
  - Although the parameters are listed in a different order in the rule expression, the parameters are matched individually.
- `/old/phones/android/pages/info.asp?id=XT1045&item=sheet-specs&unrelated=thing`
  - The resulting location would be: `/new/XT1045/specs.html`
  - The path portion of the URL matches, so the query string is also examined for matching conditions.
  - The candidate URL has an extra `&unrelated=thing` parameter, but since the named query parameters in the rule expression match, the rule is deemed to match.

  - The `unrelated` parameter would be available in the location as a token, as `<$unrelated$>`, and would have the value `thing`, even though it did not contribute to the match of the rule.

The following URLs would not match:

- `/old/pages/info.jsp`

  (The path portion of the URL does not match the path portion of the rule expression.)

- `/old/phones/android/pages/info.asp`

  (The path portion of the URL matches the path portion of the rule expression, but the query parameters in the rule expression do not match.)

- `/old/phones/android/pages/info.asp?id=cellular`

  (The path portion of the URL matches the path portion of the rule expression, but not all of the query parameters in the rule expression match.)

Defining a Token Array

You can also create an array of token definitions within the `redirects.json` file to help when configuring redirects that support multiple vanity URLs. This allows you to redirect appropriately based upon the characteristics of the incoming URL.

Use the following format in the `redirects.json` file to define tokens for use in redirect rules URLs.

```
{
     "tokenDefinitions":
     [
        {
            "token": "sitePrefix",
            "type": "hostmatch",
            "expression": "example.com",
            "value": ""
        },
        {
            "token": "sitePrefix",
            "type": "hostmatch",
            "expression": "*.com",
            "value": "/site/Starter-Site"
        },
        {
            "token": "gotoRedirect",
            "type": "pathmatch",
            "expression": "*oracle*",
            "value": "https://www.oracle.com",
            "flags": "caseinsensitive"
        }
    ]
}
```

The tokenDefinitions have the following properties:

- `"type"`: One of the following:

  - "hostmatch" to match the incoming URL's host value.
  - "pathmatch" to match the incoming URL's pathname value.
  - "querymatch" to match the incoming URL's query value.
- "token": The name of the token to define.
- "expression": The expression that should be used for matching. Wildcard characters are supported.
- "value": The value that should be used for the token.
- "comment": This is an optional property used to explain or annotate the token definition.
- "flags": The two legal flags are "globstar" and "caseinsensitive", separated by comma. The "globstar" flag changes the way that the "*" matches in a token. Normally, the "*" matches all possible characters. When "globstar" is set, the "*" matches until the next "/" character in the URL. By default, expression matching is case sensitive unless the flags value includes caseinsensitive.
- "enabled": This is an optional property used to enable or disable the token. It can be true or false. The default value is true.

When computing the value of a token, the tokenDefinitions array will be enumerated in order. The first matching definition will be used. If no token definitions satisfy the token, then an empty string will be used instead. For expediency and performance, commonly used tokens should be placed at the top of the tokenDefinitions list.

The tokenDefinitions have the following constraints:

- You can create up to 250 token definitions.
- The token name must be less than 100 characters.
- The expression can have up to 10 wildcards.
- The expression must be less than 1000 characters.
- The value must be less than 1000 characters.

Example

For example, you may have the following redirects.json file:

```
{
     "redirectRules":
     [
        {
            "type": "string",
            "expression": "/legacy-privacy-policy.html",
            "location": "<$pathPrefix$>/about/new-privacy-policy.html"
        }
    ],
     "tokenDefinitions":
     [
        {
            "token": "pathPrefix",
            "type": "hostmatch",
            "expression": "vanity.com",
            "value": "/fashion"
```

```
        }
    ]
}
```

In this case the rule's `location` property has a `<$pathPrefix$>` token. The `pathPrefix` token is defined in the `tokenDefinitions` section. If the incoming URL matches "vanity.com", then the `pathPrefix` value will be set to `/fashion`. This will be used in the `location` response, resulting in `/fashion/about/new-privacy-policy.html`.

Let's assume the first vanity domain URL is `http://example.com/legacy-privacy-policy.html`. This would match the first and only redirect rule.

The declared `location` for this rule is `<$pathPrefix$>/about/new-privacy-policy.html`. In this situation, the `<$pathPrefix$>` token needs to be evaluated. In order to do that the `tokenDefinitions` array is enumerated to find a match.

The first token definition is considered. Its `token` is the desired one, so it is further evaluated. The expression `vanity.com` does not match the incoming URL's `example.com` so this definition does not satisfy the requirements and the enumeration continues.

At this point there are no more token definitions, so the empty string is used for the value of the `<$pathPrefix$>` token. The final location returned for this redirect is `/about/new-privacy-policy.html`.

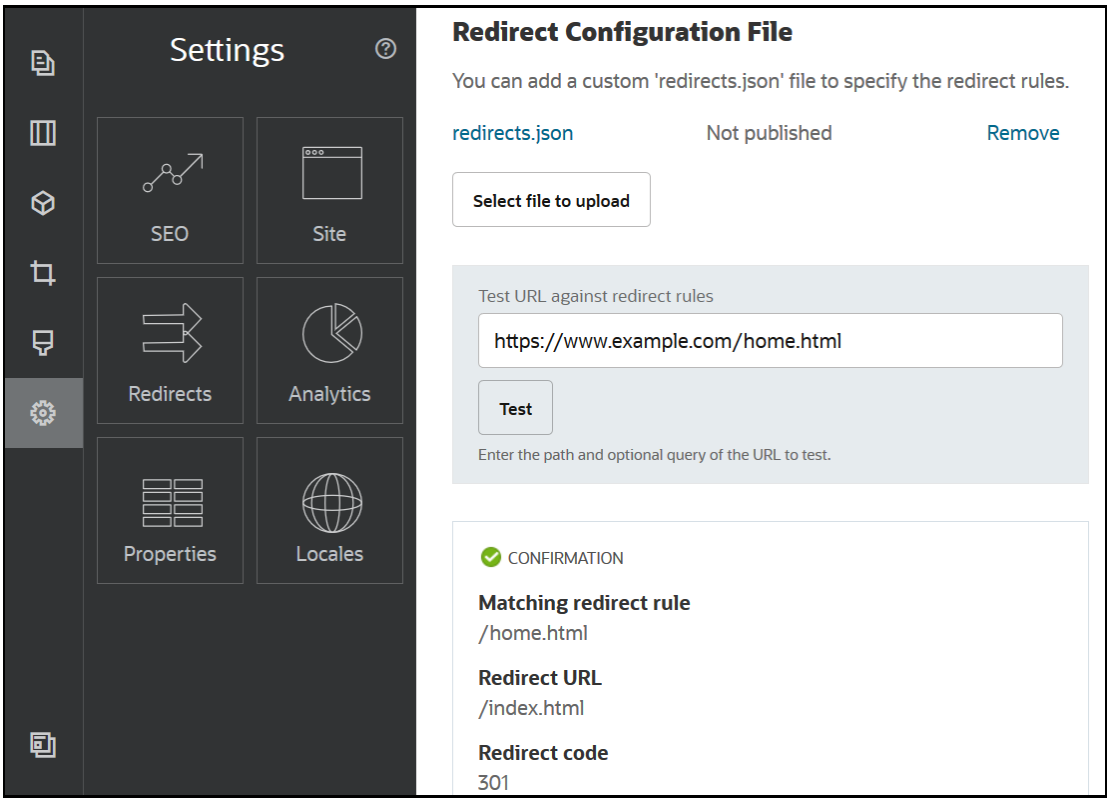
Let's assume the second vanity domain URL is `http://vanity.com/legacy-privacy-policy.html`. As with the first URL, the declared `location` for this rule is `<$pathPrefix$>/about/new-privacy-policy.html`. In this situation, the `<$pathPrefix$>` token needs to be evaluated. In order to do that the `tokenDefinitions` array is enumerated to find a match.

The first token definition is considered. As before, its `token` is the desired one, so it is further evaluated. The expression `vanity.com` does match the incoming URL's `vanity.com` so this definition does satisfy the requirements and the value `/fashion` is used as the token's value.

Because a match for the token was found, enumeration of the token definitions array stops, and the final location is computed as `/fashion/about/new-privacy-policy.html`.

Testing Site Redirects

You can test site redirects when editing a site by opening the **Settings** panel and clicking **Redirects**. Enter a URL to test and click **Test**.



Upload a Redirect Rules File to a Site

You can upload a redirect rules to a site in Oracle Content Management.

To upload a `redirect.json` file to a site:

1. Open the site for editing.
2. Click  in the sidebar and then click  .
3. Click **Select file to upload** and navigate to the file you want to use, select it, then click **OK**.
4. When you publish the update, the changes are published and put into use.

Map a Site URL

Once a site is created and published using Oracle Content Management, you can configure the Domain Name System (DNS) so that this site is accessible with a registered domain name, such as www.mysite.com

A Domain Name System (DNS) specifies where someone can find your web pages by mapping your domain name to your site's location, or canonical name (CNAME).

To map your domain name, you'll need the following:

- The URL of your Oracle Content Management instance. It's typically of the following form:
  service-tenant.documents.*datacenter*.oraclecloud.com

- The domain name as registered by your domain name registrar.
  For example, `www.example.com`. It could also be a sub-domain, such as `www.example.com/subdomain`.
- An account with a content delivery network (CDN) provider. Oracle Content Management provides integration with Akamai. Contact Oracle Support to have Akamai configured for your instance.

If you want to use your own CDN, rather than the Oracle Content Management-provided Akamai, perform the steps below.

Different Domain Name System providers have different web interfaces and different steps for updating a CNAME record. The steps below provide the information you'll need and the general steps to follow.

To map the site URL to a domain name:

1. Request a secure sockets layer (SSL) certificate from your content delivery network provider for the domain. For example, https://www.example.com.
2. Configure the content delivery network so that:
   a. The content delivery network accepts all incoming requests to the domain and forwards them using secure protocol (https).
   b. The origin points to the domain from Oracle Content Management:

      service-tenant.documents.*datacenter*.oraclecloud.com
3. Change the DNS server zone file to map the domain name to the edge server provided by the content delivery network provider:

   domain CNAME *CDN Server*
4. Wait for the update to propagate. Depending on your DNS service, this can take anywhere from 2 to 48 hours.

   After the change is propagated, you can access the site using your domain name. For example:

   `https://www.mysite.com/site_name`

By default, the endpoint for the Oracle Cloud REST API for Content Management is available if you use the standard URL provided for the site. Folder and file list components, for example, use the REST API to perform folder and file operations. If you use a custom URL, verify that you have access to the endpoint with your domain name. For example:

`https://www.mysite.com/documents`

17

Improve Site Performance

You can improve the performance of content delivery and rendering in the browser by leveraging the browser cache. Above the fold (ATF) rendering can also improve website rendering.

- Leverage Caching to Improve Performance
- Above the Fold (ATF) Rendering

Leverage Caching to Improve Performance

Delivery of content items, digital assets, and sites should take full advantage of a visitor's browser cache to improve performance of content delivery and rendering in the browser.

Sites, themes, content items, and digital assets are cached for an amount of time in the visitor's browser cache. After a site, theme, content item, or digital asset is updated, a cache-buster key in the URL is changed so that the browser has to fetch a different URL and get the new item.

The cache key helps to manage usage of the browser cache by referencing only current resources. Although the cache key is included in the URL, it is a logical element, not a physical location (folder) as is often the case. A change in the cache key does not point to a different physical location to find the resource; it simply notifies the server to fetch the current version of the resource.

Resources can be static, like CSS, JS, and image files, or dynamic, like page data, site data, and content item data. There are five categories of resources for building a website:

- Product resources – Resources that are part of the product that gets updated whenever a new version of the product is released or patched.
- Site Resources – Resources that are part of the site, like `structure.json`, page data, and images. These are updated when the site is published. The controller is described in the following text.
- Theme resources – Resources that are part of the themes, like layouts, CSS, and images. These are updated when the theme is published.
- Component resources – Resources that are part of custom components. These include HTML, JS, and CSS, and image files that make up the component. These are updated when a component is published. If one component changes and is republished, then the cache key changes for all components because it's a single key for all components.
- CaaS resources – Resources that serve content items and digital items. These are updated when content items are published or republished or the collection target is changed.

The following topics describe caching for the Oracle Content Management runtime and Site Builder:

- Runtime Caching
- Site Builder Caching

Runtime Caching

For runtime, the Oracle Content Management Cache-Control header is set to 15 days. A cache key is added to the URL for all resources.

As long as the URL is the same, the browser will serve the resource from its local cache if available. When the resource is updated, the cache key is updated in the URL, which forces the browser to make a new request to the server and update the local cache.

The controller, which contains the cache keys, is also cached for 1 minute. Because of this, any updated cache keys will not be seen for up to 1 minute.

At runtime the server returns `controller.html` with the latest cache keys for product, site, theme, components, and CaaS resources. A script with keys is added to `controller.html`; for example:

```
<script type="text/javascript">
        var SCSCacheKeys = {
                    product: '123',
                    site: '456',
                    theme: '789',
                    component: '012',
                    caas: '345'

        };

</script>
```

These keys are used by `controller.js` to construct URLs like the ones in the following table.

Type of Resources	Examples
Product Resources	*/sitePrefix/productCacheKey/_sitesclouddelivery/...* /mySite/_cache_947d/_sitesclouddelivery/
Theme Resources	*/sitePrefix/themeCacheKey/_themesdelivery/themeName/...*
Component Resources	*/sitePrefix/compCacheKey/_compdelivery/compName/...*
Site Resources	*/sitePrefix/siteCacheKey/content/...* */sitePrefix/siteCacheKey/structure.json* */sitePrefix/siteCacheKey/pages/100.json*

Type of Resources	Examples
CaaS Resources	*RegularCaaSUrl*?cacheKey=*caasCacheKey*

By inserting the cache key into the URLs like this, Oracle Content Management can force the browser to load updated resources by effectively changing the URL so the browser thinks it's actually a new resource.

> **Note:**
>
> For secure sites, only the product, theme, and component resources are cached, not the site or CaaS content.

Site Builder Caching

In Site Builder, static resources are cached for 15 days.

When you use Site Builder, caching happens for product, theme, and component resources. (It does not happen for site and CaaS resources.) Theme and component cache keys are regenerated when Site Builder is launched or refreshed.

If you make a change to a theme or component and want that change to appear in Site Builder, you need to refresh Site Builder (F5).

Above the Fold (ATF) Rendering

ATF rendering gives the appearance of a website loading faster than it actually does. The goal is to render first all the parts of a page that are visible and then, before the user scrolls down, render the rest of the page that is not initially visible.

A slot can have an "above the fold" designation, which displays an icon on the tab.

For a slot to be rendered in this new way, it must be marked with `scs-atf`, as follows:

```
<div class="scs-slot scs-atf" id="headline"></div>
```

A component needs to notify the renderer when it is done rendering. Out-of-the-box components do this by default. A custom component can make additional calls and needs to do the following:

1. Notify the renderer that it wants the renderer to wait until it is completed rendering.
2. Notify the renderer when it has completed.

For #1, against the custom component's `appinfo.json` file, add in the following property:

```
"initialData": {
     . . .
     "customRenderComplete": true,
     . . .
```

For #2, in the component's `render.js` file, make sure that you let the renderer know when you're done by calling:

```
SitesSDK.setProperty('renderComplete', true);
```

If not all components in an ATF slot report back that they are done in a timely manner, then the renderer will wait for 2 seconds before continuing with the rest of the page. If you know this will not be long enough, you can extend the time by declaring the following global variable in a page template:

```
var SCSAtfPassTimeout = 3000;
```

> **Note:**
>
> The time is in milliseconds so this example set the timeout to 3 seconds.

An API provides diagnostic data for the ATF process. You can call the following method in the debug console, or you can access it from a page if needed:

```
SCSRenderAPI.getRenderMetrics();
```

For example:

```
{currentTime: 16243.400000000001, renderStartTime: 264.36,
atfPassEndTime: 306.535, mainPassStartTime: 316.475, mainPassEndTime:
331.38500000000005, …}

1.   atfComponentCount:13

2.   atfPassEndTime:306.535

3.   completionCount:23

4.   completionRecords:Array(23)

1.   0:{atf: true, componentId: "a7afdd33-3fbb-4329-
bc1b-6be60056a995", time: 280.065}

2.   1:{atf: true, componentId: "edfcfcb4-b0d3-422f-
aa59-5c925bbbebee", time: 283.54}

3.   2:{atf: true, componentId: "c1c3aec8-e52f-406c-8c29-
ab69c05877ed", time: 283.56000000000006}

4.   3:{atf: true, componentId: "b3a31dc6-62a1-44d9-9c80-
bdb2c5bedaaa", time: 284.13000000000005}

5.   4:{atf: true, componentId: "c05aa1a2-
c11c-4ef5-9051-4799c5bee24a", time: 284.15500000000003}

6.   5:{atf: true, componentId:
```

"bafd4047-06ec-4739-9b23-9db74f573f30", time: 294.665}

7. 6:{atf: true, componentId: "e7d49528-0357-4b45-801e-b3a2716a086c", time: 297.995}

8. 7:{atf: true, componentId: "a5f33674-4022-4138-8cc5-fef00c02a557", time: 299.78000000000003}

9. 8:{atf: true, componentId: "ccfedc98-1dbd-440e-b867-5e683cea2ec5", time: 301.19500000000005}

10. 9:{atf: true, componentId: "d691bc44-fed9-474a-9806-2191f46a5e2e", time: 302.46}

11. 10:{atf: true, componentId: "cf613054-05d8-40dd-83a0-718760d7bc73", time: 303.79}

12. 11:{atf: true, componentId: "b4a6ef98-ffc8-48c7-987c-63346ee97bcc", time: 305.115}

13. 12:{atf: true, componentId: "de1fa2ce-66ba-419b-b517-2cb4a7601c3b", time: 306.535}

14. 13:{atf: false, componentId: "ba3f8ed4-31d4-4347-b6f0-f1019783a57c", time: 318.665}

15. 14:{atf: false, componentId: "ae8af486-76b3-47cd-9989-db4212eefebb", time: 320.45500000000004}

16. 15:{atf: false, componentId: "a48b5abb-49b2-4456-90bd-a3de998150c8", time: 320.48}

17. 16:{atf: false, componentId: "a9650e6d-7e7e-42a2-b758-58f2aeab18a2", time: 322.61500000000007}

18. 17:{atf: false, componentId: "aca9836a-f955-4aa7-8db2-fd3cf1189dea", time: 324.23500000000007}

19. 18:{atf: false, componentId: "e3d7941c-fbc7-4da9-963b-e3810b6467d4", time: 325.85}

20. 19:{atf: false, componentId: "eecde809-da54-4066-9326-73f9d9c35fe4", time: 327.315}

21. 20:{atf: false, componentId: "e8f4fb16-4e15-4570-b7de-304e99e449a7", time: 328.74}

22. 21:{atf: false, componentId: "a7baa06e-7f30-42c7-94f4-e171ab2edcd6", time: 330.09000000000003}

23. 22:{atf: false, componentId: "fd603b96-2beb-4e87-a54f-12d0e264cd0a", time: 331.38500000000005}

24. length:23

```
25. __proto__:Array(0)
5.   componentCount:23
6.   currentTime:16243.400000000001
7.   mainPassEndTime:331.38500000000005
8.   mainPassStartTime:316.475
9.   renderStartTime:264.36
10. __proto__:Object
```

Part IV
Developing for Sites

This part details how to begin site development by helping you understand and develop templates, themes, styles, components and other useful items. It includes the following chapters:

18

JavaScript Technologies

When developing components and themes for Oracle Content Management, you can choose any JavaScript technology stack you prefer. Samples provided with Oracle Content Management are based on vanilla JavaScript with some Mustache for basic templating, but there is no requirement to choose any specific JavaScript technology.

Load Custom Components

When using Oracle Content Management Site Builder, custom components and section layouts must be loaded into a page dynamically. This allows site contributors to add, remove, and position the components or section layouts on the page. To support this, each custom component must implement a render file that is responsible for bootstrapping the component into the page.

Components may be self-contained and work within any theme, or may rely on supporting JavaScript files in a theme's assets. Note that if custom components rely on theme assets, they will only work with sites based on that theme. Such components can be marked as **Theme Components** so that they are only available to sites that are based on the theme.

Load Components Dynamically

To dynamically load custom components into a page, the components must be built using either JavaScript Modules or, for legacy usage, RequireJS. If a component is built with JavaScript Modules, Oracle Content Management will load the component using:

```
import(".../componentName/assets/render.mjs").then((Component) => {...});
```

If a component is built with RequireJS, Oracle Content Management will load the component using:

```
require([".../componentName/assets/render"], function (Component) {...});
```

To distinguish between the technologies, components that use JavaScript Modules for component loading use a *.mjs* extension, for example *render.mjs*, and those that continue to use RequireJS will use ".js".

> **Note:**
>
> The use of JavaScript Modules is not supported by Internet Explorer. If your Oracle Content Management site still requires support for Internet Explorer, you must continue to use RequireJS for component development.

Default Inclusion of Components in a Site

If the the template used to create an Oracle Content Management site is set to use JavaScript Modules as the default JavaScript technology, then it uses a file called *render.mjs* to dynamically load custom components used in the site. If instead the template used to create a site is set to use RequireJS as the default JavaScript technology, then it uses a file called *render.js* to dynamically load custom components used in the site.

To change the default setting. select the site on the Oracle Content Management sites page and click **Properties** to open the side panel. Select the JavaScript technology to be used as the default under the **Properties** tab in the panel.

If for any reason the renderer can't find the default option for loading a custom component, it will note this in the console and try to load the component using the alternate file. This allows for backwards compatibility and helps migrate from RequireJS to JavaScript Modules.

> **Note:**
>
> Components can have both render.js and render.mjs files so that they can work regardless of the default JavaScript technology setting for a site. Typically this means that the render.js file is a simple generic wrapper that loads up the render.mjs file.

Alternatives to RequireJS Functionality

RequireJS provides additional functionality that JavaScript Modules does not. For example, with RequireJS you can access scoped technologies such as JQuery and Mustache that are exposed in Oracle Content Management through an API call. You can also take advantage of non-JavaScript resource loaders such as CSS or text.

To support this additional functionality when using JavaScript Modules, the SCSRenderAPI has been enhanced.

```
// To access scoped resources
SCSRenderAPI.getJQuery();
SCSRenderAPI.getMustache();

// To load non-JavaScript resources
SCSRenderAPI.importCSS(".../css/design.css").then(...)
SCSRenderAPI.importText(".../template.html").then(...)
```

When using content layouts in headless mode, the API call needs to provide the information and leverage the ContentSDK.

> **Note:**
>
> Implementation of these functions will update based on web browser support of ECMA standards. For example, import of stylesheets like JavaScript Modules.

Non-native JavaScript Technologies

Some choices of JavaScript technology stacks require additional processing before they can run in a browser. For example, using JSX or TypeScript in custom components will require the use of a transpiler such as BabelJS in the runtime environment unless the developer includes their own build step to create a version of the component that uses native JavaScript.

Oracle Content Management does not include Babel or provide any additional support for these non-native JavaScript technologies and it is up to the developer to make the decision on how the resulting JavaScript will run within the page.

Runtime Optimization

Adding components dynamically to a site page provides a site contributor with a custom experience for designing their pages. However, at runtime you want pages to run as fast as possible.

To support this, Oracle Content Management provides a compilation step that allows the developer of custom components and themes to generate the final HTML and CSS for the component and have it inserted directly into the page so that it is immediately available as the page renders.

There may be no need for any additional JavaScript execution for a custom component at runtime. However, if there are interactive elements for a component, there is a hydrate option available for the custom component that can be called after the page has rendered.

Default Inclusion of Components During Compile

The compilation step is run as a NodeJS application that looks for a *compile.mjs* file in the component's asset folder. If it finds the file, it will load the file and try to execute the `compile()` function within the component. If it doesn't find the file, it will try to load the *compile.js* file instead and use that.

NodeJS has historically used CommonJS for component loading. Similar to how there can be a *render.js* and a *render.mjs* file, there can be a *compile.js* file used by CommonJS for legacy components and a *compile.mjs* file for loading by JavaScript Modules.

Because JavaScript Modules are supported by both the browser and NodeJS, standardizing on .mjs files allows for easier sharing of common code between the *render.mjs* and *compile.mjs* implementations.

19

Customize Designs and Styles

Each theme for Oracle Content Management must have a design that specifies the look and feel of pages and style settings for components used in the default site for the theme.

About Designs

Each theme for Oracle Content Management must have a design, which specifies the look-and-feel of pages and style settings for components used in the theme.

When users create a site, they must select a template which includes a theme by default. You can create or choose to use a different theme for a site.

When creating a theme, along with designing page layouts you must specify the available styles that will be displayed in the Settings panel for each component type that will be available to users (Paragraph, Title, Image, and so on). There can be multiple styles within a design to specify the default settings for different components. You make choices for text fonts and font sizes, image framing, and so on, then save them as a named design. The items in the design influence the look of the page layouts when rendered.

See Work with Site Pages in *Building Sites with Oracle Content Management*.

Design Files

Two files are considered the default design files for a theme: `design.json` and `design.css`.

- `design.json` specifies styles for components
- `design.css` provides definitions for class values (such as color and font)

These files are located in the `/designs/default/` directory in a theme structure.

design.json File

The `design.json` file has the following structure:

```
{
    "componentStyles": {
        "scs-image": {
            "styles": []
```

```
        },
        "scs-map": {
            "styles": []
        },
        "scs-title": {
            "styles": []
        },
        "scs-paragraph": {
            "styles": []
        },
        "scs-divider": {
            "styles": []
        },
        "scs-button": {
            "styles": []
        },
        "scs-app": {
            "styles": []
        },
        "scs-spacer": {
        },
        "scs-gallery": {
            "styles": []
        },
        "scs-youtube": {
            "styles": []
        },
        "scs-socialbar": {
            "styles": []
        },
        "scs-document": {
            "styles": []
        }
    }
}
```

Each of the `"styles":[]` entries can contain a list of styles for that particular component. For example, the title component provides these default styles:

```
    "styles": [{
                    "name": "COMP_STYLE_FLAT",
                    "class": "scs-title-default-style"
               },
               {
                     "name": "COMP_STYLE_HIGHLIGHT",
                     "class": "scs-title-style-2"
               },
               {
                     "name": "COMP_STYLE_DIVIDER",
                     "class": "scs-title-style-3"
               }
    ]
```

> **Note:**
>
> If you create custom styles for a component and map it to styles in `design.json`, you don't need to use the full name like those provided with the system. Just specify the string you want to use. For example, instead of `"COMP_STYLE_BOX"` for the name, simply use `"Box"`. This means only the name `"Box"` will appear in the list for the **Styles** tab of the Settings panel for that component, instead of `"COMP_STYLE_BOX"`.

The name values are mapped to the actual words to display in the user interface, like this:

```
"COMP_STYLE_FLAT": "Flat",
"COMP_STYLE_HIGHLIGHT": "Highlight",
"COMP_STYLE_DIVIDER": "Divider",
```

design.css File

The `design.css` file provides the definitions for the class values. Here are a few examples.

```
.scs-title-default-style {
  color: #333333;
  display: block;
  font-family: "Helvetica Neue", "Helvetica", "Arial", sans-serif;
  font-size: 24px;
  font-weight: normal; }


.scs-title-style-2 {
  background-color: #DEF300;
  color: #333333;
  font-family: adobe-clean, sans-serif;
  padding-top: 2em;
  padding-bottom: 2em; }

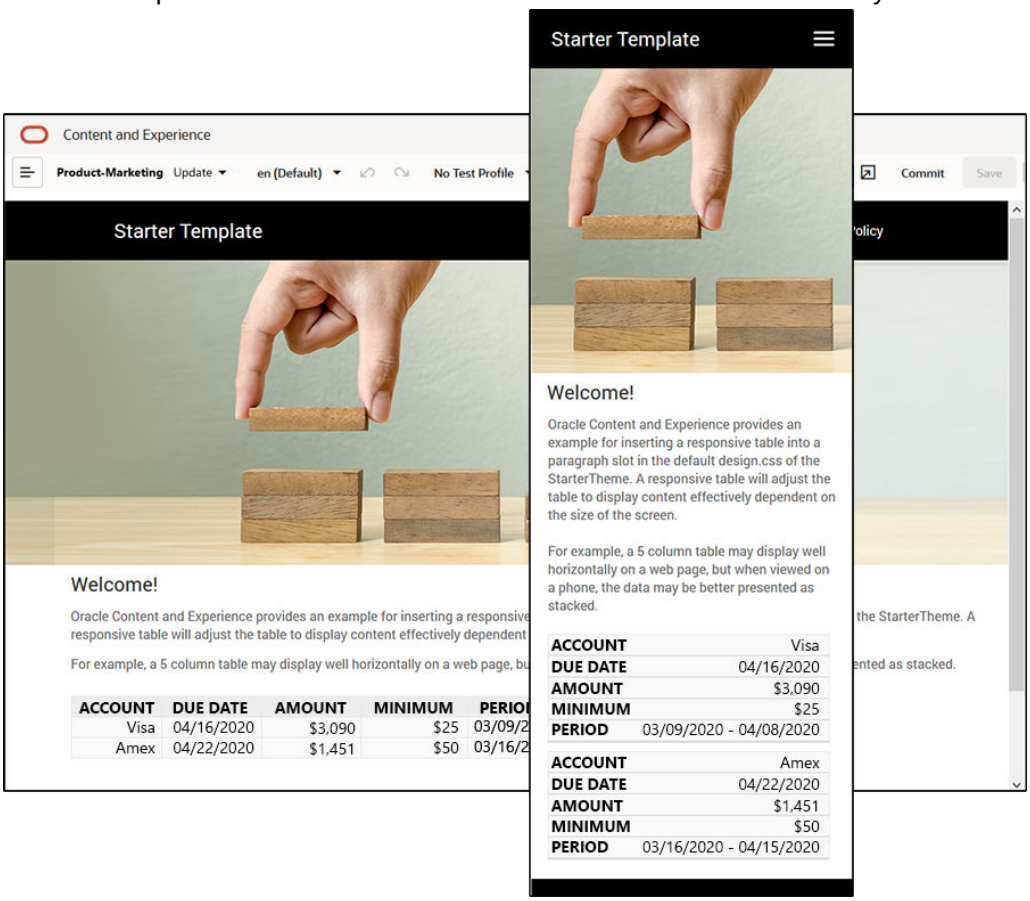

.scs-button-default-style .scs-button-button:hover {
  background: #f7f8f9;
  border: 1px solid #c4ced7;
  color: #0572ce;
  box-shadow: inset 0 1px 0 #f7f8f9;
  text-shadow: 0 1px 0 #f7f8f9; }


.scs-button-default-style .scs-button-button:active {
  background: #0572ce;
  border: 1px solid #0572ce;
  color: #fff;
  box-shadow: inset 0 1px 0 #0572ce;
  text-shadow: 0 1px 0 #0572ce; }
```

Responsive Table Design

Oracle Content Management provides an example CSS of a responsive table within a paragraph component that enables the stacking of row data when displayed on mobile devices.

A responsive table will adjust the table to display content effectively dependent on the size of the screen. For example, a 5 column table may display well horizontally on a web page, but when viewed on a phone, the data may be better presented as stacked. Note that responsive tables need a header row in order to behave correctly.



In the following generated HTML, note that there is an added data-label attribute to each table cell with values matching the column header text.

```
<thead>
    <tr>
        <th scope="col">ACCOUNT</th>
        <th scope="col">DUE DATE</th>
        <th scope="col">AMOUNT</th>
        <th scope="col">MINIMUM</th>
        <th scope="col">PERIOD</th>
    </tr>
</thead>

<tbody>
    <tr>
```

```
        <td data-label="ACCOUNT">Visa</td>
        <td data-label="DUE DATE">04/16/2020</td>
        <td data-label="AMOUNT">$3,090</td>
        <td data-label="PERIOD">03/09/2020 - 04/08/2020</td>
    </tr>
```

Once that attribute is on each cell, the TDs stack on top of each other when you apply the CSS rules below.

```
.scs-paragraph:not(.scs-paragraph-edit) table td {
  border-bottom: 1px solid #ddd;
  display: block;
  text-align: right;
}

.scs-paragraph:not(.scs-paragraph-edit) table td::before {
  content: attr(data-label);
  float: left;
  font-weight: bold;
}
```

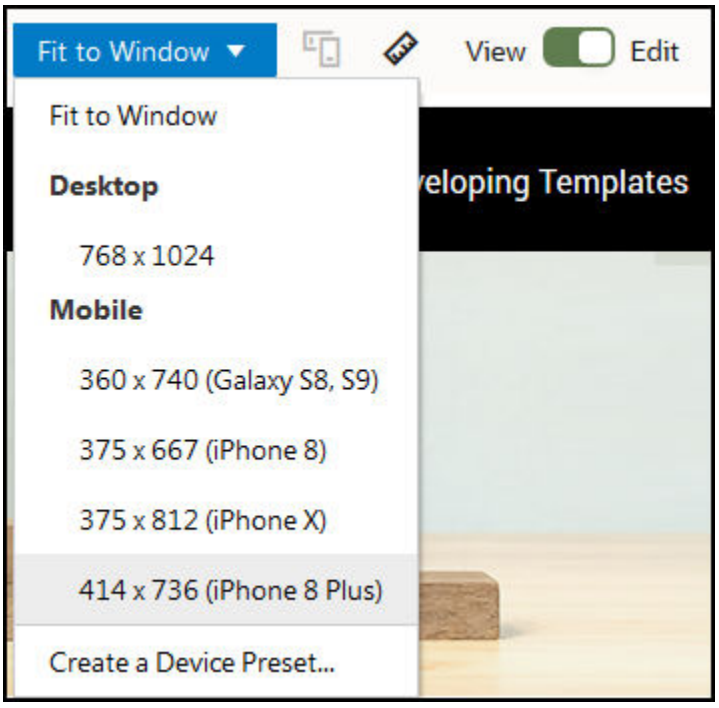
Note that the media rule below queries the screen size and will only take effect when the screen size is less than 767 pixels:

```
@media screen and (max-width: 767px) {
```

The code example for using css to allow inserted tables to be responsive is located in the default design.css of the provided StarterTheme. If building a site from the StarterTheme, the tables inserted into a paragraph slot will be responsive by default. To insert a table:

1. In Oracle Content Management, open a site and toggle to **Edit** mode.
2. Create a new update or choose an existing update to modify.
3. Drag a new paragraph component onto the page and click where you want to insert a table, or click in an existing paragraph where you want to insert a table.
4. On the rich text toolbar, click  and set the table properties. Make sure to select the first row as a header, and adjust the table width to work effectively on the smallest screen expected to be used. For example, if you expect the site to be viewed on a phone, the default width of 767 pixels will likely be too wide to display fully on a phone screen, even when stacked. You would want to set the table width to a smaller size, like 300 pixels, or set width to 100%.

When finished, toggle back to **View** mode and select a view option with a screen width of anything less than 767 pixels to preview the results. You must be in view mode, as a table does not behave responsively in Edit mode.



If you want to use responsive tables when building a site from a different theme, you will need to copy the code from the StarterTheme design.css to the design.css file of the theme you are using.

1. To copy the code from the StarterTheme design.css, click **Developer** in the side navigation of Oracle Content Management.
2. Click **View All Themes**.
3. Select the **StarterTheme** and click **Open**.
4. Click designs to open the folder and then click **default**.
5. Select the design.css file from the StarterTheme and click **Download**.
6. Open the file in a text editor and locate the section of the file that begins with the comment *An example CSS of how to render a table responsively*.
7. Select the code until the next comment and copy it.

```
/**
 * An example CSS of how to render a table responsively.
 * It enables stacking of row data on mobile devices.
 * Only do this for view mode (not for edit mode).
 *
 * On each cell rendered, it adds a user-defined attribute
 * 'data-label' with value matching the column header text.
 */
@media screen and (max-width: 767px) {
  .scs-paragraph:not(.scs-paragraph-edit) table {
    border: 0;
  }

  .scs-paragraph:not(.scs-paragraph-edit) table caption {
    font-size: 1.3em;
```

```
  }

  .scs-paragraph:not(.scs-paragraph-edit) table thead {
    border: none;
    clip: rect(0 0 0 0);
    height: 1px;
    margin: -1px;
    overflow: hidden;
    padding: 0;
    position: absolute;
    width: 1px;
  }

  .scs-paragraph:not(.scs-paragraph-edit) table tr {
    border-bottom: 3px solid #ddd;
    display: block;
    margin-bottom: .625em;
  }

  .scs-paragraph:not(.scs-paragraph-edit) table td {
    border-bottom: 1px solid #ddd;
    display: block;
    text-align: right;
  }

  .scs-paragraph:not(.scs-paragraph-edit) table td::before {
    content: attr(data-label);
    float: left;
    font-weight: bold;
  }

  .scs-paragraph:not(.scs-paragraph-edit) table td:last-child {
    border-bottom: 0;
  }
}
```

8. Repeat the steps to download the design.css file of the template you want to modify, open the template, and paste the copied code into the file.
9. Save the changes and upload the modified design.css file as a new revision to the theme you are modifying.

Customize Conversation List Styles

You can customize the style of a Conversation List component by adding selectors in the `design.css` file.

Use these CSS selectors to customize the style of the Conversation List component.

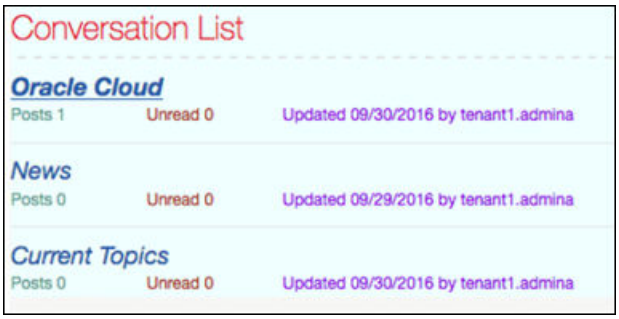
Selector Name	Description
`.scs-convo-list-cust .scs-convo-list-container`	Outmost DIV of the component
`.scs-convo-list-cust .scs-convo-list-title`	Title of a conversation in the list when it is selected

Selector Name	Description
.scs-convo-list-cust .scs-convo-list-line-separator	Separator between list title and the list
.scs-convo-list-cust .scs-convo-list-convo-title	Title of a conversation in the list
.scs-convo-list-cust .scs-convo-list-convo-line-separator	Separator between each conversation
.scs-convo-list-cust .scs-convo-list-active	Title of a conversation in the list when it is selected
.scs-convo-list-cust .scs-convo-list-convo-posts	Number of posts of a conversation
.scs-convo-list-cust .scs-convo-list-convo-unread	Number of unread messages of a conversation
.scs-convo-list-cust .scs-convo-list-convo-updated	Last updated date and of a conversation
.scs-convo-list-cust .scs-convo-list-no-convo-msg	Message when the list is empty
.scs-convo-list-cust .scs-convo-list-no-auth-msg	Message when the Conversation List is rendered in public site without user authorization

See Use Styles and Formatting in *Building Sites with Oracle Content Management*.

Example

This sample illustrates the use of customized CSS for type font, style, and color changes to a Conversation List.



The following code shows the customized CSS used to create the sample:

```
.scs-convo-list-cust .scs-convo-list-container {
    background-color: azure;
}

.scs-convo-list-cust .scs-convo-list-title {
    color: crimson;
}

.scs-convo-list-cust .scs-convo-list-line-separator {
    border-bottom: 2px dashed #dfe4e7;
}

.scs-convo-list-cust .scs-convo-list-convo-title {
```

```
    font-style: italic;
}

.scs-convo-list-cust .scs-convo-list-active {
    text-decoration: underline;
}

.scs-convo-list-cust .scs-convo-list-convo-posts {
    color: cadetblue;
    font-size: 12px;
}

.scs-convo-list-cust .scs-convo-list-convo-unread {
    color: brown;
    font-size: 12px;
    float: left;
}

.scs-convo-list-cust .scs-convo-list-convo-updated {
    color: blueviolet;
    font-size: 12px;
    clear:none;
}

.scs-convo-list-cust .scs-convo-list-no-convo-msg {
    font-size: 18px;
    color: darkorange;
}

.scs-convo-list-cust .scs-convo-list-no-auth-msg {
    font-size: 18px;
    color: red;
}
```

Customize Folder List and File List Styles

You can customize the styles of Folder List and File List components by adding selectors in the `design.css` file.

You can use a Folder List component to list the folders within a specified folder from your Oracle Content Management account. The folder list automatically communicates with a File List component and document manager on the page to display the files in a folder selected in the folder list.

You can use a File List component to provide a view of files from a specified folder in your Oracle Content Management account. The file list automatically communicates with a Folder List component on the page to display the files in a folder selected in the folder list.

Folder List CSS Selectors

Use these CSS selectors to customize the style of the Folder List component.

Selector name	Description
`.scs-folder-list-cust .scs-folder-list-container`	Outmost DIV of the component
`.scs-folder-list-cust .scs-folder-list-folder-title`	Folder name
`.scs-folder-list-cust .scs-folder-list-line-separator`	Separator between the folder name and the list of subfolders
`.scs-folder-list-cust .scs-folder-list-sub-folder-title`	Subfolder name
`.scs-folder-list-cust .scs-folder-list-sub-folder-title-active`	Subfolder name when it's selected
`.scs-folder-list-cust .scs-folder-list-no-folder-msg`	Message when there is no subfolder to display

File List CSS Selectors

Use these CSS selectors to customize the style of the File List component.

Selector name	Description
`.scs-file-list-cust .scs-file-list-container`	Outmost DIV of the component
`.scs-file-list-cust .scs-file-list-folder-title`	Folder name
`.scs-file-list-cust .scs-file-list-line-separator`	Separator between the folder name and the list of files
`.scs-file-list-cust .scs-file-list-row`	Row that contains the information for a file
`.scs-file-list-cust .scs-file-list-left-col`	File's thumbnail located in the left section of the component
`.scs-file-list-cust .scs-file-list-mid-col`	Middle section of the component, which contains the name, description, last modification, and size of a file
`.scs-file-list-cust .scs-file-list-file-title`	File name located in the middle section of the app
`.scs-file-list-cust .scs-file-list-file-desc`	File description located in the middle section of the app
`.scs-file-list-cust .scs-file-list-file-lastModified`	Last modification of the file
`.scs-file-list-cust .scs-file-list-file-size`	Size of the file with a vertical separator from the last modification
`.scs-file-list-cust .scs-file-list-file-size-no-sep`	Size of the file without a vertical separator (last modification not shown)
`.scs-file-list-cust .scs-file-list-right-col`	Right section of the app
`.scs-file-list-cust .scs-file-list-file-download-icon`	Download icon located in the right section of the app

Selector name	Description
`.scs-file-list-cust .scs-file-list-no-file-msg`	Message when there is no file to display

See File Lists and Folder Lists in *Building Sites with Oracle Content Management*.

Customize Social Bar Icons

You can create custom social icons to use in the social bar in a theme's default site.

Social icons that appear in the social bar in a site are determined by the design of the site's theme. If you change the theme for a site, the social icons change with the theme. Common social icons are included with Oracle Content Management themes for Facebook, Twitter, LinkedIn, Google+, and YouTube.

You can add custom social icons to the social icon component by editing the `design.json` and `design.css` files.

design.json File

In the `design.json` file, you can specify new icons using the name and class structure as shown in this sample code:

```
"componenticons": {
    "scs-socialbar" {
        "icons": [
            {
                "name": "COMP_ICON_FACEBOOK",
                "class": "scs-facebook-icon"
            },
            {
                "name": "COMP_ICON_LINKEDIN",
                "class": "scs-linkedin-icon"
            },

            {   "name": "COMP_ICON_TWITTER",
                "class": "scs-twitter-icon"
            },
            {
                "name": "COMP_ICON_GOOGLEPLUS",
                "class": "scs-googleplus-icon"
            },
            {
                "name": "COMP_ICON_YOUTUBE",
                "class" "scs-youtube-icon"
            }
        ]
    }
```

design.css File

In the `design.css` file, you can add new icons using the name and `url` specification as shown in this sample code:

```
.scs-facebook-icon {
  background-image: url("facebook.png"); }
.scs-twitter-icon {
  background-image: url("twitter.png"); }
.scs-linkedin-icon {
  background-image: url("linkedin.png"); }
```

Configure Interview Styling Extensions for Oracle Intelligent Advisor

You can style Oracle Intelligent Advisor (OIA) (formerly Oracle Policy Automation) interviews to unify the appearance with your corporate look and feel.

You can configure the following CSS class selectors in the `design.css` file. All of these class selectors have the prefix "`scs-opainterview-`".

Each class defines all styling for the specified component. The class has full control, and the existing OIA style won't be used.

Selector	Applies to	Description
interview	interview	The interview region comprises the entire interview content, including the header, footer and navigation area.
interviewContent	interview content	The interview content region includes the screen title and controls but excludes the header, footer and navigation area.
screenTitleBlock	screen title block	The screen title block composes the region that includes the screen title as well as any other widgets that are contained in that row, such as the screen drop-down list and/or next and back buttons.
screenTitle	screen title	The screen title region is just the region containing the screen title.
nextButton	next button	The next button.
backButton	back button	The back button.
restartButton	restart button	The restart button.
exitButton	exit button	The exit button.
header	header	The header region.
footer	footer	The footer region.
question	question text	Styling for question text.
control	container for controls	Styling for the element that contains controls.
label	label control	Styling for label controls.
controlError	all controls	Styling for error text container.
controlErrorText	all controls	Styling for error text span.

Selector	Applies to	Description
textInput	single line text box, password and masked	Styling for text input controls.
textAreaInput	multi-line text box	Styling for multiline text input controls.
calendarInput	calendar	Styling for calendar input controls. Supports an `iconColor` field that allows the color of the calendar icon to be changed, and a `keepIcon` field that indicates whether the calendar icon should be displayed.
dropDownInput	drop-down list	Styling for drop-down list input controls.
filterDropDownInput	filtered drop-down list	Styling for filtered drop-down list input controls. Supports an `iconColor` field that allows the color of the drop-down arrow to be changed.
listInput	fixed list	Styling for fixed list input controls.
radioInput	radio buttons	Styling for radio button input controls. They can be styled with `borderColor` and `fillColor` options. The `iconType` property can change the type of icon used. Currently `'tick'` and `'fill'` are the only supported alternate options.
checkboxInput	checkbox	Styling for checkbox input controls. They can be styled with `borderColor` and `fillColor` options. The `iconType` property can change the type of icon used. Currently `'square'` and `'fill'` are the only supported alternate options.
autoCompleteInput	custom search	Styling for the `autocomplete` field when a `customSearch` extension is used.
captchaInput	CAPTCHA input field	Styling for the input field that the user is entering the CAPTCHA into.
signatureInput	signature control	Styling for signature controls. Supports an additional `inkColor`field that allows changing the pen ink for the signature.
explanationHeader	explanation control	Styling for the top level expandable header for explanation controls.
explanationText	explanation control	Styling for the expanded explanation control text.
signatureClearButton	signature control	Styling for the clear button on a signature control.
uploadAddButton	upload control	Styling for the add button on the upload control.
entityRemoveButton	entity collect control	The entity collect control.

Examples with Style Extensions Defined in design.css

```
.scs-opainterview-interviewContent {
   background-color: beige;
}
.scs-opainterview-screenTitleBlock {
   background-color: bisque;
}
```

```
.scs-opainterview-screenTitle {
    font-style: italic;
    font-size: 20px;
}
.scs-opainterview-nextButton {
    color: darkgreen;
}
.scs-opainterview-backButton {
    color: crimson;
}
.scs-opainterview-question {
    color: green;
}
.scs-opainterview-control {
    background-color: cornflowerblue;
}
.scs-opainterview-label {
    color:aqua;
}
.scs-opainterview-textInput {
    color: red;
    cursor:crosshair;
}
.scs-opainterview-radioInput {
    background-color: pink;
}
.scs-opainterview-checkboxInput {
    cursor: pointer;
}
```

20
Understand Background Use

You can specify the background color and image for site pages and for individual slots on a page. Backgrounds for pages, slots, and components layer on top of one another. For example, if you specify a background for a slot, it is layered above the background specified for the page.

About Background and Themes

The background feature is primarily intended for use in site pages and slots. Backgrounds set for pages and slots are meant to be configured by users, not developers, at the theme level.

Background effects can be configured in theme slots, however, these effects may override any background effects set for pages when editing a site.

The background feature does not alter themes or constituent theme files. A theme can't be altered by the background settings for a page, and another page based on the same layout in a site won't inherit any of the background settings for the source page. Adding or changing the background through a theme requires an update to the theme.

Users can configure different backgrounds on every page of the site, and also on different pages that use the same layout. This would not be possible within a theme. Although a theme may specify a background for a page, users' background settings can override this specification. (Background settings will only apply overrides for a particular page; it does not in any way modify the theme itself.)

Be careful to avoid having the theme override background styles that will be set up in site pages and slots by users. This can happen in several ways:

- Page background settings in a site can be overridden by a theme when you use element-based `"style"` attributes on the `<body>` and the slots.
- Page background settings in a site can be overridden by a theme when you mark theme background styles as `"!important"` in the theme cascading style sheet (CSS) file.

See Change the Background or Theme in *Building Sites with Oracle Content Management*.

How Backgrounds Are Implemented

To implement the page and slot backgrounds feature, Oracle Content Management dynamically creates a CSS stylesheet in the `<head>` of a page.

The selector for the styles is a tag-based selector (`body`) for the page background settings. For the slot background settings and ID-based selector, the slot ID is used.

For example, setting a background color for a page might yield the following CSS markup in the <head> of the page:

```
body
{
     background-color: #fa7c9d;
}
```

Similarly, setting a background image on a slot might yield the following CSS markup in the <head> of the page:

```
#PageFooter
{
     background-image: url("footer_image.png");
}
```

This implementation means that styles directly specified in the "style" attribute of the body tag or the slot element can override the settings configured in the stylesheet in the <head> code.

> **Important:**
>
> Theme developers should take care not to override background settings with element-based styles.

See Change the Background or Theme in *Building Sites with Oracle Content Management*.

Where Settings Are Stored

When pages are rendered, background settings are dynamically written to "style" tags in the <head> code for the page.

Background settings are persisted in the page model files (for example, <*pageid*>.json). In particular, the page background settings are stored in the properties.styles section, and the slot background settings are stored in the slots[<*slot_id*>].styles section.

Background settings are stored in the page JSON files, specifically within "styles" as shown in this representative sample.

```
{
     "properties":
     {
           "pageLayout" : "oneslot.htm",
           "styles": [
                "background-image: url([!--$SCS_CONTENT_URL--]/
background_image.gif)",
                "background-position: center",
                "background-size: auto",
                "background-repeat: repeat",
```

```
                "background-origin: padding-box",
                "background-clip: border-box"
            ]
        },

        "slots":
        {
            "slot100":
            {
                "components":
                [
                        "dedda3a8-615d-44ad-ad71-51f2fa465cef",
                        "95eb0fd6-bcfc-4e5e-ba67-a5c8c5d9c315"
                ],
                    "grid": "<div class=\"scs-row\"><div class="scs-
col\"style=\"width: 50%;\">
                    <div id=\"dedda3a8-615d-44ad-ad71-51f2fa465ced\">
                    </div>
                    </div>
                    <div class=\"scs-col\" style=\"width: 50%;\">
                    <div id=\:95eb0fd6-bcfc-4e5e-ba67-a5c8c5d9c315\">
                    </div>
                    </div>
                    </div>",
                    "styles": [
                        "background-image: url([!--$SCS_CONTENT_URL--]/
oracle-cloudworld.jpg)",
                        "background-position: center",
                        "background-size: cover",
                        "background-repeat: no-repeat",
                        "background-origin: padding-box",
                        "background-clip: border-box",
                        "background-color: transparent"
                    ]
            }
        },

        "componentInstances":
        {
            "dedda3a8-615d-44ad-ad71-51f2fa465cef":
            {
                "type": "scs-title",
                "data": {
                    "alignment": "fill",
                    "backgroundColor": "",
                    "borderColor": "#808080",
                    "borderRadius": 0,
                    "borderStyle": "none",
                    "borderWidth": 1,
                    "fontColor": "#333333",
                    "fontFamily": "'Helvetica Neue', Helvetica, Arial, sans-
serif",
                    "fontSize": 24,
                    "marginBottom": 5,
                    "marginLeft": 5,
```

```
                    "marginRight": 5,
                    "marginTop": 5,
                    "styleClass": "",
                    "useStyleClass": "true",
                    "userText": "<div>My Test Title</div>\n",
                    "width": 400
               }
          },
          "95eb0fd6-bcfc-4e5e-ba67-a5c8c5d9c315":
          {
               "type": "scs-image",
               "data": {
                    "styleClass": "",
                    "useStyleClass": "true",
                    "imageUrl": "[!--$SCS_CONTENT_URL--]/example.jpg",
                    "defaultImageUrl": "/components/comp/images/
default_image.png",
                    "style": "",
                    "imageWidth": 0,
                    "borderStyle": "none",
                    "borderWidth": 1,
                    "borderColor": "black",
                    "borderRadius": 0,
                    "altText": "My Image",
                    "title": "My Title",
                    "caption": "My Caption",
                    "imageAlignment": "center",
                    "imageHref": "",
                    "imageTarget": "_self",
                    "marginTop": 0,
                    "marginRight": 0,
                    "marginBottom": 0,
                    "marginLeft": 0,
                    "linkType": "scs-link-no-link"
               }
          }
     }
}
```

21

Set Triggers and Actions

Communication between components (including components rendered in inline frames) can be configured so a trigger within a component calls an action on another component.

About Triggers and Actions

Communication between components (including components rendered in inline frames) can be configured so a **trigger** within a component calls an **action** on another component.

Triggers are part of Oracle Content Management intercomponent communication. Any component can raise any number of triggers. The component can provide a payload for a trigger, which then is passed to any action that is executed when the trigger is raised. You can select what actions will be executed for each trigger. Components that are built to work together can automatically raise triggers to execute actions on the other component without user interaction.

The basic process involves:

1. Registering triggers
2. Raising triggers
3. Registering actions
4. Executing actions to verify the setup

For example, you can use the Button component to perform one or more actions such as showing or hiding page components and showing messages. You could have a list of business office locations in one component and when a location in the list is clicked, then details about the location are displayed in another component.

For components you customize, the triggers and actions are part of the component registration data and not part of the component implementation. In the registration data there is a `"triggers": [],` and `"actions":[],` entry that contains the list of triggers and actions the component supports. The actual syntax is the same as for local and remote components, only the location and how it's retrieved is different.

See Use Triggers and Actions in *Building Sites with Oracle Content Management*.

Set Triggers

A component can include triggers that will execute actions in other components. You must register triggers to be raised by components.

The component provides a payload for a trigger, which is passed to any action that is executed when the trigger is raised. You can select what actions will be executed for each

trigger. Components that are built to work together can automatically raise triggers to execute actions without user interaction.

Register Triggers

For a custom component, triggers are registered as part of the registration data for the component. To add a trigger, update the `"triggers"` property array with each trigger the component supports. You also must specify the payload the trigger supports so that the user interface can be created to allow users to map values within the payload to properties supported by the action.

1. Edit the `appinfo.json` file and review the `"triggers":[],` entry:

```
"triggers": [{
    "triggerName": "helloWorldWhoAreYou",
    "triggerDescription": "Show Who I Am",
    "triggerPayload": [{
      "name": "whoAreYou",
      "displayName": "Who I Am"
   }]
}],
```

2. Sync the file to the sites server.

In this sample trigger entry, you've defined a `triggerName` (`"helloWorldWhoAreYou"`). The name value must be unique. You've then given the trigger a description (`"Show Who I Am"`), which is used by the user interface dialog to display your trigger. Finally, you've defined a single value payload for the trigger; users will be able to select entries in this payload and map them to fields in the action.

Once a trigger is registered, you should be able to see and select the trigger when you go to the **Link** tab in the Settings panel for your component.

Raise Triggers

Triggers can be raised at any point by a component. Typically, a trigger is raised by a user interaction, such as clicking a button or selecting a row in a table. A component can raise the trigger based on any criteria, for example, when data changes because of a REST API call. You can execute any number of actions when a trigger is raised.

Here's an example of how to raise a trigger:

1. Edit the `render.js` file and add a JavaScript function in the `viewModel` object that will call the Sites SDK to raise the trigger.

```
self.raiseTrigger = function (triggerName) {
  SitesSDK.publish(SitesSDK.MESSAGE_TYPES.TRIGGER_ACTIONS, {
    'triggerName': 'helloWorldWhoAreYou',
    'triggerPayload': { "whoAreYou": "This is " + self.whoAreYou()
+ "!"}
  });
};
```

2. Add an entry in the user interface to call the function to raise the trigger (`-edit template.html`) and a button before the `</div>`.

```
<button data-bind="click raiseTrigger">Who Am I?</button>
```

3. Sync or upload the `render.js` file to your Oracle Content Management instance server.

In the `ViewModel` object, you created a JavaScript function that is called when the button is clicked. This function calls the Sites SDK to tell it to trigger all the actions defined for this trigger `"helloWorldWhoAreYou"`. It also passes through a `triggerPayload` that has a single field, `"whoAreYou"`. These values `"helloWorldWhoAreYou"` and `"whoAreYou"` match those you entered when you registered the trigger in the previous step.

> **Note:**
>
> There is no predefined order to when an action is executed. Although each action will be called in the order it is listed, there is no wait for it to complete before the next action is called. If an action makes an asynchronous call, it may not complete before the next action is executed.

Set Actions

You can set a component to leverage action registration so that it can be dropped on a page that will execute actions within your component.

Register Actions

Actions are called on components when triggers are raised. A component can register any number of actions and also define the payload the action supports. When a user selects an action, they can populate the payload to be passed to the action.

As with registering triggers, you can register actions that your component supports in the `appinfo.json` registration data for your theme.

Here's an example of how to register an action:

1. Edit the `appinfo.json` file for your component and update the `"actions":[],` entry.

```
"actions": [{
  "actionName": "helloWorldChangeWhoIAm",
  "actionDescription": "Change Who I Am",
  "actionPayload": [{
    "name": "whoAreYou",
    "description": "Who are you?",
    "type": {
      "ojComponent": {
       "component": "ojInputText"}
      },
    },
    "value": ""
  }]
}]
```

2. Once registered, the action will be visible in the action dialog that's invoked when you click a trigger on the **Link** tab on the Setting panel for your component.

Execute Actions

Once an action is registered, you'll be able to drop components onto the page that execute actions within the component. For a component to execute an action, it must listen for the `EXECUTE_ACTION` message. This message also includes the payload passed to the action from which you must extract the expected values.

As an example, to listen for the `EXECUTE_ACTION` message, edit the `render.js` file and update the `ViewModel` object with these entries:

```
self.executeActionListener = function (args) {
  // get action and payload
  var payload = $.isArray(args.payload) ? args.payload[0] : {},
        action = args.action,
        actionName = action && action.actionName;

  // handle 'helloWorldChangeWhoIAm' actions
  if ((actionName === 'helloWorldChangeWhoIAm') && (payload.name ===
'whoAreYou')) {
    self.whoAreYou(payload.value);
  }
};
```

This creates a JavaScript function to execute the action, then uses the Sites SDK to call the function whenever the `EXECUTE_ACTION` message is raised.

The action will be called whenever an `EXECUTE_ACTION` message is raised, and it's up to the component to handle only actions it is designed to handle. To do this, you must check the name of the action to ensure it is one you can handle.

The payload for the action is an array of values. In the example, it's assumed that the value is the first entry in the array. Typically, you must find the payload values you care about from the array.

> **Note:**
>
> Because the action listener is a callback, you should use JavaScript Closure or appropriately bind the function to ensure you have access to your `ViewModel` when the function is executed.

22

Develop Site Templates

A site template package contains the development version of a site, a theme with page layouts, style and navigation, and associated components used in the site. Oracle Content Management provides a set of site templates to be used for creating sites and as starting points for creating custom site templates.

About Site Templates

A site template contains all the pieces that users need to start creating a website, including a site with sample pages and content, a theme with styling, navigation, assets such as images, and associated components.

Oracle Content Management provides a number of site templates for use in creating sites. These site templates are typically installed by your administrator when the service is initialized. See Configure Sites Settings in *Administering Oracle Content Management*.

Whenever you create a new site, you must select a site template. Templates combine themes with sites and components to drive a function or solution, such as a partner portal or a marketing campaign.

For a list of out-of-the-box site templates, see Understand Site Templates.

While both developers and users can create new site templates and modify and replace existing site templates, one of the main tasks for developers is to design new site templates. This process basically consists of these steps.

1. Create a new site template by copying an existing site template, such as the JET Starter Template. This also gets you the theme associated with the site template.

   For example:

   ```
   cec create-template My_JET_Template -f JETStarterTemplate
   ```

2. Export the site template in a `.zip` file to your development environment.
3. Open the files in the site template package and make your changes.
4. Create a revised site template package in a `.zip` file.

5. Use the Oracle Content Management interface to import the new site template to your instance.
6. Share the site template so others can use it.

An alternative is to use the Oracle Content Management interface to modify the site template by adding and modifying page layouts and assets, expand the site structure, add components to site pages, and add seed content you want to appear in any sites that use the theme in this site template.

See also Manage Site Templates.

Basic Site Template Structure

The basic structure of a site template includes a site (with assets, layouts, pages, and content), an associated theme, and any custom components.

When you create a site, you must choose a site template to provide the site structure and initial content, a theme with design and layout specifications, and any custom components.

A site template is organized in a specific structure, as illustrated in this example showing basic folders and files.

```
template_name
    components
        component_name
            assets
                render.js
                settings.html
            _folder.json
            _folder_icon.jpg
            appinfo.json
    template
        assets
        content
        layouts
        pages
            100.json
            200.json
            300.json
            400.json
        variants
        _folder.json
        _folder_icon.png
        componentsused.json
        controller.html
        siteinfo.json
        structure.json
    theme
        assets
            css
                main.css
            js
                topnav.js
        designs
```

```
        default
            design.css
            design.json
            facebook.png
            googleplus.png
            linkedin.png
            twitter.png
            youtube.pgn
    layouts
        index.html
    responsepages
        404.html
    _folder.json
    _folder_icon.png
    components.json
    viewport.json
```

Component and theme folders and files are described elsewhere in this guide. See About Developing Components and Basic Theme Structure.

> **Notes:**
>
> - Generally a theme is shared between site templates, unless you use the JET Starter Template or Starter Template, which uses a copy of a theme. See Work with a Starter Template.
> - The theme no longer contains the site. The site folders and files are in the `/template` folder.

The *`template_name`*`/template` folder contains the folders and files for the site.

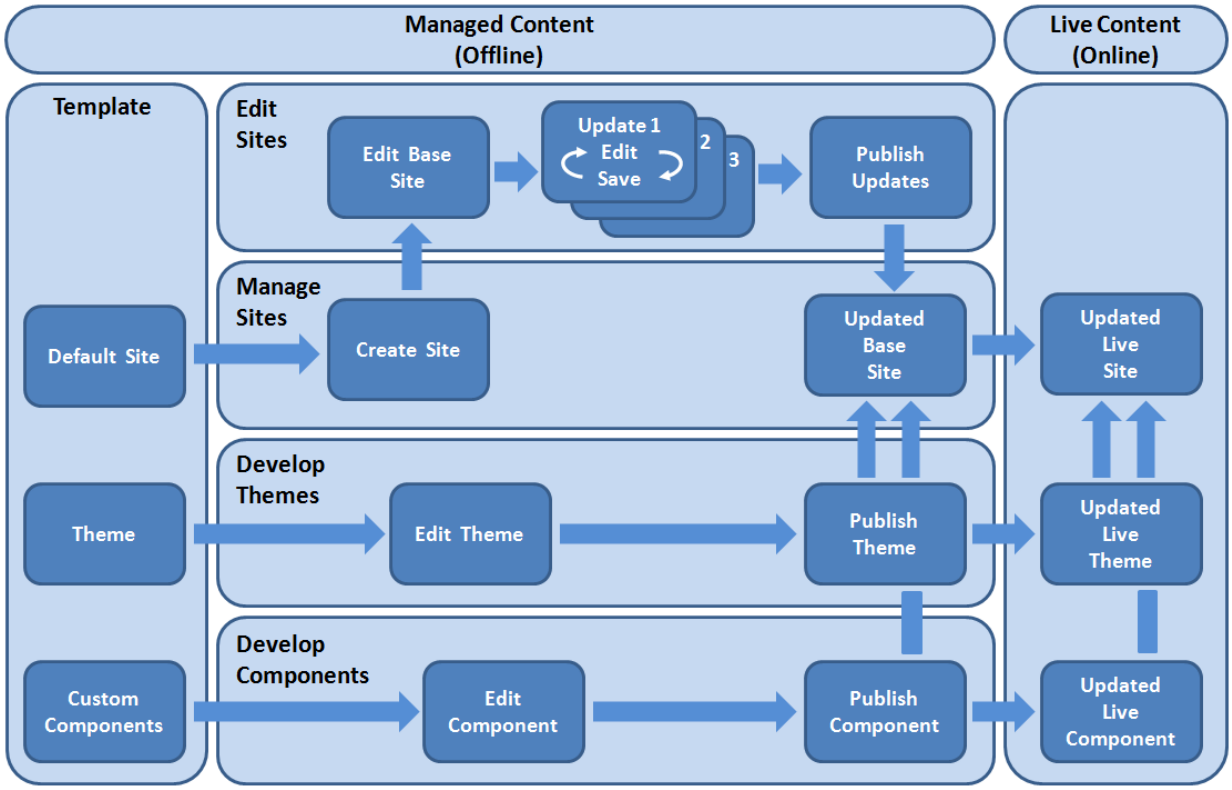
- `assets`: Contains images that are displayed in the site template details page in the user interface.
- `content`: Contains managed content used in the site.
- `layouts`: Not used at this time.
- `pages`: Contains all the page JSON files with data. Uses the format *`nnn`*`.json`, where *`nnn`* is the page ID.
- `variants`: Contains details of all the updates for the site.
- `_folder.json`: Contains metadata for the site template, such as site author, site name, item GUID, short and long site descriptions.
- `_folder_icon.png`: Represents the site in the user interface.
- `componentsused.json`: (Deprecated.) Records the custom components that are used, if any, within the site. Maintained only for backwards compatibility.
- `controller.html`: Contains the key code that displays the site in a browser. If you want to make changes to this file, Oracle recommends that to do this through the site settings in the interface. You can modify the file offline. See Customize the Controller File.

- `siteinfo.json`: Identifies the site name and the name of the associated theme along with other metadata for the site. Don't modify this file.
- `structure.json`: Defines the hierarchy of the site for pages (parent and child pages). The Render API can be used to draw out the tree structure when setting up navigation for the site. See Site Navigation and Render API Reference.

Create a Site Template

If you have a site that you want to use as a starting point for other sites, you can create a site template from that site. You also can create a new site template by copying an existing site template and making changes to the copy.

If you create a site template from an existing site, the new site template uses a copy of the site as its default site. The site template references the theme used by the site and any custom components used in the site pages. The theme and custom components are not copied to the site template, but are referenced in the same way they are by the site. The site template reflects the site used to create it at the time the site template is created. Further changes to the site used to create the site template are not reflected in the site stored with the site template.



If you create a new site template by copying an existing site template and renaming the copy, you make changes to the copy. Note that when you copy a site template, sharing information for the site template isn't copied.

Don't use the following names for site templates, themes, components, sites, or site pages: authsite, content, pages, scstemplate_*, _comps, _components, _compsdelivery, _idcservice , _sitescloud, _sitesclouddelivery, _themes,

_themesdelivery. Although you can use the following names for site pages, don't use them for site templates, themes, components, or sites: documents, sites.

If you want to create your own custom site template (with site and theme), it's best to use the JET Starter Template or Starter Template provided by Oracle Content Management, which contains basic elements for a site and an associated theme. A starter site template includes information and instructions written into the site pages to help you explore how to layout and design a site and theme in a custom site template.

See Manage Site Templates and Work with a Starter Template.

Export a Site Template

You can export a site template to modify it offline and then import it either as a new site template or to replace the existing site template. You can also export a site template to move it to another Oracle Content Management instance and import it there.

When you export a site template, you essentially copy the site template to a folder in Oracle Content Management as a single `.zip` file. You can download the site template package directly from the folder to unpack and work with the individual files. When you are done working with the site template files, create a `.zip` file that contains the site template package and import it into Oracle Content Management to overwrite the original site template or create a new one.

> **Note:**
>
> When you export a site template, sharing information for the site template isn't included.

To export a site template:

1. In the Oracle Content Management side navigation, click **Developer**.

   The **Developer** page is displayed.

2. Click **View all Templates**.

   A list of existing site templates is displayed.

3. Select a site template and choose **Export** in the right-click menu or click  in the actions bar.

4. Navigate to a folder or create new folder by clicking **Create**, providing a name and an optional description, and clicking **Create**.

   To open a folder, click the folder icon or the folder name.

5. Select a folder by clicking the checkbox for the associated folder and click **OK**.

   A site template package file is created in the selected folder with the site template name and a `.zip` extension.

Import a Site Template

You can export a site template to modify it offline and then import it either as a new site template or to replace the existing site template. You can also export a site template to move it to another Oracle Content Management instance and import it there.

When you export a site template, you essentially copy the site template to a folder in the Oracle Content Management as a single `.zip` file. You can download the site template package directly from the folder to unpack and work with the individual files. When you are done working with the site template files, create a `.zip` file that contains the site template package and import it into Oracle Content Management and overwrite the original site template or create a new one.

To import a site template package:

1. In the Oracle Content Management side navigation, click **Developer**

   The **Developer** page is displayed.

2. Click **View all Templates**.

   A list of existing site templates is displayed.

3. Click **Create** and choose **Import a template package**.

4. If you have uploaded the site template package, navigate to the folder that contains the site template package. To open a folder, click the folder icon or the folder name.

   If you have not yet uploaded the site template package:

   a. Navigate to the folder where you want to upload the site template package or create a folder by clicking **New**, providing a name and an optional description, and clicking **Create.**

   b. Click **Upload**.

   c. Locate and select the site template package, then click **Open**.

      A progress bar shows the file name and the upload status.

5. Select a site template package by clicking the checkbox next to the file name and click **OK**.

   If there are no conflicts between the contents of the imported site template and any existing site templates, themes, or custom components, new Oracle Content Management folders are created for the site template, its associated theme, and any custom components.

6. If the site template, theme, or custom component names or IDs exist, you are prompted to resolve the conflicts.

   Depending on the nature of the conflict, you are given the option to create a new site template, theme, or custom component, or in some cases, you can overwrite the existing site template, theme or custom component with the imported version.

Work with a Starter Site Template

You use a copy of a starter site template provided by Oracle Content Management to create a new site, with a theme and custom components.

A starter site template collects all the pieces you need to build a custom website in one package: the default site, layout, navigation, sample content, theme, associated content items, and so on. Different from other site templates provided with Oracle Content Management, a starter site template provides you with an easy-to-use basic framework for creating a new site, including a new theme.

> **Note:**
>
> A theme is usually shared between site templates, but a starter site template uses a copy of a theme.

The starter site templates, StarterTemplate and JETStarterTemplate, are provided along with other site templates in Oracle Content Management when your administrator enables site templates during installation and configuration of the service. The Oracle JavaScript Extension Toolkit (JET) starter site template includes the latest JET styling for site templates, incorporates some page content (as JET components), and provides starter components for building JET-based site templates and sites.

Basic Process

Here's the basic process for working with a starter site template:

1. Create a new site and select the starter site template. A new starter theme is created along with the new site.

   > **Note:**
   >
   > Choose the site name carefully. The name you give the site is duplicated as the name of the new theme, and the theme will be visible to users once you publish the site. You can't change the name of the theme after it is created.

   The starter site and theme contain a set of folders and files that are required to start site and theme development.
2. Sync the new theme to your desktop. You can work on your desktop to extend and customize the theme with layouts and static assets. Because this is a copy of a theme and uses a name specific to your site template, you can make changes to the theme without affecting the source theme.
3. Open the site in Site Builder. You can use Site Builder to add to the site structure, add components and interactions to pages, and add content that will become part of the default site of the new site template. You can reuse or remove the site content provided in the starter site template.
4. When the site and theme are ready, sync the changes with your Oracle Content Management instance, then create a new site template from the selected site.
5. Share the site template so others can use it.

Create a Template Using a Starter Template

To use a starter site template:

1. In the Oracle Content Management side navigation, click **Sites**.

A list of existing sites is displayed.

2. Click **Create**.
3. In the Create Site dialog, select the starter site template to use as the basis for your site.

   The new site uses the theme provided with the starter site template, renamed to match the new site name.
4. In the dialog, enter a name for the site. This name is used in the site URL. You can use letters, numbers, underscores (_), and hyphens (-). If you enter a space, it's automatically replaced with an underscore.

   Don’t use the following names for site templates, themes, components, sites, or site pages: authsite, content, pages, scstemplate_*, _comps, _components, _compsdelivery, _idcservice , _sitescloud, _sitesclouddelivery, _themes, _themesdelivery. Although you can use the following names for site pages, don’t use them for site templates, themes, components, or sites: documents, sites.

> **Note:**
>
> The path for an Oracle Content Management site URL is case-sensitive. Case in the query or fragment strings is managed by developers in their custom code.
>
> ```
> https://host[:port][/]path[?query][#fragment]
> _____________________/____________________________/
>   Location              Data
> ```

5. Optionally, enter a description for the site.
6. When you’re ready, click **Create**.

   A progress bar shows the new site name and creation status. When the site is created, the name appears in the list of sites. Its initial status is offline.

   To quickly find your newly created site in the list, sort the list by **Last Updated**. The site you just created will appear at the top of the list.

> **Note:**
>
> You are automatically assigned the role of manager for the site you created.

7. Use the desktop app to sync the theme to your desktop.

   You should now see the folder hierarchies and files for the theme.
8. The starter theme contains a minimal set of folders and files, like these:

```
theme
    assets
        css
            main.css
```

```
        js
            topnav.js
    designs
        default
            design.css
            design.json
            facebook.png
            googleplus.png
            linkedin.png
            twitter.png
            youtube.png
            x-close.png
layouts
        index.html
    resonsepages
        404.html
    _folder.json
    _folder_icon.jpg
    components.json
    viewports.json
```

The x-close.png file contains the default close icon for the cookie consent popup.

The `/layouts` folder contains a starter page layout file (`index.html`) with the following contents:

- A set of HTML tags that allow the file to be used as a page layout.
- A single slot that has seeded text with instructions, such as how to sync the theme to your desktop, how to add a new page layout, how to add components to the page layout, and how to build site hierarchy using the new page layout.
- A simple JavaScript navigation file that provides an example of how to use the renderer API JavaScript functions and objects. The rendered API is needed for traversing the site hierarchy and generating required HTML markup to allow navigation within the site.

See also Basic Theme Structure.

9. When you're finished modifying the theme, sync the theme folders and files to your Oracle Content Management instance. To see how your changes to the theme look and behave in a site, open the site in Site Builder. This will likely be an iterative process.
10. To view or modify the site, select the site and click **Open** in the right-click menu or click  in the actions bar.
11. Toggle the editor mode to **Edit** so you can make changes to the site. You can modify existing pages and add new pages using the page layout available in the starter themes.

> **Note:**
>
> If you are familiar with the page layout structure and usage, you can delete sections provided by the starter site template that you don't want and switch the layout on the sections that you do want to one of the new layouts.

12. When you are finished making changes to the site, **Save** the site, then click **Publish** to merge the update to the base site.
13. Select the site and create a site template based on the site. This pulls in the assets and theme for the new site template.
14. Share the site template with members you want to be able to use the site template.

Create a Site Template from Bootstrap or a Website Design Template

The open architecture of Oracle Content Management means you can use work done in other coding frameworks such as Foundation or Bootstrap. With a few changes, you can turn a Bootstrap site template into a theme and make it part of an Oracle Content Management site template.

Basic Process

Here's an overview of the steps described in detail in the sections that follow:

1. Prerequisites
2. Create a Site
3. Synchronize the Theme Folder
4. Set Up the Basic Theme
5. Update the Site Pages
6. Update Navigation
7. Update Site Layouts
8. Publish the Site
9. Create the New Template

Prerequisites

- Have the Oracle Content Management desktop app for synchronizing folders and files to your local computer set up and running.
- Download the Bootstrap site template theme folders, files, and content to your local computer and have them ready for use.

Create a Site

Create a site from an Oracle Content Management starter site template:

1. In the side navigation, click **Sites**.

   A list of existing sites is displayed.
2. Click **Create**.
3. In the Create Site dialog, select **JET Starter Template** or **Starter Template** to use as the basis for your site.

4. In the dialog, enter a name for the site. This name is used in the site URL. You can use letters, numbers, underscores (_), and hyphens (-). If you enter a space, it's automatically replaced with a hyphen.

   Don’t use the following names for site templates, themes, components, sites, or site pages: authsite, content, pages, scstemplate_*, _comps, _components, _compsdelivery, _idcservice , _sitescloud, _sitesclouddelivery, _themes, _themesdelivery. Although you can use the following names for site pages, don’t use them for site templates, themes, components, or sites: documents, sites.

   > **Note:**
   >
   > The path for an Oracle Content Management site URL is case-sensitive. Case in the query or fragment strings is managed by developers in their custom code.
   >
   > ```
   > https://host[:port][/]path[?query][#fragment]
   > _____________________/_____________________________/
   >   Location                      Data
   > ```

5. Optionally, enter a description for the site.
6. When you’re ready, click **Create**.

   A progress bar shows the new site name and creation status. When the site is created, the name appears in the list of sites. Its initial status is offline.

   To quickly find your newly created site in the list, sort the list by **Last Updated**. The site you just created will appear at the top of the list.

Synchronize the Theme Folder

When you create a site from a starter site template, a copy of the starter site template theme is created and named with the site name followed by the theme name. For example, the theme for `My_New_Site` is `My_New_SiteTheme`.

Use the desktop app to synchronize the theme folder and files for the site to your local computer. See Get Started with sync in *Collaborating on Documents with Oracle Content Management*.

You should now see the site template theme folder hierarchy and files on your local desktop. Here’s an example:

```
theme_name
    assets
        css
            main.css
        js
            topnav.js
    designs
        default
            design.css
            design.json
            facebook.png
            googleplus.png
```

```
        linkedin.png
        twitter.png
        youtube.pgn
layouts
    index.html
responsepages
    404.html
_folder.json
_folder_icon.png
components.json
viewport.json
```

Set Up the Basic Theme

1. Copy these Bootstrap files into the synchronized theme folders on your local desktop to overlay the existing files.
   - `html` files go into the *`theme_name`*`/layouts` folder
   - `css` files go into the *`theme_name`*`/assets/css` folder
   - `js` files go into the *`theme_name`*`/assets/js` folder
   - `image` files go into the *`theme_name`*`/assets/images` folder, which may be grouped in subfolders with images for background, footer, people, and so on
2. Modify the `html` files in the `layout` folder to update relative paths and add required elements. For a typical Bootstrap theme there will be many relative paths to the `/assets` folder, so you must modify them to point to the theme folder.

   Fix the paths for the `css`, `js`, and `images` folders to use:

   ```
   _scs_theme_root_/assets/css/
   _scs_theme_root_/assets/js/
   _scs_theme_root_/assets/images/
   ```

   > **Note:**
   >
   > Once this step is completed, the `_scs_theme_root` part will automatically adjust to the environment the theme is being used in.

3. There are three requirements for each Oracle Content Management layout:
   a. Include the following tags in the `<head>` tag of the `html` file:
      - `<!--$SCS_RENDER_INFO-->`
      - `<!--$SCS_SITE_HEADER-->`
      - `<!--$SCS_PAGE_HEADER-->`
   b. Include the renderer script at the end of the layout files, just inside the `<body>` tag. Both of these paths automatically adjust in Site Builder and the runtime environments.

      ```
      <script data-main="/_sitescloud/renderer/renderer.js" src="/
      _sitescloud/renderer/require.js"></script>
      ```

c. Include the following tags after the include of the `renderer.js` file:

- `<!--$SCS_PAGE_FOOTER-->`
- `<!--$SCS_SITE_FOOTER-->`

Verify that the modified theme files are synchronized with the site in Oracle Content Management.

Update the Site Pages

When you open the site in Site Builder, you'll initially see the pages that are in the default site. You can delete the pages you don't want and switch the layout on the pages you want to keep to one of the new layouts.

1. In the side navigation, click **Sites**.

   A list of existing sites is displayed.

2. Select the site and choose **Open** in the right-click menu or click  in the actions bar.
3. Enter a name for the update and an optional description, then click **Create**.

   For the update name you can use letters, numbers, underscores (_), and hyphens (-). If you enter a space, it's automatically replaced with a hyphen.

   If you already have updates to the site, select an update from the list and click  .
4. Site Builder opens in preview mode. To make changes or to use the navigation options in the sidebar, make sure that the Edit switch  is set to **Edit**.
5. To edit a particular page, choose the page using the site tree in the sidebar or using the site's own navigation.
6. To remove a page you don't want, select the page and click  .
7. To add a new page, click **Add Page**. You can reposition the page in the site tree by dragging and dropping it.
8. To change the layout associated with a page, choose the page in the site tree and click  to display the page settings.

   Go to the **Page Layout** field and select a different layout from the menu. The number and type of page layouts depends on the theme associated with your site.
9. **Save** to save your changes to the current update. You can continue working in the current update or create new updates if needed.

Update Navigation

When you look at the navigation in the site preview, it doesn't match the current hierarchy because of the hard-coded navigation in the layout from the Bootstrap theme.

Update the navigation to replace the hard-coded code in the Bootstrap theme with dynamically generated code from the site hierarchy.

Edit your local synchronized copies of the theme files.

1. Remove the hard-coded navigation code from the layouts. Here's an example of the hard-coded navigation that would have to be removed from a typical Bootstrap theme. This is a typical header section, with the logo, the 'Toggle Navigation' parts for the

'Hamburger' menu when the page is too narrow (the responsive part) and the hard-coded page navigation for the other pages.

```
<header id="header" class="header navbar-fixed-top">
        <div class="container">
            <h1 class="logo">
                <a href="index.html"><span class="text">Velocity</
span></a>
            </h1><!--//logo-->
            <nav class="main-nav navbar-right" role="navigation">
                <div class="navbar-header">
                    <button class="navbar-toggle" type="button"
data-toggle="collapse" data-target="#navbar-collapse">
                        <span class="sr-only">Toggle navigation</
span>
                        <span class="icon-bar"></span>
                        <span class="icon-bar"></span>
                        <span class="icon-bar"></span>
                    </button><!--//nav-toggle-->
                </div><!--//navbar-header-->
                <div id="navbar-collapse" class="navbar-collapse
collapse">
                    <ul class="nav navbar-nav">
                        <li class="active nav-item"><a
href="index.html">Home</a></li>
                        <li class="nav-item"><a
href="features.html">Features</a></li>
                        <li class="nav-item"><a
href="pricing.html">Pricing</a></li>
                        <li class="nav-item dropdown">
                            <a class="dropdown-toggle" data-
toggle="dropdown" data-hover="dropdown" data-delay="0" data-close-
others="false" href="#">Pages <i class="fa fa-angle-down"></i></a>
                            <ul class="dropdown-menu">
                                <li><a
href="download.html">Download Apps</a></li>
                                <li><a
href="blog.html">Blog</a></li>
                                <li><a href="blog-single.html">Blog
Single</a></li>
                                <li><a href="blog-
category.html">Blog Category</a></li>
                                <li><a href="blog-
archive.html">Blog Archive</a></li>
                                <li><a href="about.html">About
Us</a></li>
                                <li><a
href="contact.html">Contact</a></li>
                            </ul>
                        </li><!--//dropdown--
>
                        <li class="nav-item"><a
href="login.html">Log in</a></li>
                        <li class="nav-item nav-item-cta last"><a
class="btn btn-cta btn-cta-secondary" href="signup.html">Sign Up
```

```
Free</a></li>
                    </ul><!--//nav-->
                </div><!--//navabr-collapse-->
            </nav><!--//main-nav-->
        </div><!--//container-->
    </header><!--//header-->
```

2. Write JavaScript code to traverse the site structure information and generate the navigation code, then include the JavaScript code on the layouts, for example:

   ```
   <script type="text/javascript" src="_scs_theme_root_/assets/js/
   navbar.js"></script>
   ```

3. Modify the exact output of the `topnav.js` file to match the markup expected in your particular CSS.

Update Site Layouts

At this point the site is functional but has no editable areas (slots) in the layouts.

1. Locate or add a DIV element in a layout and designate it as a slot.

   Slots are DIV elements in the layout that have the value `"scs-slot"` in the class attribute. Each slot must have a unique `id` attribute. For example:

   ```
   <div id="slot-content1" class="scs-slot scs-responsive"></div>
   ```

   To make the slot adjust automatically to the size of the browser viewport, include the class attribute `"scs-responsive"`.

2. Repeat this step as needed to create additional slots in the layout or in other layouts.

Publish the Site

After you've completed and saved all your changes to a site, you must **Publish** the site.

Publishing the site takes all of the changes in the current update and merges that into the base, making what was in the update into the new base site.

Create the New Template

You now have a functional site with an associated theme, so you can make this into a site template to share with others so they can create sites from it.

1. In the Oracle Content Management side navigation, click **Developer**.
2. Click **View all Templates**.
3. Click **Create** and choose **From existing site**.
4. Select the new site you've created using a starter site template and the Bootstrap site template.
5. Enter a name for the new site template and click **Create**.
6. To package the site template for use with other Oracle Content Management instances, select the site template and choose the **Export** menu option to create a `.zip` file that can be downloaded.

Develop Site Templates with Developer Cloud Service

You can use Developer Cloud Service to develop site templates for Oracle Content Management.

Take the following steps to develop a site template in Developer Cloud Service, test it locally, and then export it into Oracle Content Management:

1. Set Up Oracle Content Management Toolkit on Your Local Machine.
2. Sign in to the Developer Cloud Service Console for Oracle Content Management.
3. Create a Project in Developer Cloud Service.
4. Add Oracle Content Management Toolkit to the Project Code in the New Git Repository.
5. Create a Template in Developer Cloud Service.

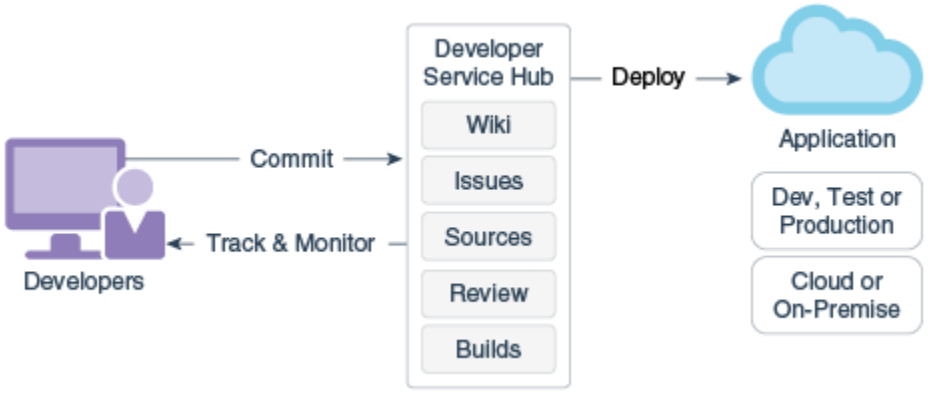
   You can create a new site template to develop, copy an existing site template in Developer Cloud Service, or import a site template from Oracle Content Management.
6. Test the Template in a Local Test Harness.
7. Merge Changes.
8. Export a site template from Developer Cloud Service into Oracle Content Management.

Sign in to the Developer Cloud Service Console for Oracle Content Management

Start developing your custom components for Oracle Content Management on the Developer Cloud Service console.

As an administrator for Oracle Cloud services, you can use My Service Administration to create and manage your Cloud services. If you're a service instance administrator for Oracle Content Management and a service administrator for Standard Developer Service, you can set them up and start using them:

1. Sign in to Oracle Cloud, using the information that was provided for your account.
2. Sign in to My Service Administration to create and manage your Oracle Content Management instance and your Standard Developer Service.



3. Verify your Oracle Developer Cloud Service email, as requested.
4. Set up you Oracle Content Management instance, using the subscription details for your service, and go to the Oracle Content Management URL for your instance.
5. Go to your URL for the Standard Developer Service.
6. Sign in to your Oracle Developer Cloud Service account.

Access the Developer Cloud Service URL and sign in to the console.

Create a Project in Developer Cloud Service

You can create a project in Developer Cloud Service using the "Content Experience Cloud" project template, or you can create a project with an empty Git repository and import the Content Toolkit from your Oracle Content Management instance.

- Create a Developer Cloud Service Project with an Oracle Content Management Template
- Create a Project in Developer Cloud Service with a Content Toolkit Download from Oracle Content Management
- Add Oracle Content Management Toolkit to the Project Code in the New Git Repository

Create Site Templates in Developer Cloud Service

You can use the cec command-line utility to create Oracle Content Management site templates from the available source site templates.

Use the `cec create-template` command to create a site template from one of the available source site templates. Typing `cec create-template -h` on the command line gives the available source site templates.

Here is an example of creating a site template:

```
cec create-template CafeSupremoLite_yourname -f CafeSupremoLite
```

Windows: This command creates a symlink for themes to render in an external HTML WYSIWYG editor (such as `_scs_theme_root_`) while you are creating site templates. To create symlinks in Windows, you normally need to run the command-line utility with administrative privileges. If you are not using a WYSIWYG editor to edit the theme, you don't need to run with administrative privileges, and you can ignore the symlink creation error.

The preceding example creates the `CafeSupremoLite_yournameTheme` site template and makes the source code available at `cec-components/src/main/`. The following table shows the locations of the source code after you create a site template in Developer Cloud Service.

Template Source Code	Theme	Components for Template
`cec-components/src/main/templates`	`cec-components/src/main/themes`	`cec-components/src/main/components`

The local test harness shows the components too, with the ability to filter them by site template and type.

You can edit theme and component files with any text or code editor. See Test with a Local Test Harness. Refresh the browser after editing the theme or component to see your changes.

> **Important:**
>
> The source code for your site templates, themes, and components exists in `src/main/`. You shouldn't modify any files outside `src/main` because they are needed for the functioning of the Oracle Content Management local server.

Copy a Site Template in Developer Cloud Service

You can copy an existing Oracle Content Management site template in Developer Cloud Service.

To copy one of your existing site templates from `src/main/templates`, use the `cec copy-template` command. If the site template contains assets from other repositories, optionally provide the repository mapping otherwise those assets will not copy.

The following example copies the `Temp1` site template to a new site template named `Temp2`:

```
cec copy-template Temp1 -n Temp2
```

Import a Site Template into Developer Cloud Service

You can import site templates from Oracle Content Management into Developer Cloud Service for further development.

If you have a site template zip file created from a Oracle Content Management server, you can import that file into Developer Cloud Service for further development, such as to edit the theme or components. Use the following command:

```
cec import-template <location of the site template zip file>
```

Specify the folder that contains the zip file in Oracle Content Management.

See About Site Templates and Export a Site Template.

Merge Changes

After you create a component, site template, or content layout or edit source code on your local machine, you need to merge new and changed components and site templates into your project's Git repository.

To merge changes into your Git repository, enter the following commands, in order, in a terminal window.

```
cd cec-components git pull
git add .
git status
git commit -a -m "Your comments" git pull
git push
```

Export a Site Template from Developer Cloud Service

You can export a site template zip file from Developer Cloud Service and use the file to create a site in Oracle Content Management.

Once the site template development is complete, you can run the following command to export the site template. The command response tells you where the zip file for the site template is created in Oracle Content Management.

```
cec export-template CafeSupremoLite_yourname
```

See About Templates and Import a Site Template.

23

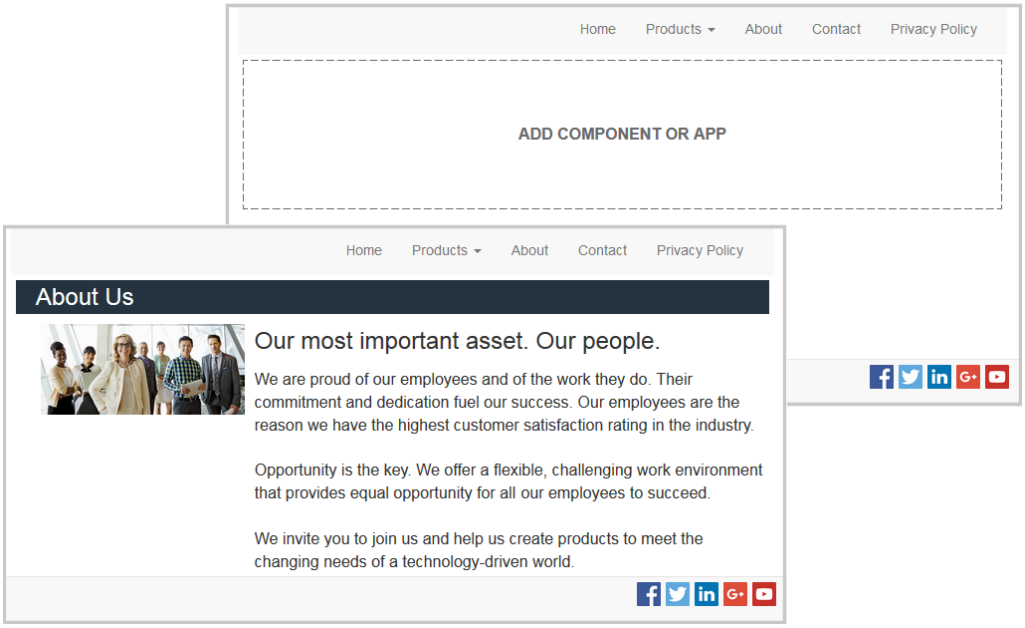
Develop Themes

A theme defines the general look-and-feel — the overall style — of a site, including color scheme, font size, font type, and page backgrounds. A theme provides visual consistency between the pages in a site. You can create unique themes and variations of themes, specifying the design and sample content, which then can be used to create sites to promote your brand and your vision.

About Themes

Themes define the general look-and-feel of a site, including content, appearance, and behavior. A theme provides visual consistency between the pages in a site.

Designing a new theme means specifying the layout, style, sample content, navigation, and all the basic information that serves as a starting point for a new site. Theme designers set the expectations for how a site will look and behave. A theme should be designed keeping in mind the way it will be used; for example, most or all users will be expected to access the site with a mobile device. Designing custom themes is useful if you have users who want to create many similar sites. You can design a theme using page layouts for common patterns that can be shared across themes.



A theme contains page layouts that are used to design content, appearance, and behavior for sites. You change the design and settings, and add content, to create a site that sells your style, your brand, and your vision.

A theme includes:

- Assets for background images or other content that are part of page layouts (images, JavaScript files, and so on)
- Style settings for a site (CSS)
- Several page layouts (HTML files)
- Code to construct navigation for the site (JavaScript files)
- A list of basic styles that can be used with the components (specified in `design.css` and `design.json` files)

A theme also can include *seed data*, which is used to populate a new page created from one of the page templates. For example, a user creates a new page for a Products section and chooses the page layout called `new_product.html`. If the theme contains a file called `new_product-pageseed.json`, the new page will be populated with the contents of the page seed file when it's first created. As with sample content, this seed data can be modified and is only there to provide a starting point for you to build out the page.

You can create a theme that uses a subset of components that are intended to work with that theme. When a user chooses that theme for their site, they'll see only the components that are specified for that theme. See Associate Components with Themes.

Each website uses a theme. When you create the site from a template, you inherit the theme from the template. You can change the theme for a site at any time. Oracle Content Management provides a number of templates with themes that you can use to get started.

If a site uses a new, unpublished theme, the theme is automatically published with the site when you place the site online for the first time. If you make changes to a theme

and want to update online sites to show the changes, you must explicitly publish the theme. Only the theme owner or a user with manager privileges can explicitly publish a theme.

> **Note:**
>
> If you publish changes to a theme, all online sites that use the theme will reflect the change. For example, if you change the default font specified in the theme and publish the theme, all sites that use the theme will use the new default font.

See also Manage Themes.

Basic Theme Structure

The basic structure of a theme includes the design, navigation, and styles specified in folders stored in Oracle Content Management. A theme is part of the template for a site.

When a user selects a template to create a new site, the associated theme data is automatically loaded. If you use the JET Starter Template or Starter Template, the theme is automatically copied instead of referenced. As a developer, if you're using the starter template, you want your own copy of the theme.

A theme is organized in a specific folder and file structure, as illustrated in this example showing basic folders and files:

```
theme
     assets
         css
             main.css
         js
             topnav.js
     designs
         default
            design.css
            design.json
     layouts
     publish
     responsepages
         404.html
     viewport.json
     _folder.json
     _folder_icon.png
     components.json
```

Certain folders contain specific types of information, including the following folders:

- `assets`: JavaScript, Cascading style sheet (CSS), images, and other support files that are referenced by the layouts.
- `designs`: `design.css` and `design.json` files, which are used to specify style options for components.
- `layouts`: HTML files for page templates, which are used to display pages of the site.

> **Note:**
>
> All HTML files must have a DOCTYPE element at the start of the file that looks like this: `<!DOCTYPE html>`

- `publish`: After a theme has been used in a published site, this directory is listed and contains copies of files. This directory is visible if you have sync'd the theme using the desktop app, but it's not included if you've exported the theme as part of a template.
- `responsepages`: Special page for handling errors (404).

> **Note:**
>
> If a page in a site is flagged as an Error Page, then the 404 error message from the theme will be ignored and the designated Error Page is used instead.

- `viewport.json`: Specifies Viewport settings for the theme.
- `_folder.json`: Specifies the name and GUID for the theme. For example:

```
{
      "themeName":"MarketingCampaignTheme",
      "itemGUID":"TB79D65F699B022AC4E11F4D4EE870070A1ADD86BBBB"
}
```

  The GUID is created by Oracle Content Management when the theme is first imported or when it is copied. The theme name is assigned by the theme developer when creating a theme.

- `components.json`: Records the custom components used within the theme.

There are two key files that you'll work with when creating a new theme. These files set styles for components:

- `design.css`
- `design.json`

It is good practice to put navigation information into one JavaScript file; for example, a file named `nav.js`. The theme's `/assets/js/` folder is a good location for such a file.

Site Navigation

The hierarchy of a site is stored in the `structure.json` file associated with the site. The hierarchy is loaded into memory and made available in the page context as the `SCS.structureMap` object.

Site Builder reads the `structure.json` file to draw the site tree in Site Builder. The `structure.json` file will contain code for the site pages. For example:

```
"pages": [  {
     "id": 100,
     "name": "Home",
     "parentId": null,
     "pageUrl": "index.html",
     "hideInNavigation": false,
     "linkUrl": "",
     "linkTarget": "",
     "children": [ 200,
                   300,
                   400,
                   500 ],
     "overrideUrl":false
     }

     {
     "id":200,
     "name":"Products"
     "parentId":100,
     "hideInNavigation":false,
     "LinkUrl":"",
         "linkTarget":"",
         "children":[ 204, 205],
         "overrideUrl":false
     }

     {
     "id":204,
     "name":"Hiking Boots",
     "parentId":200,
         "pageUrl":"products/hiking_boots.html",
         "hideInNavigation":false,
         "linkUrl":"",
     "linkTarger":"",
     "children":[],
     "overrideUrl":false
     }
```

Navigation JavaScript code is necessary within the site pages to also read that structure and draw out the navigation links for the site. Templates provided with Oracle Content Management include sample navigation JavaScript files that illustrate how this works.

The `topnav.js` file used in some of the themes provided with Oracle Content Management is an example of how you can use the `SCS.structureMap` object along with the Render API calls such as `SCSRenderAPI.getPageLinkData` to traverse the site structure and draw out the HTML markup needed to render the navigation menus in the page. Here is code from the sample `topnav.js` file:

```
function renderNode(id, navBar)
{
    if (id >= 0)
```

```
    {
        var navNode = SCS.structureMap[id];
        if( navNode &&
            (
                ( typeof navNode.hideInNavigation != "boolean" ) ||
                ( navNode.hideInNavigation === false )
            ) )
        {
            var navItem = document.createElement("li");
            var navLink = document.createElement("a");
            var navText = document.createTextNode(navNode.name);

            var linkData = SCSRenderAPI.getPageLinkData(navNode.id) ||
{};
            if( linkData.href ) {
                navLink.href = linkData.href;
            }
            if( linkData.target ) {
                navLink.target = linkData.target;
            }

            navLink.appendChild(navText);
            navItem.appendChild(navLink);

            if (navNode.children.length > 0)
            {
                var navSub = document.createElement("ul");

                for (var c = 0; c < navNode.children.length; c++)
                {
                    renderNode(navNode.children[c], navSub);
                }

                navItem.appendChild(navSub);
            }
            navBar.appendChild(navItem);
        }
    }
}

function renderNav()
{
    var topnav = document.getElementById("topnav");        // expected
to be an empty <div>

    if (topnav)
    {
        var navBar = document.createElement("ul");

        renderNode(SCS.navigationRoot, navBar);

        topnav.appendChild(navBar);
    }
}
```

```
// Must wait for all our script to be ready...
if (document.addEventListener)
{
    document.addEventListener('scsrenderstart', renderNav, false);
}
else if (document.attachEvent)
{
    document.documentElement.scsrenderstart = 0;
    document.documentElement.attachEvent("onpropertychange",
        function(event)
        {
            if (event && (event.propertyName == "scsrenderstart"))
            {
                renderNav();
            }
        }
    );
}
```

You can use Render API calls to generate navigation links that will function in your site Edit and Preview modes and in a published online site. See Render API Reference.

It is good practice to put navigation information into one JavaScript file, such as `topnav.js`. The JavaScript file is usually stored in the `/assets/js/` folder of the theme, as you can see in the sample themes provided with Oracle Content Management.

Create a Theme

You can create a new theme by copying an existing theme and making changes to the copy. You also can import and use Bootstrap content in a new theme.

> **Note:**
>
> Whether you are creating a new theme or making updates to an existing theme, always make a copy of the theme and work on the copy. Test it with a sample site or a copy of your real site to ensure it works correctly. Note that changes made to an existing theme will be implemented on any sites that use the theme immediately after the revised theme is published.

Copy a Theme

1. On the home page, click **Developer**.

   The **Developer** page is displayed.

2. Click **View All Themes**.

   A list of existing themes is displayed. You can control how themes are displayed by clicking the view icon and selecting an option from the list.

3. Select a theme and choose **Copy** in the right-click menu or click  in the actions bar.

   All the folders and files of the theme are copied, including any sample pages and content.

> **Note:**
>
> When you copy a theme, sharing information for the theme isn't copied.

4. Enter a name for the copied theme. You can't use a name used by another theme.

   You can use letters, numbers, underscores (_), and hyphens (-) in the name. If you enter a space, it's automatically replaced with a hyphen.

   Don't use the following names for site templates, themes, components, sites, or site pages: authsite, content, pages, scstemplate_*, _comps, _components, _compsdelivery, _idcservice , _sitescloud, _sitesclouddelivery, _themes, _themesdelivery. Although you can use the following names for site pages, don't use them for site templates, themes, components, or sites: documents, sites.

5. Optionally, enter a description for the theme.

6. Click **Copy**.

   A progress bar shows the new theme name and copy status. When the theme is copied, the name appears in the list of themes. You can explore the folders and files that make up the theme by clicking the theme name in the list of themes.

7. Use the Oracle Content Management desktop app to sync the theme folders and files to your local system. This enables you to browse the local folders and work directly with files. Changes you make to the theme are automatically synced. You can make changes using your favorite HTML, code, or text editing tools.

Set Whether Custom Styles Can Be Used in a Site

A theme administrator can specify styles that come with a theme, or customize the styles.

A setting in a theme's `components.json` file specifies whether custom styling can be done in Site Builder. This is to control that a site contributor keeps within the style of the site when building it and does not use, for example, other fonts and colors.

To not allow the use of custom styles in a site:

1. Hide the **Customize** option in the **Settings** panel for components.
   In the `components.json` file for the theme, add the following object along with other component definitions:

```
[
    {
        "showStyleClassOnly": "true"
    }
]
```

2. Customize toolbar groups and buttons of the rich text editor to remove custom styling:
   In the `components.json` file for the theme, you can specify the `toolbarGroups` and `removeButtons` properties of the rich text editor to customize the groups and buttons within groups for `Title` and `Paragraph` components.

For example, add the following objects along with other component definitions to remove styles, font, and colors from the `Styles` group:

```
[
    {
        "name": "",
        "list": [
            {
                "type": "scs-title",
                "id": "scs-title",
                "initialData": {
                    "toolbarGroups": [
                        {
                            "name": "basicstyles",
                            "groups": ["basicstyles"]
                        },
                        {
                            "name": "styles",
                            "groups": ["styles"]
                        },
                        {
                            "name": "colors",
                            "groups": ["colors"]
                        },
                        "/",
                        {
                            "name": "undo",
                            "groups": ["undo"]
                        },
                        {
                            "name": "links",
                            "groups": ["links"]
                        },
                        {
                            "name": "paragraph",
                            "groups": ["list", "indent"]
                        },
                        {
                            "name": "align",
                            "groups": ["align"]
                        },
                        {
                            "name": "cleanup",
                            "groups": ["cleanup"]
                        }
                    ],
                    "removeButtons":
"Styles,Subscript,Superscript,Strike,Anchor,Blockquote,Link,Unlink,Font,Te
xtColor,BGColor"
                }
            },
            {
                "type": "scs-paragraph",
                "id": "scs-paragraph",
                "config": {
```

```
                    "toolbarGroups": [
                        {
                            "name": "basicstyles",
                            "groups": ["basicstyles"]
                        },
                        {
                            "name": "styles",
                            "groups": ["styles"]
                        },
                        {
                            "name": "colors",
                            "groups": ["colors"]
                        },
                        "/",
                        {
                            "name": "undo",
                            "groups": ["undo"]
                        },
                        {
                            "name": "links",
                            "groups": ["links"]
                        },
                        {
                            "name": "paragraph",
                            "groups": ["list", "indent"]
                        },
                        {
                            "name": "align",
                            "groups": ["align"]
                        },
                        {
                            "name": "insert",
                            "groups": ["image", "table"]
                        },
                        {
                            "name": "cleanup",
                            "groups": ["cleanup"]
                        }
                    ],
                    "removeButtons":
"Styles,Subscript,Superscript,Strike,Anchor,Blockquote,Link,Unlink,F
ont,TextColor,BGColor"
                }
            }
        ]
    }
]
```

Not allowing custom styles has the following effects:

- In all built-in components, the **style** tab hides the (o) Customize option.
- In the rich text editor, the toolbar buttons for setting styles are hidden, and specification of font family, font color, and so on are overridden.

Hide Components and Section Layouts for a Theme

You can hide components and section layouts to prevent a site developer from using them by editing a theme's component.json file. You may want to do this in order to promote a consistent look and feel. See Hide Components and Section Layouts for a Theme.

Verify the Theme with a Site

After you've completed editing the theme, you need to check that all the pieces work together with the site as planned.

1. Make sure you've synchronized your copied folders and files with the Oracle Content Management desktop app and that all the modifications have been saved.
2. Open a site (it can be a test or existing offline site) in Site Builder and switch the site to use the new theme.
3. Test the site with the theme by adding pages and using settings, viewing images, checking the navigation, and anything else you've changed in the new theme.
4. Check that everything renders correctly, including all files and links.
5. Publish the theme so the site will implement the theme changes, then check it in a runtime environment.

See Publish Themes.

Use a Bootstrap Theme

Similarities between Oracle Content Management themes and Bootstrap themes makes it possible to convert existing Bootstrap theme pages and content for use in an Oracle Content Management theme.

Bootstrap is a free and open-source collection of tools for creating websites and web applications. It contains HTML- and CSS-based design templates for interface components and JavaScript extensions.

You follow the same instructions for creating a theme, but you work with code (using whichever editor you choose) to import and edit any pages or content you want from the Bootstrap theme.

It's unlikely that you would use all of a Bootstrap theme with all of its pages and content. Typically you will select parts of the theme to use in an Oracle Content Management theme, maybe only a few of the page templates and only some parts of those. For example, a Bootstrap theme contains several blocks within the templates, including a header, a navigation block, a body block and a footer block. Because of the way that Bootstrap themes are shipped as a collection of files that can be viewed directly from the file system (without a web server), they contain a lot of duplication within each page (they have to because there is no page assembly engine involved). When you use this information in an Oracle Content Management theme, you need to take these parts and add them to reusable Oracle Content Management page templates, and then use Oracle Content Management to dynamically assemble multiple pages from those templates.

One example of adding information to reusable templates is the navigation section. Navigation in a typical Bootstrap theme is duplicated on all pages, but when you move that into an Oracle Content Management theme, you must use JavaScript code to dynamically walk the hierarchy of the site and generate the navigation structures. Then you just include that script on all the pages and they all get the navigation, and that navigation adapts as pages are added or removed from the site.

See Create a Site Template from Bootstrap or a Website Design Template.

Hide Components and Section Layouts for a Theme

By default, all components and section layouts are available to the people developing a site. There may be times when you want to hide a component or section layout so it is unavailable when using a particular theme. For example, if you want to promote a consistent look and feel that doesn't use any button components or slider section layouts, you can hide them in a theme by editing the theme's components.json file.

To hide components and section layouts in a theme:

1. In Oracle Content Management, click **Developer** in the left navigation menu.
2. On the Developer page, click **View all Themes**.
3. On the Themes page, select the theme to modify and click **Open** in the right-click menu or click  in the actions bar.
4. Select the components.json file and click **Download**.
5. Open the components.json file in a text editor.
6. Add objects to specify the component type and ID, and set the property `hidden` equal to `true`.
   For example, the following is an example entry that hides the button component:

   ```
   "type":"scs-button",
   "id":"scs-button",
   "hidden":true
   ```

   Similarly, the following is an example entry that hides the slider section layout:

   ```
   "type":"scs-sectionlayout",
   "id":"scs-sl-slider",
   "hidden":true
   ```

   The full entry into the components.json file to hide both the button component and slider section layout would be as follows:

   ```
   [
       {
           "name":"",
           "list":[
               {
                   "type":"scs-button",
                   "id":"scs-button",
                   "hidden":true
               },
               {
                   "type":"scs-sectionlayout",
                   "id":"scs-sl-slider",
                   "hidden":true
               }
           ]
   ```

```
    },
    {
        "name": "Starter",
        "list": [
            {
                "type": "component",
                "id": "StarterComponent",
                "themed": true
            },
            {
                "type": "component",
                "id": "StarterFooter",
                "themed": true
            }
        ]
    }
]
```

7. Once you've made the desired changes, upload the components.json file to the theme as a new version.

Hide Component Alignment, Width or Spacing Options for a Theme

By default, most components allow someone building sites to specify the alignment, width, and spacing options in the settings dialog of a component. As a developer, however, you may want to hide these options on a custom component to prevent contributors from rendering content in a way inconsistent with the site theme. For example, if you want to promote a consistent look and feel that centers a button and keeps the spacing set to 30 pixels, you can hide the options to change them in the settings by editing the theme's components.json file.

There may also be times when you want to override a custom component's initial default values, which can also be done by editing the components.json file. Any changes to the initial values will only apply to components added to the page after the edited components.json file has been uploaded as a new version.

To hide the settings options for alignment, width, or spacing for a component in a theme:

1. In Oracle Content Management, click **Developer** in the left navigation menu.
2. On the Developer page, click **View all Themes**.
3. On the Themes page, select the theme to modify and click **Open** in the right-click menu or click  in the actions bar.
4. Select the components.json file and click **Download**.
5. Open the components.json file in a text editor.
6. Add objects to specify the component type and ID, and set the properties you want to hide equal to `true`.
   For example, the following is an entry that hides the alignment, width, and spacing properties of a button component in the settings dialog:

```
"type":"scs-button",
"id":"scs-button",
```

```
"hideAlignmentAndWidth": true,
"hideSpacing": true,
```

> **Note:**
>
> The id field value is case sensitive.

If you want to override the default alignment and spacing properties of the button component, the following is an example of how to specify initial values:

```
"initialData": {
    "alignment": "center",
    "marginTop": 30,
    "marginBottom": 30
```

The full entry into the components.json file to specify default values and hide alignment, width, and spacing options in the settings dialog of a button component used in a theme would be as follows:

```
[
    {
        "name": "",
        "list": [
            {
                "type": "scs-button",
                "id": "scs-button",
                "hideAlignmentAndWidth": true,
                "hideSpacing": true,
                "initialData": {
                    "alignment": "center",
                    "marginTop": 30,
                    "marginBottom": 30
                }
            }
        ]
    },
    {
        "name": "Starter",
        "list": [
            {
                "type": "scs-component",
                "id": "StarterComponent",
                "themed": true
            },
            {
                "type": "scs-component",
                "id": "StarterFooter",
                "themed": true
            }
        ]
```

```
        }
]
```

7. Once you've made the desired changes, upload the components.json file to the theme as a new version.

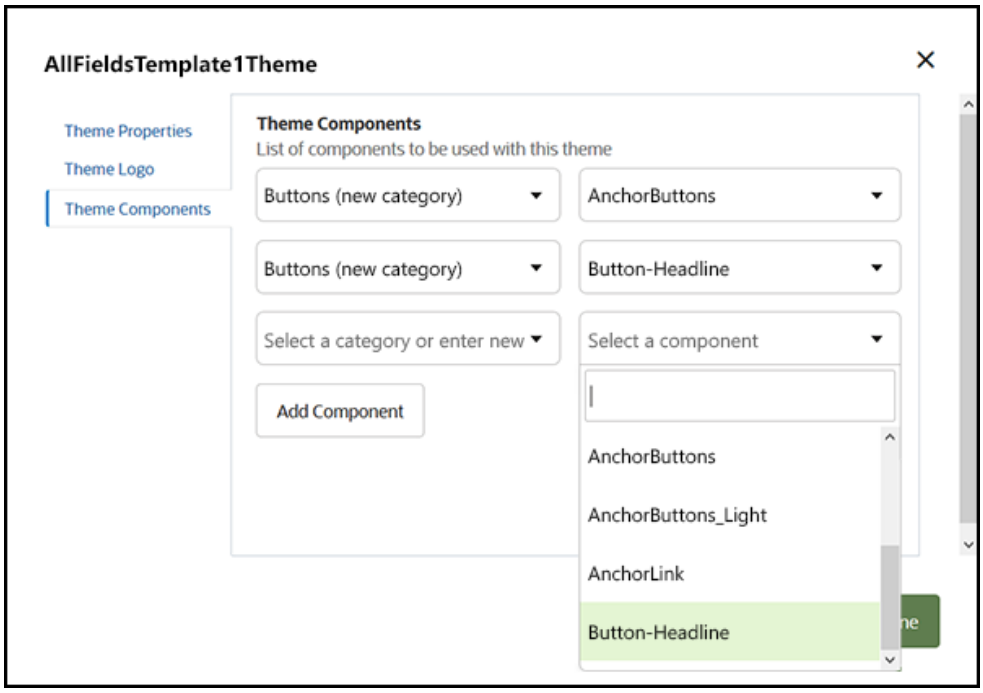
Associate Components with Themes

You can associate components, section layouts, and component groups with a theme to use a specific subset of components with the theme.

As a developer, you can create a theme and components for a template that marketers will use to create sites that promote your organization's products. By associating a component with a theme, you make it available when the user selects **Theme Components** in Site Builder. On the **Theme Components** tab, the user will see only the components you associated with the theme. Associating a component with a theme ensures that this component will be exported with the site template, even if it is not used on the site.

To associate a component with a theme:

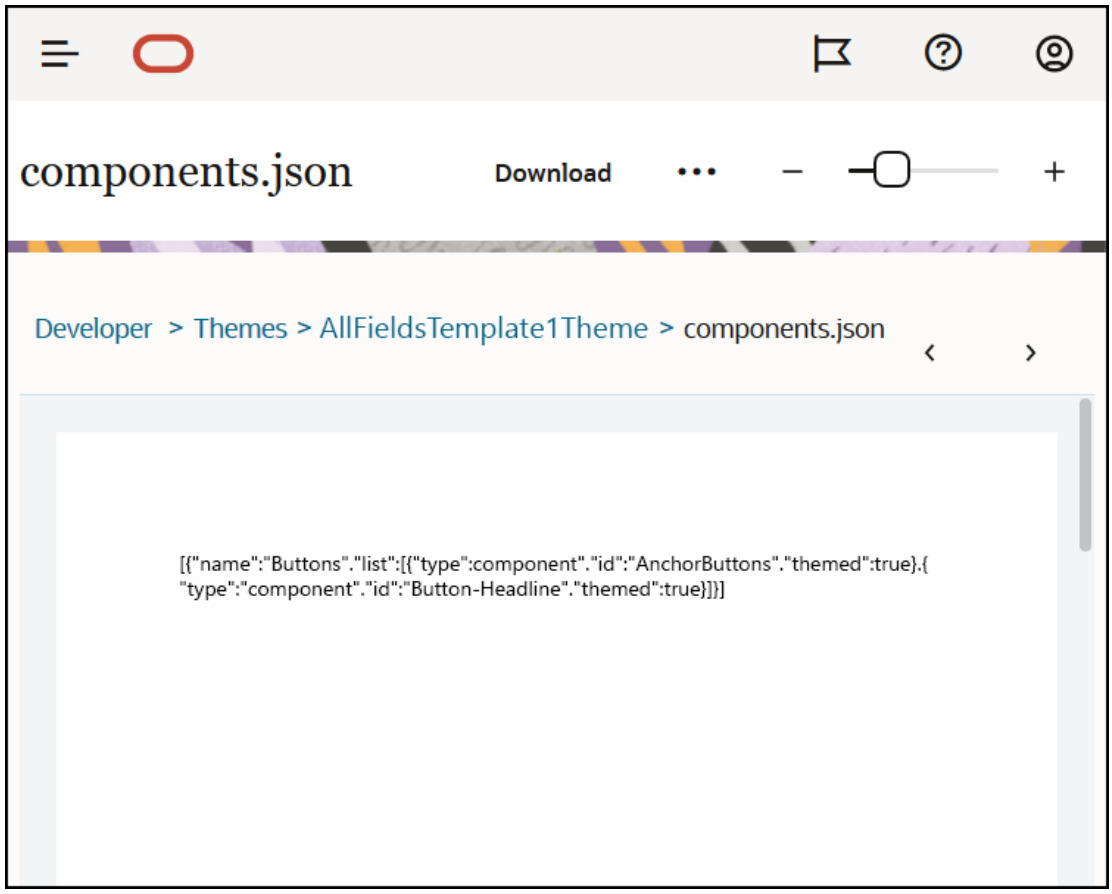
1. In Oracle Content Management, click **Developer** in the left navigation menu.
2. On the **Developer** page, click **View all Themes**.
3. On the Themes page, select a theme and click **Properties** in the right-click menu or click in the actions bar.
4. Click **Theme Components** to open the **Theme Components** tab.
5. Select one or more components to associate with the theme and categorize the components:
    a. From the **Select a component** drop-down list, choose a custom component.
    b. For the first component you choose, enter the name of a category to create a category for the component.
    c. For each additional component you choose, select an existing category or create a new category.
    d. Click **Add Component**.



6. Click **Save** to associate the components you selected with the theme.
7. On the **Themes** page, select a theme and click Properties to open the **Theme Components** tab again.
8. Verify that the components you selected were saved.

   To remove an associated component from the theme, click **X** next to the component name on the **Theme Components** tab, and then click **Yes** in the **Remove Component** dialog.

You can see a list of components associated with the theme that a template is using on the **Details** page for the template. The references to the components associated with the theme are stored in the `components.json` file for the theme.



After configuring the theme association, you can deploy a template to Oracle Content Management and share it with the marketing team. When marketers choose that theme for their site, they'll see only the components you specified for the theme on the **Theme Components** tab in Site Builder. The lists of **Custom** and **All** components also include the associated themed components.

Associate a Component with a Theme in Oracle Content Management Toolkit.

To associate a component with a theme in Content Toolkit, you can use the following cec command. The component will appear on the **Theme Components** tab in Site Builder, as well as in lists of **Custom** and **All** components.

```
cec add-component-to-theme <component>
```

The following **cec** command removes the association of a component from a theme.

```
cec remove-component-from-theme <component>
```

For information about Content Toolkit, see Develop with Oracle Content Management Toolkit.

Sites Rendering API

The Sites Rendering API for Oracle Content Management (SCSRenderAPI) is a window-global object present on all Oracle Content Management web pages. It is primarily

responsible for rendering the slots and components of the page, and it provides an interface for JavaScript code present on theme layouts. If you're working on themes or components as a developer, you may find these events and functions.The **runtime** SCSRenderAPI renders the view and preview display modes, while the **design time** SCSRenderAPI renders the navigation, edit, and annotation display modes.

In runtime, the Sites Rendering API has three general purposes:

- Populate the slots on the page with components and content.
- Satisfy informational requests made by JavaScript code found on the page layout.
- Raise events during the lifecycle of the page-rendering process.

In design time, the Sites Rendering API has four general purposes:

- Populate the slots on the page with components and content.
- Satisfy informational requests made by JavaScript code found on the page layout.
- Raise events during the lifecycle of the page-rendering and editing process.
- Interact with Site Builder to allow page editing and annotation.

The Sites Rendering API is loaded in Oracle Content Management web pages by placing the following script tag on theme layouts:

```
<script data-main="/_sitescloud/renderer/renderer.js" src="/
_sitescloud/renderer/require.js"></script>
```

This is normally placed at the bottom of layouts. (Note that the URLs in this tag will automatically be adjusted for the appropriate environment: design time or runtime.) The SCSRenderAPI object loads asynchronously; custom JavaScript code can listen for the availability of the Sites Rendering API by handling the scsrenderstart event.

24

Develop Layouts

A layout defines how content is arranged on a page and is used to produce the HTML for pages used in Oracle Content Management sites.

About Layouts

A layout defines how content is arranged on a site page. Different layouts can contain a different number of named slots. A slot is a region that spans the width of the page and can contain one or more types of content.

Every theme has several page layouts. See About Themes.

When you add a page to a site, you select a layout to use for that page. Each layout has areas on the page—known as slots—where you can drag and drop content. What content goes into these slots is completely up to you. It can be anything from titles, text, and dividers to multimedia, galleries, and social media.

A layout contains valid HTML constructs as well as special markup understood by the Oracle Content Management renderer. A layout must begin with a DOCTYPE statement in order to configure the browser to render the page in a standards-compliant mode; for example: `<! DOCTYPE html>` This statement is required by certain components to achieve best results.

This sample code shows a minimal layout:

```
1  <!DOCTYPE html>
3  <head>
4          <meta http-equiv="X-UA-Compatible" content="IE=edge">
5
6          <script src="/_themes/[!--$SCS_THEME_NAME--]/assets/js/
topnav.js"></script>
7          <link rel="stylesheet" type="text/css"
8                 href="/_themes/[!--$SCS_THEME_NAME--]/assets/css/main.css">
9          <link rel="stylesheet" type="text/css"
10                href="/_themes/[!--$SCS_THEME_NAME--]/designs/
```

```
[!--$SCS_DESIGN_NAME--]/design.css">
11
12          <!--$SCS_RENDER_INFO-->
13          <!--$SCS_SITE_HEADER-->
13          <!--$SCS_PAGE_HEADER-->
14   </head>
15   <body>
16          <div id="topNavigation"></div>
17          <div id="mainContentSlot" class="scs-slot scs-responsive"></
div>
18
19          <script data-main="/_sitescloud/renderer/renderer.js"
20                  src="/_sitescloud/renderer/require.js"></script>
21          <!--$SCS_SITE_FOOTER-->
22   </body>
23   </html>
```

Various tokens are expanded when a page is rendered in the browser.

- `[!--$SCS_THEME_NAME--]`

  This expands to the name of the theme currently chosen for the site. Using this token allows the theme to be copied, because URLs that use this token will reference the current theme.

- `[!--$SCS_DESIGN_NAME--]`

  This expands to the name of the design currently chosen for the site. This allows the layout to be used by multiple designs within the theme.

- `[!--$SCS_RENDER_INFO--]`

  This expands to a script tag that holds the page hierarchy and component rendering information for the page. This should be placed in the `<head>` section of the layout.

- `[!--$SCS_SITE_HEADER--]`

  This expands to the site header value that is specified in the Header field in the Search Engine Optimization (SEO) properties. Site-wide markup that you want placed on all pages can be entered here. See Set Search Engine Properties.

- `[!--$SCS_PAGE_HEADER--]`

  This expands to the page header value that is found in the Page Header field in the Page Settings properties in Site Builder. Page-specific markup that you want placed for this page can be entered here.

- `[!--$SCS_SITE_FOOTER--]`

  This expands to the site footer value that is found in the Footer field in the Search Engine Optimization (SEO) properties. Site-wide markup that you want placed on all pages can be entered here. See Set Search Engine Properties.

> **Note:**
>
> The tokens can also use the `<!--$` prefix and the `-->` suffix as delimiters in place of `[!--$` and `--]`.

When a site is online, the following tokens in the link are replaced with real values that are aware of the context in which they are being used. This enables the link to function when a site is being edited and in the published site when it is online.

- `/_sitescloud/` is replaced with `/_sitesclouddelivery/`
- `/_themes/` is replaced with `/_themesdelivery/`

Slots are DIV elements in the layout that have the value `"scs-slot"` in the class attribute. A slot is where users can add components to fill in the site content. Multiple DIV elements can be designated as slots by assigning the `"scs-slot"` class attribute. Each slot must have a unique `id` attribute.

> **Note:**
>
> Slots can't be nested, but you can give the suggestion they are by using CSS to overlay them. If you want to do this, use a component group or section layout. See Create a Section Layout.

Slots that additionally have a class attribute value of `"scs-responsive"` will refresh their content as the browser viewport changes resolution. This allows slots to render responsively for a large desktop display or a small mobile device.

The final `<script>` tag (line 19 in the example) loads the Oracle Content Management rendering code. This code is responsible for drawing the components on the page, and it also allows custom code to access the Render API. Without this `<script>` tag, pages based on the layout can't be changed by Site Builder.

Search Engine Optimization (SEO)

You can provide keywords to help search engines identify the contents of your site.

SEO Settings

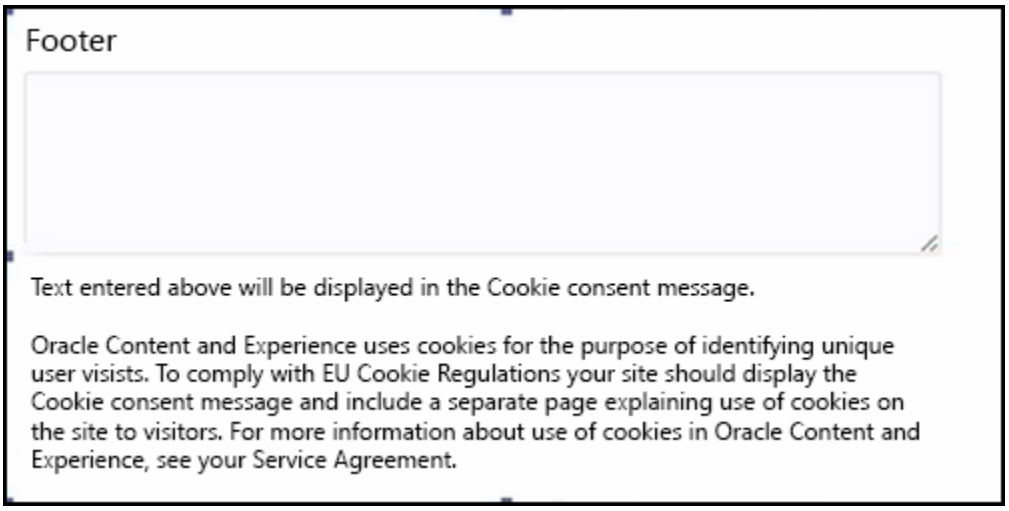
Search engine optimization (SEO) settings are defined at the site level and at the page level. The SEO text will be rolled into all out-of-the box templates, in the footer.

See Set Search Engine Properties.

Cookies for Site Visitors

Site visitors use cookies for SEO, one cookie for each browser and each site, for billing purposes. Each cookie needs to be renewed hourly or after 24 hours.

Each site must include a popup that notifies visitors about cookies.



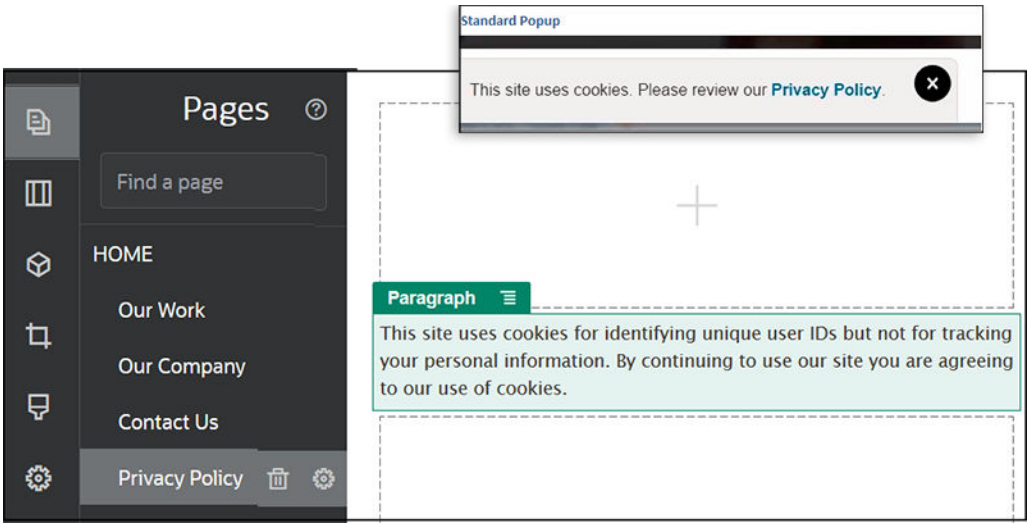
The popup should include the following information:

- SEO header and footer text
- Div in footer with a specific ID, where the text will be picked up and linked to

```
<div class="scs-cookie-consent" style="display: none;">This site uses cookies.  Please review our
<a href="[!--$SCS_PAGE--]500[/!--$SCS_PAGE--]">Privacy Policy</a>.  <a href="#" alt="close"
class="scs-cookie-consent-close"></a></div>
```

Privacy Policy Page

The text will link to the privacy policy page, which has advisory text.



Inner HTML for Search Engine Optimization

When you save component data in Site Builder, you can save the Inner HTML that would be produced if the component were rendered at runtime. This Inner HTML is stored in the page data so that when the page is rendered, the data can be inserted into the page in place of the component. This can happen very early in the page rendition, allowing a search engine more chance to successfully crawl the page content.

Understand the components.json File and Format

The components.json file is used for themed components. A theme must have a components.json file located at `/ThemeName/components.json`, which specifies the components used in the theme. This is different from the appInfo.json file, which is used when a component is dropped onto a page and there is no entry in the components.json file. With the components.json file, you could have different themes with different values for the component in each theme. However with the appInfo.json file, there is only one value for a component.

The components.json file must contain valid JSON, and the minimum that the file must contain is an empty JSON array `[]`.

The `components.json` file syntax lists all local components and fully supports categorization of components. (Remote components are registered in the Component Catalog.)

No matter what components are added to the theme-level `components.json` file (including none), Oracle Content Management populates a default set of components available to users. This default set is defined in the source code. The following list shows the components and (seeded) components rendered in inline frames. In addition, any remote components registered at the service level and made available to users in your instance will be available in Site Builder.

The following local components are included with Oracle Content Management.

Name	Type	ID
Title	scs-title	scs-title
Paragraph	scs-paragragh	scs-paragragh
Text	scs-title	scs-text
Image	scs-image	scs-image
Gallery	scs-gallery	scs-gallery
Gallery Grid	scs-gallerygrid	scs-gallerygrid
Document	scs-document	scs-document
Button	scs-button	scs-button
Map	scs-map	scs-map
Divider	scs-divider	scs-divider
Spacer	scs-spacer	scs-spacer
YouTube	scs-youtube	scs-youtube
Social Bar	scs-socialbar	scs-socialbar
Video	scs-video	scs-video

Name	Type	ID
Dynamic List	scs-dynamiclist	scs-dynamiclist
Recommendation	scs-recommendation	scs-recommendation
Article (custom component)	scs-component	scs-comp-article
Headline (custom component)	scs-component	scs-comp-headline
Image and Text (custom component)	scs-component	scs-comp-image-text

These components, rendered in inline frames, are included with Oracle Content Management. They don't include registered remote components.

Name	Type	ID
Conversation	scs-app	Conversation
Documents Manager	scs-app	Documents Manager
Project Library	scs-app	Project Library
Conversation List	scs-app	Conversation List
Start Form	scs-app	Start Form
Task List	scs-app	Task List
Task Details	scs-app	Task Details
Folder List	scs-app	Folder List
File List	scs-app	File List
Facebook Like	scs-app	Facebook Like
Facebook Recommend	scs-app	Facebook Recommend
Twitter Follow	scs-app	Twitter Follow
Twitter Share	scs-app	Twitter Share

General Format

The general format of the `components.json` file follows:

- Properties for components are specified within each component. Top-level `"components"` or `"apps"` properties are deprecated.
- Each component has a `"type"` property. Components can only have certain values (all possible values are listed in the table for default components).
- Each component has an `"id"` property, which must be unique. This property is used to distinguish between components with the same `"type"`. Previously, apps had the `"appName"` property. While `"appName"` still works if the `"id"` property is not available, the `"appName"` property is deprecated.
- Each component has a `"name"` property that is the display name in the user interface. If fallback values are not specified, for components the value is the name of the corresponding default component, and for remote components the value is the ID.

Here is an example of a `components.json` file:

```
[
    {
        "name": "COMP_CONFIG_TEXT_CATEGORY_NAME",
        "list": [
            {
                "type": "scs-title",
                "id": "my-headline",
                "name": "My Headline",
                ...
            },
            {
                ...
            },...
        ]
    },
    {
        "name": "My own category name",
        "list": [ ... ]
    }
]
```

The general structure is a JSON array of category objects. Each category object has a `"name"` property and `"list"` property. The `"name"` property can be a key that maps to a localized String. If these default categories are not sufficient, you can provide your own category name, which won't be localized. The following table lists available default categories and corresponding keys.

Key	Category Name (English)
COMP_CONFIG_CONTENT_CATEGORY_NAME	Content
COMP_CONFIG_CUSTOM_CATEGORY_NAME	Custom
COMP_CONFIG_MEDIA_CATEGORY_NAME	Media
COMP_CONFIG_SOCIAL_CATEGORY_NAME	Social
COMP_CONFIG_TEXT_CATEGORY_NAME	Text

The `"list"` property in each category object contains an array of component objects. Each component or object must have `"type"` and `"id"` properties. Other properties are optional.

- The `"type"` property must be equal to one of the types found in the default components. If the `"type"` does not already exist, the component won't be displayed.
- The `"id"` property must be unique across components. If the `"id"` is found to already exist, the component won't be displayed.
- The `"name"` property is the display name for the component in the user interface. This replaces the previous `"appName"` property for apps (now remote components).
- All other properties are treated the same as in previous releases.

Add New Components to components.json

You are not allowed to modify the default components. However, you can create a new component based on an existing default component. For example, you could create a new component based on the `"scs-title"` component, which sets some default text. The minimum required to add a new component is to specify the `"type"` and `"id"` properties.

- The `"type"` must be equal to one of the types found in the default components. If the `"type"` does not already exist, the component won't be displayed.
- The `"id"` must be unique across components. If the `"id"` is found to already exist, the component won't be displayed.

Here's an example of code to add a new Title component. This component will display along with the default title component.

```
[
    {
        "name": "COMP_CONFIG_TEXT_CATEGORY_NAME",
        "list": [
            {
                "type": "scs-title",
                "id": "my-headline"
            }
        ]
    }
]
```

Here's an example of code to add a new Title component with a display name and default text.

```
[
    {
        "name": "COMP_CONFIG_TEXT_CATEGORY_NAME",
        "list": [
            {
                "type": "scs-title",
                "id": "my-headline",
                "name": "My Headline",
                "initialData": {
                    "userText": "This is a second title component"
                }
            }
        ]
    }
]
```

Note that the title component takes all the properties of the default Title component as the base, and applies theme-level modifications on top of it to create the new component.

Backwards Compatibility

The `components.json` files in the previous format can still be read.

- Files with top-level `"components"` or `"apps"` properties.
- If the file contains an `"apps"` property, user-defined remote components under this property are still loaded.
- If the file contains a top-level `"apps"` property, then assume any remote components listed beneath have type `"scs-app"`.
- If the `"appName"` property is present, set `"id"` to the `"appName"` value. The display name will be the same as `"name"`, if specified, or falls back on the `"id"` value.

Customize Toolbar Groups in Site Builder

For custom components, you can customize copies of out-of-the-box toolbar groups displayed in the Site Builder toolbar.

You can create your own version of an out-of-the-box component that has a restricted set and is available on the custom tag. Any customizations do not affect the out-of-the-box components.

You can create your own version of the out-of-the-box component that has a restricted set and is available on the custom tag, but it won't affect any of the out-of-the-box components.

Toolbar groups define what you see in the toolbar when you click the Title or Paragraph component to edit it. You can remove and reorder what is supported by the Title or Paragraph component, but additional plug-ins are not allowed.

To customize toolbar groups in Site Builder, you need to use the same syntax as the rich text editor uses for its toolbarGroups configuration. TinyMCE syntax is supported.

Title and Paragraph components support the following groups:

- "basicstyles" - restricted to bold/italic/underline
- "styles" - Font Styles
- "colors" - Text and Background colors
- "undo" - Undo/Redo of current instance in the rich text editor
- "links" - Custom Plugin to link dialog
- "paragraph" - bullet/numbered list and indentation support
  - "list"
  - "indent"
- "align" - left/right/center
- "cleanup" - remove any styles for selected text

The Paragraph component also supports the image and table insert plug-ins”

- "insert"
  - "image"
  - "table"

In addition, you can use the row separator entry:

- "/"

> **Note:**
>
> If you set any other value in the toolbar-group configuration, the value will be removed before the Site Builder toolbar is created. You cannot provide "extraPlugins". Only the "name"/"groups" configuration is supported. Any "items" entries will be ignored.

For example, if you want to prevent your users from defining fonts, colors, styles, or sizes, you can update the toolbar configuration as follows. For the "id" values, you need to specify custom values that are different from out-of-the-box values.

```
[{
    "name": "<category name>",
    "list": [{
        "type": "scs-title",
        "id": "<custom-value>",
        "config": {
            "toolbarGroups": [{
                    "name": "basicstyles",
                    "groups": ["basicstyles"]
                }, {
                    "name": "undo",
                    "groups": ["undo"]
                },
                "/", {
                    "name": "links",
                    "groups": ["links"]
                }, {
                    "name": "paragraph",
                    "groups": ["list", "indent"]
                },
                "/", {
                    "name": "align",
                    "groups": ["align"]
                },
                {
                    "name": "insert",
                    "groups": ["image", "table"]
                }, {
                    "name": "cleanup",
                    "groups": ["cleanup"]
                }
            ]
        }
    }, {
        "type": "scs-paragraph",
        "id": "<scs-paragraph>",
        "config": {
            "fontSize_sizes": "16/16px;24/24px;48/48px;"
```

```
        }
    }]
}]
```

Validation

Replace your `components.json` file with the preceding code and then edit your site (refresh the browser if you are already editing). At this point, when you edit a Title component, it will no longer show the font styles or colors for selection. The Paragraph component will continue to show these, and the list of font sizes available will be limited to 16, 24, and 48.

Default Toolbar Groups

The default toolbar groups for Title and Paragraph follow:

- Title

```
[{
        "name": "basicstyles",
        "groups": ["basicstyles"]
    }, {
        "name": "styles",
        "groups": ["styles"]
    }, {
        "name": "colors",
        "groups": ["colors"]
    }, {
        "name": "undo",
        "groups": ["undo"]
    },
    "/", {
        "name": "links",
        "groups": ["links"]
    }, {
        "name": "paragraph",
        "groups": ["list", "indent"]
    }, {
        "name": "align",
        "groups": ["align"]
    }, {
        "name": "cleanup",
        "groups": ["cleanup"]
    }
]
```

- Paragraph

```
[{
        "name": "basicstyles",
        "groups": ["basicstyles"]
    }, {
        "name": "styles",
        "groups": ["styles"]
    }, {
        "name": "colors",
```

```
            "groups": ["colors"]
        }, {
            "name": "undo",
            "groups": ["undo"]
        },
        "/", {
            "name": "links",
            "groups": ["links"]
        }, {
            "name": "paragraph",
            "groups": ["list", "indent"]
        }, {
            "name": "align",
            "groups": ["align"]
        }, {
            "name": "insert",
            "groups": ["image", "table"]
        }, {
            "name": "cleanup",
            "groups": ["cleanup"]
        }
    ]
```

Restrict Components in Slots

For any layout slot, you can specify certain restrictions on the components allowed in the slot.

If you restrict components in a slot, any user dragging a component that is not allowed will see a warning message and will not be able to add or move a component to that slot.

To configure this restriction, you edit the layouts in your theme (for example, a layout file `themes\theme_name\layouts\oneslot.htm`) and add custom data attributes to the slot div.

This is the format of the custom attributes. The main difference from previous versions is that users must specify only the `id` of the component. Previously to restrict a component, users had to use the verbose `"<type> <id>"` syntax (which is still supported).

```
data-allowed-items='["<id>:","<type>,"<type>:<id>",...]'
data-disallowed-items='["<id>":"<type>","<type>:<id>",...]
```

Note:

The value for `data-allowed-items` and `data-disallowed-items` can use double quote marks (") or single quote marks (‘). In the following example, `data-allowed-items` uses single quote marks around the JSON array, while `data-disallowed-items` uses double quote marks around the JSON array:

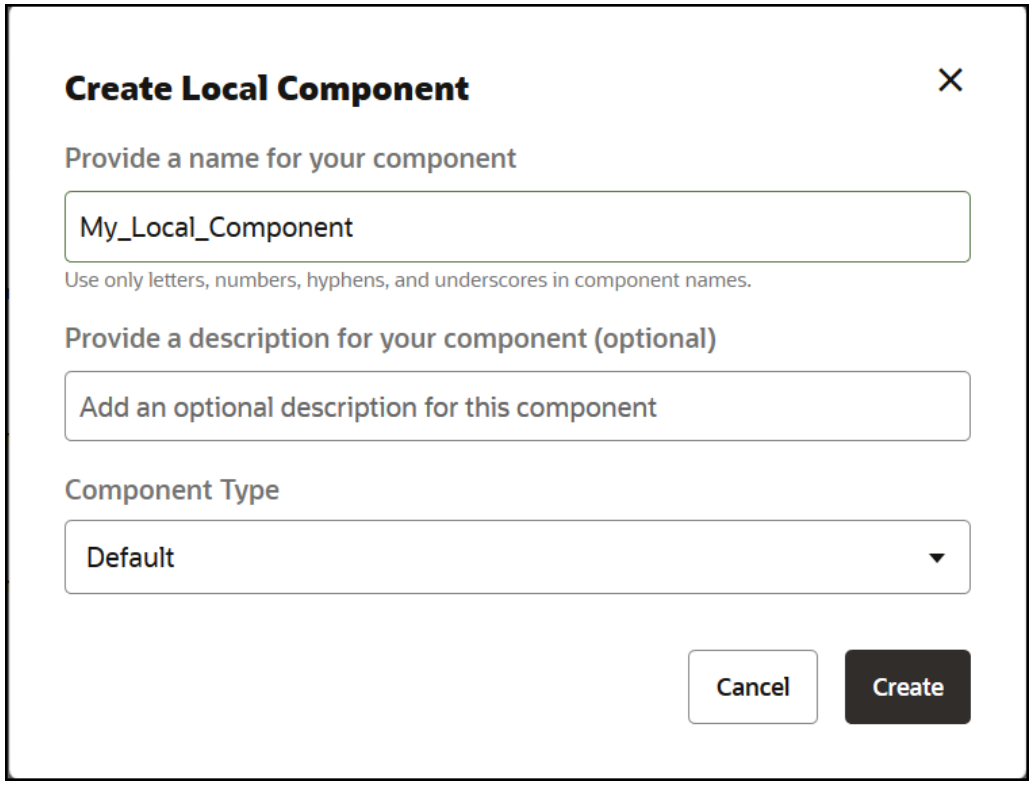
```
<div id="slot101"
     class="scs-slot"
     data-allowed-items='["scs-app","scs-title"]'
     data-disallowed-items="['File List', 'scs-map']">
</div>
```

The following table lists components and their respective IDs provided with Oracle Content Management. To prevent any naming conflict, *do not* prefix any customized (local or remote) component ID with `scs-` or use any type or ID listed in this table.

Name	Type	ID
Documents Manager	scs-app	Documents Manager
Facebook Like	scs-app	Facebook Like
Facebook Recommend	scs-app	Facebook Recommend
File List	scs-app	File List
Folder List	scs-app	Folder List
Twitter Follow	scs-app	Twitter Follow
Twitter Share	scs-app	Twitter Share
Button	scs-button	scs-button
Article (custom component)	scs-component	scs-comp-article
Headline (custom component)	scs-component	scs-comp-headline
Image and Text (custom component)	scs-component	scs-comp-image-text
Component Group	scs-componentgroup	scs-componentgroup
Content Search	scs-contentsearch	scs-contentsearch
Content List	scs-contentlist	scs-contentlist
Content Placeholder	scs-component	scs-contentplaceholder
Content Item	scs-component	scs-contentitem
Divider	scs-divider	scs-divider
Document	scs-document	scs-document
Gallery	scs-gallery	scs-gallery
Gallery Grid	scs-gallerygrid	scs-gallerygrid
Image	scs-image	scs-image
Map	scs-map	scs-map

Name	Type	ID
Paragraph	scs-paragraph	scs-paragraph
Social Bar	scs-socialbar	scs-socialbar
Spacer	scs-spacer	scs-spacer
Title	scs-title	scs-title
YouTube	scs-youtube	scs-youtube

Users can create local or remote components. The name provided in this sample (`My_Local_Component`) is the ID that can be used to specify this component for restricting inside slots.



Make Layout Content Editable

You can configure certain text or image content in a layout to make it editable by users working with pages based on the layout.

This functionality can be used in any theme, including bootstrap themes. The modifications automatically assume the style of the original page.

You can add simple markup to the following HTML tags in the layout: `<p>`, `<h1>` through `<h6>`, `<div>` and `<img>`.

Text and image formatting options include:

- Text: Allows users to specify Bold, Italic, Underscore, and Link.

- Image: Allows users to specify a link to an image file. Change properties for title (what the user sees when hovering the cursor over an image) and add alternate text for accessibility.

Modifying a text or image element so it can be edited by users requires two things:

1. Add `scs-editable` to the `class`.
2. Add a unique `id` attribute.

Once a layout has been modified, the functionality will be available for all pages based on that layout (even new pages).

If you copy and paste a page, the modifications will be copied to the new page.

Modifying a Heading Tag

Here's an example of how to modify a heading tag in a layout so it can be edited by users.

1. Sync the layout file to your local desktop, or edit the HTML source file for the `<h1>` header.
2. Add `scs-editable` to the `class`, and add the attribute `id="test-heading"` in the line of code for the heading, so it allows an editor to be attached to it. For example:

   ```
   <h1 class="brand-heading scs-editable" id="test-heading">Sample Heading
   Value</h1>
   ```

3. Save the file.
4. Sync up with Oracle Content Management and reload the browser.

   A black border should appear around the heading when the user hovers the cursor over it, indicating that the content can be edited. When a user clicks the heading, the border becomes green, indicating the user can now edit the heading content.
5. If the theme hasn't been published, do so. If it has been published, the change will appear when you refresh your browser.

   Once the theme is published, site users can click the header and edit it in the Site Builder.

   A pseudo component is created for the `scs-editable` element so that you can change it in Site Builder and store it with the page data. At runtime, before the page is rendered, the controller replaces the scs-editable tags with the values you set in Site Builder.

Modifying an Image Tag

The procedure to modify an image tag in a layout so it can be edited by users is similar to that for text.

1. Add `scs-editable` to the `class`.
2. Add a unique image `id`.

Users can click the image, then change properties to use a different image.

Here's sample code for an image that can be edited by users:

```
<img class="scs-editable" id="test-image" src="_scs_theme_root_/assets/img/
downloads-bg-small.jpg"/>
```

Create a Section Layout

Create a section layout to arrange content within a slot on a site page.

An enterprise user can arrange content items on a site based on section layouts that you provide as a developer. You can create new section layouts from the default layout.

You can export a section layout to modify it offline and then import it either as a new section layout or to replace the existing section layout. Export the section layout individually or as part of a template package that includes custom components and layouts.

The following out-of-the-box section layouts are available:

- Horizontal
- Two Column
- Three Column
- Vertical
- Tabbed
- Slider

You can use these right away in Site Builder without having to create anything.

The files for these section layouts have comments with more details about the structure of section-layout files. To see the comments, you can create a new section layout based on an out-of-the-box one and then export the new layout for editing, as described in the following procedure.

To create a section layout:

1. On the home page, click **Developer**.
2. Click **View all Components**.
3. From the **Create** drop-down menu on the right, choose **Create Section Layout**.
4. In the **Create Section Layout** dialog box, provide a name and description for your section layout component.
5. To export the section layout for editing, select it, and then click **Export** in the right-click menu or click  in the actions bar.
    a. Navigate to an existing folder, or click **Create** to create a new folder and provide a name and, optionally, a description.
    b. Select the checkbox next to the folder, and click **OK**.
    c. Click the folder's icon or name to open it.

    A layout package file is created in the selected folder with the section layout name and a `.zip` extension. Download the file to your development environment to edit the files.

    You can find information about the Section Layouts API in Develop Custom Section Layouts with APIs.

6. Import your modified files either as a new section layout or to replace the existing section layout.
    a. On the home page, click the **Content** tab, and then click **Documents**.
    b. Upload the modified section layout to a folder, in a file with a **.zip** extension, which includes the same folder and file names that you exported.
    c. On the **Developer** page, click **View all Components**.
    d. From the **Create** menu, choose **Import Component**.
    e. Select the checkbox next to the uploaded zip file that contains the modified section layout, and click **OK**.

    Your modified section layout is imported to the folder you selected.

You can also export a section layout to copy or move it to another Oracle Content Management instance and import it there.

Create a Section Layout That Supports Lazy Load

The content list can call the section layout with additional components as they get queried.

When you render a content list, you have the option of selecting a section layout to render all the content items that are returned. This enables you to create various different layouts for the content items, such as a table, a slider, or an eight-column layout. These custom section layouts can also take part in the more advanced pagination features.

Content lists support the following pagination:

- **Pagination**
- **Load on scroll**
- **Load on click**

For the standard pagination feature, the section layout doesn't need to do anything. It will be re-rendered with the next set of items when the user clicks the next page. However, for **Load on scroll** and **Load on click**, rather than having the section layout re-render, additional components are added to the section layout. This is used mostly for the infinite scrolling model, where you load the first *n* items and, as the user scrolls down the page, you fetch and render the next set of items. To support **Load on scroll** and **Load on click**, the custom section layout needs to do

1. `render.js`: Implement the `addComponent()` API. This will be called with each new component that is to be added to the section layout.

```
// dynamic API for adding additional components through "load more" when
used in a Content List
           addComponent: function (parentObj, component) {
               // create the component div and add it to the parent object
               $(parentObj).append(this.createComponentDiv(component));
           }
```

2. `appInfo.json`: Include the following to let the content list know that the section layout supports the `addComponent() api`.

```
"contentListData": {
  "addComponent": true
},
```

Once the `appInfo.json` is updated, when the user selects this section layout in the settings panel and goes to the pagination screen, they will see the `Load on click` and `Load on scroll` options.

Develop Custom Section Layouts with APIs

You can develop custom section layouts in Oracle Content Management with the Section Layout API, which includes Rendering APIs and Editing APIs.

For starter files to look at, see Create a Section Layout. The starter files for section layouts include comments with details about the structure of section-layout files.

Rendering APIs

The Rendering APIs, loaded from the `render.mjs` module, are used in Site Builder and at runtime.

Rendering API	Description	Input Parameter(s)	Return Result
(Constructor)	Initializes the Section Layout rendering module.	A JavaScript object that contains the following properties: • **sectionlayoutData (Object)** : The section layout data found in the page model. • **componentId (String)**: The `componentId` value of the section layout, typically a GUID. • **renderMode (String, optional)**: The render mode for the rendering operation. • **customSettingsData (Object)**: A copy of the `customSettingsData` found in `sectionLayoutData`.	The Section Layout Rendering APIs are initialized.
render	Emits DOM elements appropriate for the section layout to the page, including container DIVs for child components.	**container (Element)** : The DOM element into which the section layout's markup should be rendered.	After this method returns, child components will be rendered. You can identify child components by finding child `div[id]` elements

Rendering API	Description	Input Parameter(s)	Return Result
addComponent	Used with content list components to dynamically add child components to a section layout. This function is optional.	**container (Element)**: The DOM element into which the new component should be rendered. **componentId (String)**: The ID of the new component to add to the section layout.	After this method returns, the element whose ID matches the `componentId` input will be rendered.

Editing APIs

The `edit.mjs` module is loaded *if* the `hasEditHandlers` property is set to `true` in the `appinfo.json` file associated with the section layout.

The Editing APIs are used in Site Builder.

All of the functions in this module except the Constructor are optional.

Editing API	Description	Input Parameter(s)	Return Result
(Constructor)	Initializes the Section Layout editing module.	A JavaScript object that contains the following property: • **componentId (String)**: The `componentId` value of the section layout, typically a GUID.	The Section Layout Editing APIs are initialized.

Editing API	Description	Input Parameter(s)	Return Result
getCapabilities	Returns an object describing the editing capabilities of the section layout.	A JavaScript object that describes the editing capabilities of the section layout. Upon input, the default capabilities will be provided to the function. The function can modify the Capabilities object as needed. The Capabilities object can include the following capabilities: • **title (String)**: The title of the section layout to display to the user. • **settingsTitle (String)**: The title to display in the Settings Panel dialog. • **hasSettings (Boolean)**: Indicates if the section layout supports a Settings Panel. • **allowMove (Boolean)**: Indicates if the section layout allows child items to be moved. • **allowDelete (Boolean)**: Indicates if the section layout allows child items to be deleted. • **isHidden (Boolean)**: Indicates if the section layout is currently hidden in response to user options. • **dropTarget (Boolean)**: Indicates if the section layout is the target for drag and drop operations. • **customMenuOptions (Array)**: Custom menu options to add to the Section Layout context menu. Each menu-option object has the following properties: – **label (String)** : The display text of the menu item. – **action (Function)**: The function to invoke when the menu item is clicked. – **disabled (Boolean)**: Indicates that the menu item should display in a disabled state. – **icon (String)**: The URL to display alongside the label in the menu item. (This property is reserved for future use.) – **checkmark (Boolean)**: Indicates that a checkmark should appear alongside the label in the menu item. – **subMenuItems (Array)**: Menu options to display in a submenu.	(Object): The capabilities for the section layout.

Editing API	Description	Input Parameter(s)	Return Result
getCaptionContent	Returns the section layout display name, which will appear in UI elements.	None.	(String): The display name of the section layout.
filterCapabilities	Allows the section layout to modify the Capabilities object before menus are displayed to the user. You can use this API to adjust or remove menu options. (See also **getCapabilities**.)	A JavaScript object that describes the editing capabilities of the section layout. On input, the default capabilities will be provided to the function.	(Object) The capabilities for the section layout.
onDragOver	Called during a Drag and Drop operation to indicate if the dragged item can be dropped on the section layout.	**eventObject (Event Object)**: An event object that holds information about the drag event. **dataTransfer (DataTransfer Object)**: A `DataTransfer` object that holds information about the item being dragged over the section layout.	(Boolean) A value indicating if the dragged item can be accepted by the section layout. Returns *true* if the section layout can accept the dragged item, *false* otherwise.
onDrop	Called during the drop portion of a Drag and Drop operation to indicate that the dragged item should be placed inside the section layout.	**eventObject (Event Object)** : An event object that holds information about the drop event. **dataTransfer (DataTransfer Object)** A DataTransfer object that holds information about the item being dropped on the section layout.	(Boolean) A value indicating if the drop operation was handled by the section layout. Returning *true* bypasses the default logic.
onAddComponent	Notifies the Section Layout that a Drag and Drop operation added an item in the section layout.	**eventObject (Event Object)**: An event object that holds information about the drag event. **dataTransfer (DataTransfer Object)**: A DataTransfer object that holds information about the item being dropped on the section layout. **componentId (String)**: The `componentId` value of the newly added item.	Section layout notification.

Editing API	Description	Input Parameter(s)	Return Result
onMoveComponent	Notifies the section layout that a Drag and Drop operation moved an item in the section layout.	**eventObject (Event Object)**: An event object that holds information about the drag event. **dataTransfer (DataTransfer Object)**: A DataTransfer object that holds information about the item being dropped on the section layout. **componentId (String)**: The `componentId` value of the moved item.	Section layout notification.
getSettingsData	Allows the section layout to change the Settings data before the Settings Panel is displayed.	**settingsData (Object)**: The default settings data computed for the Section Layout	(Object) The settings data for the section layout.
updateSettings	Allows the section layout to change its settings after the Settings Panel has been closed. This API is called just before the settings are stored in the page model.	**parameters (Object)**: The raw parameters object returned from the Settings Panel. **sectionLayoutData (Object)** : The section layout data that will be stored. Default data will be generated from `parameters (Object)` and passed to the function in this parameter.	(Object) The section layout data to store in the page model.
dispose	Allows the editing module to free memory, detach events, and deallocate resources associated with the edit handlers. This API is called when the section layout needs to completely redraw, as in the case of an Undo/Redo operation.	None.	Redrawing the section layout is enabled.

Develop Content Layouts

Content layouts help users view the data in content items through content list or content placeholder components used in sites pages. You can create multiple content layouts for a content type to create different views or to represent different parts of a content item.

For example, a Blog-Post content type may want different content layouts depending on how and where the Blog-Bost content is to be used. The site's home page may display a list of Blog-Post items, but when a blog post is clicked on the home page, the **Details** page may show details about that blog post.

The Home page has a content list configured to list items of the Blog-Post content type, using the Blog-Post-Summary content layout as an item view.

The Details page uses the Blog-Post-Header content layout in a content placeholder to show a header image and title. The two-column section layout encloses two content placeholders, in 70 percent and 30 percent widths, using the Blog-Post-Content and Blog-Post-Author content layouts. As you can see, four different content layouts are used to visualize the same content Type.

You can create a content layout in either of two ways:

- In Oracle Content Management choose **Developer** > **View all Components** > **Create** > **Create Content Layout**.
- In an Content Toolkit project, use the `cec create-contentlayout` command.

Oracle Content Management creates a default content layout for the content type. To modify the default content layout, you can edit the following files:

- `assets/layout.html`

  Edit this file to change the HTML view.

- `assets/design.css`

  Edit this file to style the content layout.

- `assets/render.mjs`

  Edit this file to change the data used in `layout.html` or to add dynamic behavior to the content layout.

Content layout components render a content item from the Oracle Content Management server. Most of the assets are stored in the Oracle Content Management server. Sometimes you may want to use a static asset that is locally available in the content layout itself, such as a background image for styling purposes. For example, in the content layout that follows, the absolute URL to `images/background.jpg` can be generated in `render.mjs` and used in `layout.html`.

For render.mjs, the simplest way to generate the absolute URL is to use the built-in import.meta.url value to generate the URL, as the following code shows:

```
const assetsFolder = import.meta.url.replace('/render.mjs', '');
const imageURL = `${assetsFolder}/images/background.jpg`;
```

The following topics describe how to develop content layouts:

- Create Content Layouts with Oracle Content Management
- Pass a Layout View to a Content Layout
- Generate a Site Details Page URL with an API
- Develop Content Layouts Locally with Developer Cloud Service
- Expand Macros in Content List Queries
- Develop Robust Content Layouts
- Create the Sample Blog Template
- Add Content Layout Mappings to Templates
- Test Content Layouts with the Local Test Harness
- Import Templates with Content Layouts into Oracle Content Management

Related Topics

For information about how to manage content layouts, see Manage Custom Components and Layouts.

For information about using digital assets and other content items in a site, see Use Assets and Manage Digital Assets in *Managing Assets with Oracle Content Management*.

Create Content Layouts with Oracle Content Management

Create a content layout for laying out fields in a content item when displayed on a site page. When a content item is added to a page, it will use the chosen content layout.

An enterprise user can create and use content items based on content types and layouts that you provide as a developer. You can create new content layouts from the default layout. Multiple content layouts associated with the content type make it possible for the site designer to display content items in different contexts without changing the content.

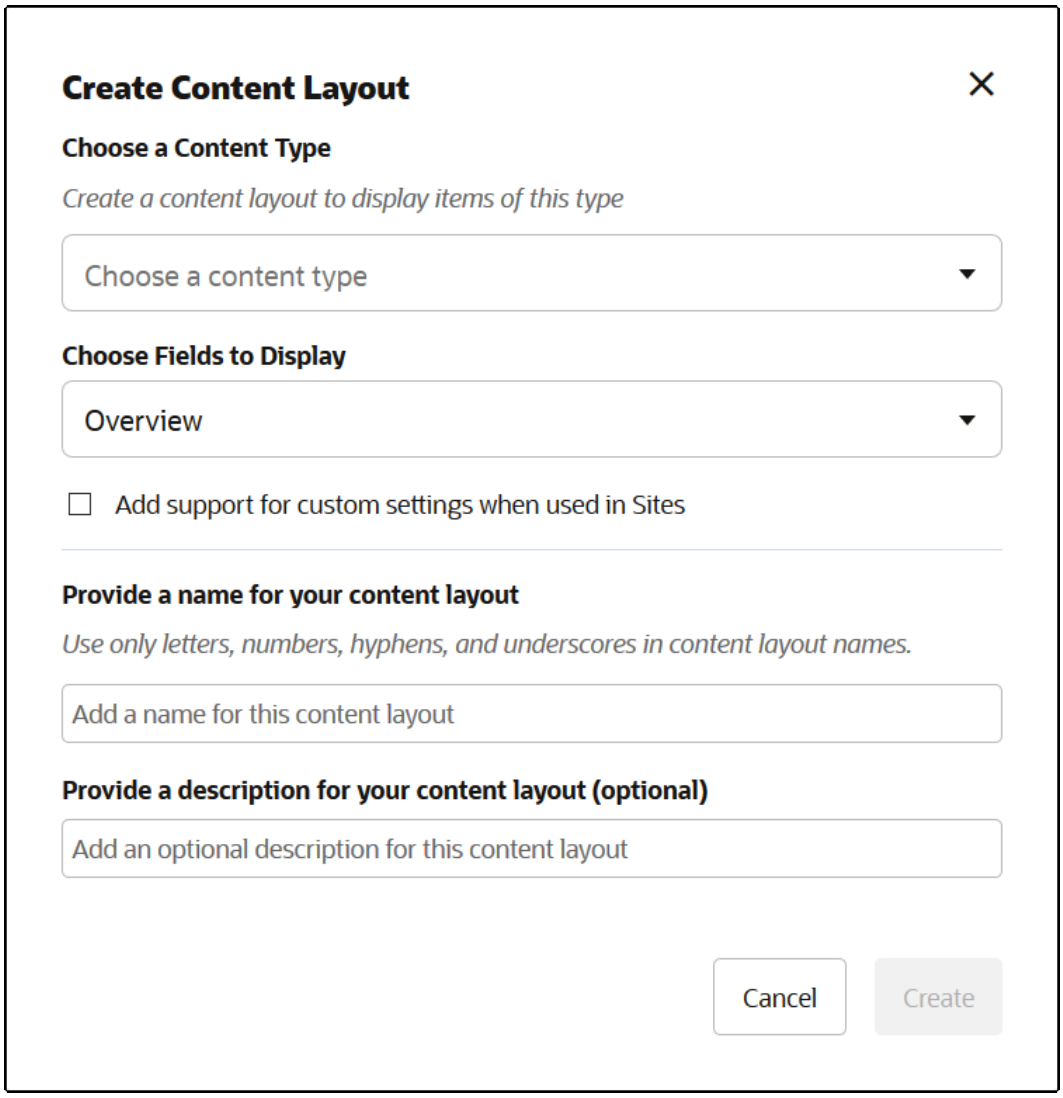
If you use a content layout in a content list component, then the content layout is repeated once per content item. The content layouts are then arranged by the section layout.

You can export a content layout to modify it offline and then import it to replace the existing content layout.

To create a content layout:

1. On the Oracle Oracle Content Management home page, click **Developer** in the side navigation.

   The **Developer** page is displayed.
2. Click **View all Components**.
3. From the **Create** drop-down menu on the right, choose **Create Content Layout**.
4. In the **Create Content Layout** dialog box, select the content types that will use the layout, choose the fields to display, and enable **Add support for custom settings when used in Sites** if you want site creators to be able to add custom settings and styles when adding content item and content list components to a page.



5. Provide a name and description for your content layout component and click **Create**.

   The content layout is added to your components.

> **Note:**
>
> Only alphanumeric characters, hyphens, and underscores are valid in content layout titles.

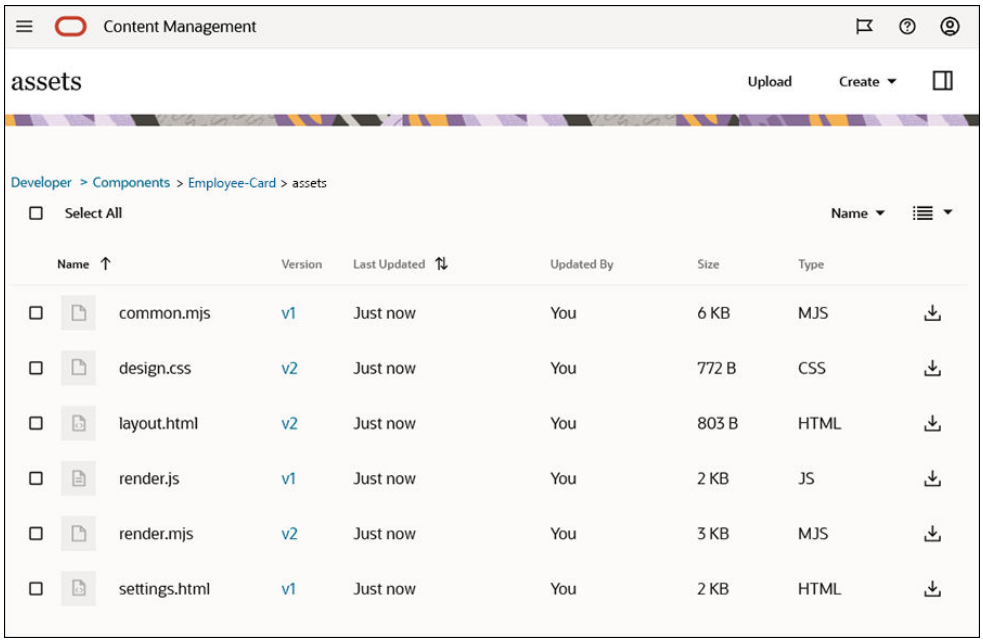
6. To export the content layout for editing, select it, and then click **Export** in the right-click menu or click  in the actions bar.

   a. Navigate to an existing folder, or select **Create** then **Folder** to create a new folder and provide a name and, optionally, a description.

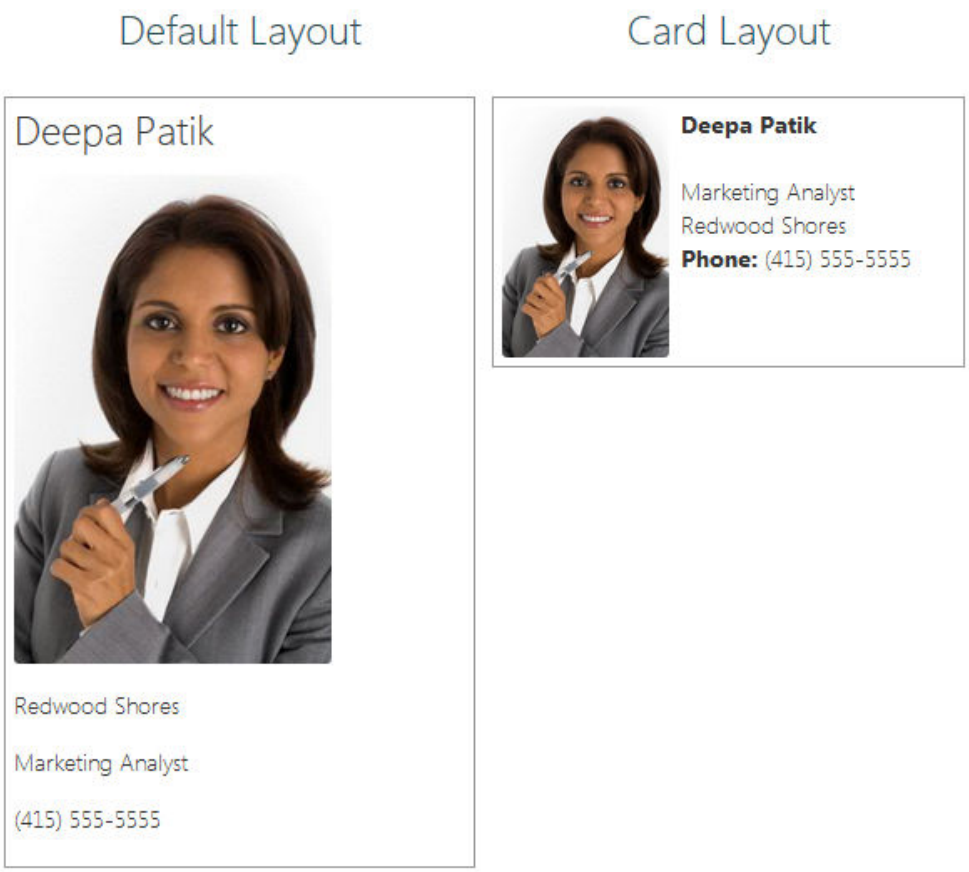
   b. Select the checkbox next to the folder, and click **OK**.

   c. Click the folder's icon or name to open it.

   A layout package file is created in the selected folder with the content layout name and a `.zip` extension. Download the file to your development environment to edit the files.



These files control the layout of the fields in content items that use the Employee-Card content layout. If you enabled the **Add support for custom settings when used in Sites**, then an additional file named **settings.html** is also created which provides a default rendering of a single content item so it can be displayed.



7. Edit the `design.css`, `layout.html`, and `render.js` files to get the content layout you want.

   For example, the following files specify the Employee-Card content layout:

   a. Edit the `design.css` file:

```
.scs-tile-layout {
    font-family: 'Helvetica Neue', 'Segoe UI', sans-serif-regular,
Helvetica, Arial;
    font-size: 16px;
    margin: 0px;
    padding: 0px;
    font-style: normal;
    color: #333;
}

.scs-tile-layout li {
    list-style: none;
    font-size: 14px;
    font-style: normal;
    font-variant-caps: normal;
    font-weight: 200;
    margin: 0px;
}

.scs-tile-layout-img-container {
    height: 150px;
    width: 100px;
    float: left;
    margin: 0em 0.5em 0em 0em;
    padding: 0px;
    border-radius: 3px;
    overflow: hidden;
    position: relative;
}

.scs-tile-layout-img {
    position: absolute;
    left: -100%;
    right: -100%;
    top: -100%;
    bottom: -100%;
    margin: auto;
    height: 100%;
    min-width: 100%;
}

.scs-tile-layout p {
    margin: 0px;
}
```

b. Edit the `layout.html` file:

```
{{#data.employee_profile_pictureURL}}
<div class="scs-tile-layout-img-container">
    <img class="scs-tile-layout-img"
src="{{data.employee_profile_pictureURL}}" />
</div>
{{/data.employee_profile_pictureURL}}
<ul class="scs-tile-layout">
    <li>
        <p><b>{{name}}</b></p>
    </li>
    <li> </li>
    <li>
        <p>{{data.employee_job_title}}</p>
        <p>{{data.employee_location}}</p>
        <p><b>Phone: </b>{{data.employee_phone}}</p>
    </li>
    {{#scsData.detailPageLink}}
    <li>
        <a href="{{scsData.detailPageLink}}" title="Go to detail
page"><span class="detail-page">Profile</span></a>
    </li>
    {{/scsData.detailPageLink}}
</ul>
```

c. Edit the `render.mjs` file:

```
/* globals Mustache */
import CommonUtils from './common.mjs';

export default class {
    // class variables
    contentVersion = ">=1.0.0 <2.0.0"

    // Content Layout constructor
    constructor(params) {
        // store passed in values
        this.contentItemData = params.contentItemData || {};
        this.scsData = params.scsData;
        this.contentClient = params.contentClient;
        this.libs = this.contentClient.getLibs() || {};

        // store path to the "assets" folder
        this.assetsFolder = import.meta.url.replace('/
render.mjs', '');

        // access resources
        this.Mustache = params.Mustache || this.libs.Mustache ||
window.Mustache;
    }

    // Main rendering function:
    // - Updates the data to handle any required additional
requests and support granular permissions
```

```
    // - Expand the Mustache template with the updated data
    // - Appends the expanded template HTML to the parentObj DOM
element
    render(parentObj) {
        const contentClient = this.contentClient;
        const commonUtils = new CommonUtils(contentClient);
        const noPermissionToViewMsg = "You do not have permission to
view this asset";

        let contentType;
        let customSettings;
        let secureContent = false;


        // extract the content that will be used as the model
        this.content = Object.assign({}, this.contentItemData);

        // If used with CECS Sites, Sites will pass in context
information via the scsData property.
        if (this.scsData) {
            this.content = Object.assign(this.content, {
                "scsData": this.scsData
            });
            contentType = this.content.scsData.showPublishedContent
=== true ? "published" : "draft";
            customSettings = this.content.scsData.customSettingsData
|| {};
        }

        //
        // Handle fields specific to this content type.
        //

        // If displaying detail items, get the IDs of any referenced
assets
        // we will do an additional query to retrieve these so we can
render them as well.
        const referedFields = [];
        const referedIds =
commonUtils.findReferenceFieldIds(referedFields, this.content.fields,
noPermissionToViewMsg);

        // Handle expansion of URLs and check permissions for access
to referenced digital asset
        const digitalAssetFields = ["starter-blog-author_avatar"];
        commonUtils.updateDigitalAssetURLs(digitalAssetFields,
this.content.fields, noPermissionToViewMsg);

        // Handle markdown expansion
        const markDownFields = ["starter-blog-author_bio"];
        commonUtils.updateMarkdownFields(markDownFields,
this.content.fields);

        // Handle richText expansion
        const richTextFields = [];
```

```
        commonUtils.updateRichTextFields(richTextFields,
this.content.fields);

        // Handle date field formatting
        const dateTimeFields = [];
        commonUtils.updateDateTimeFields(dateTimeFields,
this.content.fields);


        //
        // Fetch any referenced items and resources used to
render the content layout
        //
        Promise.all([
            commonUtils.getRefItems(referedIds),
            contentClient.importText(this.assetsFolder + '/
layout.html'),
            contentClient.importCSS(this.assetsFolder + '/
design.css')
        ]).then((resources) => {
            const results = resources[0];
            const templateHtml = resources[1];

            // Store the retrieved referenced items in the model
used by the template.
            commonUtils.addReferencedItems(referedFields,
results, this.content.fields);

            // apply the model to the template
            try {
                // Use Mustache to expand the HTML template with
the data.
                const template =
this.Mustache.render(templateHtml, this.content);

                // Insert the expanded template into the passed
in container.
                if (template) {
                    parentObj.insertAdjacentHTML('beforeend',
template);
                }
            } catch (e) {
                console.error(e.stack);
            }
        });
    }
}
```

For information about editing the render.js and other files, see Develop Components.

8. Import your modified files to replace the existing content layout.
   a. On the home page, click **Documents**.

b. Upload the modified content layout to a folder, in a file with a **.zip** extension, which includes the same folder and file names that you exported.

   If you want to import it as a new content layout, you need to change the GUID of the content layout in `_folder.json`.

c. On the home page, click **Developer**.

   The **Developer** page is displayed.

d. Click **View all Components**.

e. From the **Create** menu, choose **Import Component**.

f. Select the checkbox next to the uploaded zip file that contains the modified component, and click **OK**.

Your modified content layout is imported to **Components**.

You can also export a content layout to copy or move it to another Oracle Content Management instance and import it there.

Pass a Layout View to a Content Layout

When you develop a content layout, you can get at properties for the underlying component if the content layout is being used in sites.

For this specific use case, you can get at the content layout view chosen for the content type in a content list. Then you can alter the way the component renders depending on what the category is. Without access to this property, you would need to create two content layouts that are effectively the same.

The `contentLayoutCategory` property is available for content layouts rendered for both the content item and the content list components. You can access this property through the Sites SDK, as follows.

```
scsData.SitesSDK.getProperty('contentLayoutCategory', function
(layoutCategory)
          { console.log(layoutCategory);});
```

See the Oracle Content Management SDKs.

This property is available only when the content layout is being rendered from a content item or content list, which are in an Oracle Content Management site. It isn't available when the content layout is rendered from a third-party app.

Generate a Site Details Page URL with an API

If you're rendering a list of content items in a content layout from your own query, you can create a link to a detail page for a content item. You can use the Sites SDK `SCSRenderAPI.getPageLinkData` API to generate a **Site Details** page URL.

The `detailPageId` property is the ID of the detail page selected in the content item or the content list. If the value wasn't set, the value returned is the first page in the `SiteStructureMap` that has the `isDetailPage` property set.

To access this property, you can use the Sites SDK. This is available only when the content layout is used for an Oracle Content Management site. It can be accessed only through

scsData, which is passed as one of the arguments when the content layout is created. For example:

```
scsData.SitesSDK.getProperty('detailPageId', function (detailPageId)
{   console.log(detailPageId);});
```

Once you have the `detailPageId`, you can use it to construct the link to the detail page.

The SCSRenderAPI has a function, `getPageLinkData()`, that takes in a `pageId` and additional options and constructs the required URL to the page passing through the options. The signature for this function follows:

```
SCSRenderAPI.getPageLinkData(pageId,
      options);
```

It has the following parameters:

- **pageId:** The same as the `detailPageId` returned from the Sites SDK `detailPageId` property.
- **options:**
  - `contentType`
  - `contentId`
  - `contentName`

The return value is an object with these properties:

- `hideInNavigation`
- `href`
- `href`

The next example puts this all together:

```
scsData.SitesSDK.getProperty('detailPageId', function (detailPageId) {
  var pageDetails = SCSRenderAPI.getPageLinkData(pageId,     {
        'contentType': contentType,
        'contentId': contentId,
        'contentName': contentItemData.slug || contentItemData.name
    });


   // get the URL to the page
   console.log(pageDetails.href);
});
```

This would print out as: "`/sites/{site}/{detailPageName}/{contentType}/{contentId}/{contentSlug}`"

If the `pageId` is not a detail page, then the content values are not added to the URL.

See the Oracle Content Management SDKs.

Develop Content Layouts Locally with Developer Cloud Service

You can use Developer Cloud Service through the Oracle Content Management Toolkit to create, edit, configure, and test content layouts locally for Oracle Content Management.

Take the following steps to prepare for developing content layouts with Developer Cloud Service:

1. Set Up Oracle Content Management Toolkit on Your Local Machine.
2. Sign in to the Developer Cloud Service Console for Oracle Content Management
3. Create a Project in Developer Cloud Service.
4. Add Oracle Content Management Toolkit to the Project Code in the New Git Repository.
5. Create a Content Layout

The following topics describe how to use the Content Toolkit to develop a content layout locally with Developer Cloud Service:

- Create a Content Layout with Developer Cloud Service
- Define the RequireJS module
- Configure the Constructor Function Parameter
- Render the Content Layout
- Edit the Content Layout in the Mustache Template
- Add Dynamic DOM Manipulation
- Define Styles in the design.css File
- Get Reference Items
- Get a Media URL
- Raise Triggers
- Navigate to a Search Page with a Search Query
- Expand Macros and Render Rich Text
- Link to the Details Page

After you develop your content layout with the Developer Cloud Service template, merge changes with your project's Git repository.

Create a Content Layout with Developer Cloud Service

You can create a content layout for a content type in your Developer Cloud Service project with the `cec create-contentlayout` command.

To create a content layout in your Developer Cloud Service project:

1. In a terminal window, go to the `cec-components` directory.
2. Enter `cec create-contentlayout` to see options and examples for the command:

```
Usage: cec create-contentlayout <name>

Creates a content layout based on a content type from a local template or
```

```
from CEC server.
By default, an "overview" content layout is created. Optionally
specify -s <style>
to create in a different style.

Valid values for <style> are:
  detail
  overview

Options:
  --contenttype, -c  <contenttype> Content layout is based
on         [required]

  --template, -t       <template> Content type is from
  --server, -r         flag to indicate the content type is from
server
  --style, -s          <style> Content layout style
  --addcustomsettings, -a  Add support for custom settings when
used in Sites
  --help, -h           Show
help                                          [boolean]

Examples:
  cec create-contentlayout Blog-Post-Overview-Layout -c Blog-Post -
t BlogTemplate
  cec create-contentlayout Blog-Post-Detail-Layout -c Blog-Post -t
BlogTemplate -s detail
  cec create-contentlayout Blog-Post-Overview-Layout -c Blog-Post -
t BlogTemplate -a
  cec create-contentlayout Blog-Post-Overview-Layout -c Blog-Post -r
  cec create-contentlayout Blog-Post-Overview-Layout -c Blog-Post -
r -s detail
```

3. Enter the following command to see what content types are available on your server:

   ```
   cec list-server-content-types
   ```

4. As shown in the "Usage", you can either create a content layout for the content type in the Oracle Content Management server or for the content type in the templates under `cec-components/src/main/templates`. For example, the following command creates the content layout for the type in the server:

   ```
   cec create-contentlayout Blog-Post-Overview-Layout -c Blog-Post -r
   ```

You can edit the following files to modify the content layout:

- `assets/layout.html`

  This file specifies the HTML view. See Edit the Content Layout in the Mustache Template.

- `assets/design.css`

  This file specifies the style for the content layout. See Define Styles in the design.css File.

- `assets/render.js`

  This file specifies the data used in `layout.html` and lets you add dynamic behavior to the content layout. See Define the RequireJS Module.

If you created a layout that allows for custom settings (using the option `-a` when creating the layout), an additional file named `settings.html` is also created which provides a default rendering of a single content item so it can be displayed.

Define the RequireJS Module

You can define a RequireJS module in the `render.js` file. Sites loads the dependencies, such as for JQuery, Mustache, the RequireJS Text Plugin, and the RequireJS CSS plugin.

```
define([
        'jquery',
        'mustache',
        'text!./layout.html',
        'css!./design.css'
], function ($, Mustache, templateHtml, css) {
```

You can use the Mustache template system to render the layout.

The `assets/render.js` file for a content layout has the following properties:

- It should be a RequireJS module
- It should return a JavaScript Constructor function. Sites invokes the Constructor function by passing a parameter object. The parameter object has the content item data and the APIs required to render the layout.
- This Constructor function should have a `render(parentObj)` method that handles rendering the content layout. It should append the content layout DOM object to the `parentObj` object that is passed to the `render()method`.
- The RequireJS module can use the dependencies, including JQuery, Mustache, the RequireJS Text Plugin, and the RequireJS CSS plugin. These dependencies will be loaded by sites. You can use other libraries too.

Configure the Constructor Function Parameter

When sites creates a new instance of the constructor function, it passes a parameter that contains `contentItemData`, `scsData`, and `contentClient` to help with content layout development.

Here is example code for the constructor function:

```
function ContentLayout(params) {
    this.contentItemData = params.contentItemData || {};
    this.scsData = params.scsData;
    this.contentClient = params.contentClient;
}
ContentLayout.prototype = {
    render: function (parentObj) {
        var content = {
            blogTitle: this.contentItemData.data['starter-blog-post_title'],
        };
```

```
        if (this.scsData) {
            content = $.extend(content, {
                'scsData': this.scsData
            });
        }

    }
};
return ContentLayout;
```

The constructor function parameter includes the following objects:

- **params.contentItemData:** Contains the content item, including its name, description, ID, and data. For example, the field '`blogpost_title`' in the content item can be accessed using `params.contentItemData.data['blogpost_title']`.
- **params.scsData:**:This object passes in information when the constructor is called from within sites. This object doesn't exist for content layouts rendered in third-party applications. This object contains a Sites SDK object, the method `contentTriggerFunction` to raise a trigger, and the **Details** page links.
- **params.contentClient:**This is the `contentClient` object created from the Content SDK and used to call the content layout. It is therefore configured with the appropriate parameters for the content server. If you need to make additional calls to the content server, you can use this `contentClient` object rather than creating your own. This object contains client APIs for the content. APIs are available to query, search, and get content items and their content types. Other helper APIs are also available; for example, `expandMacros()` to expand the macros used in rich text.

Render the Content Layout

The `render(params)` method of `ContentLayout` renders a content layout from a template. The Mustache template is used by default for content layouts, but you can use any template technology you want.

The `render(params)` method of `ContentLayout` can use the following code to render the template with the data:

```
try {
    // Mustache
    template = Mustache.render(templateHtml, content);
    if (template) {
        $(parentObj).append(template);
    }

    // Dynamic DOM Manipulation can be done here

} catch (e) {
    console.error(e.stack);
}
```

You can add the required data to the `content` object created from `params.contentItemData`. Oracle recommends that you merge the properties from

`params.scsData` into this object, so the template can make use of them too. The rendered template should be appended to the parent object passed to the `render()` method.

Edit the Content Layout in the Mustache Template

You can edit the default Mustache template in the `assets/layout.html` file, which contains the default content layout.

A simple template to render a blog title follows:

```
<h1>
  {{blogTitle}}
</h1>
```

Add Dynamic DOM Manipulation

You can add Dynamic DOM manipulation to `render.js` after `Mustache.render()` is called and the template is appended to the parent object.

For example, you could attach a listener, dynamically changing the style:

```
// Dynamic DOM Manipulation can be done here
$('h1').click(function (event) {
    alert('Title is : ' + $(this).text());
});
```

Define Styles in the design.css File

In the `design.css` file, you can define any style used in the content layout template.

The `design.css` file is loaded in the module definition using the RequireJS CSS plugin.

Get Reference Items

You can get a reference item for a content type with a reference data field that refers to another content type.

For example, the `Author` field in the Blog-Post content type is a reference to the Author content type. In the content layout for Blog-Post, `contentClient.getItems()` associates the details of the Author reference item with the current Blog-Post item .

```
var author_id = this.contentItemData.data['starter-blog-post_author'].id;
var ids = [author_id];
// Get the author reference item
contentClient.getItems({
                            'ids': ids
}).then(function (items) {
                // Use the item
 }, function (error) {
                 // Handle error
});
```

Get a Media URL

You can use `contentClient.getRenditionURL()` to get the default rendition of a digital asset, such as an image.

```
blogHeaderImage = contentClient.getRenditionURL({
                            'id': data['blog-post_header_image'].id
                            }),
```

If you need other renditions, such as `thumbnail`, you can get the digital asset using `contentClient.getItems()` and refer to `item.data.renditions.default` and `item.data.renditions.thumbnail`.

Raise Triggers

You can use `scsData.contentTriggerFunction(payload)` to raise a trigger from a content layout.

Here is an example of an Author content layout raising a trigger when an author name is clicked:

```
{{#data}}
<div class="author-container">
    <span class="author-
name" onclick='{{scsData.contentTriggerFunction}}("field:starter-blog-
post_author:equals={{author_id}}")'>{{starter-blog-author_name}}</span>
{{/data}}
```

The payload is a search query for the currently selected author, which other Content List items on the page can listen to.

Navigate to a Search Page with a Search Query

A common use case is to navigate to a search page with a dynamic search query when clicking on a link inside a content layout.

For example, assume that you want to navigate to a search page named "Authors" when clicking on the "More articles from this author" link in your content layout, passing a search payload. The following code will achieve this. Notice that the global objects `SCS` and `SCSRenderAPI` are available to use in the content layout when running inside a sites page.

```
$('.more-from-author').click($.proxy(function () {
    var childrenPages = SCS.structureMap[SCS.navigationRoot].children;

    if (!childrenPages) return; // No pages

    // Find the Authors page
    for (var i = 0; i < childrenPages.length; i++) {
        var page = SCS.structureMap[childrenPages[i]];
        if (page.name === 'Authors') {
            var linkData = SCSRenderAPI.getPageLinkData(page.id);
```

```
            if (linkData && linkData.href) {
                var href = linkData.href,
                    searchPayload = content.author_id + '*',
                    contentType = "Starter-Blog-Post";
                // if both the page URL and the search query exists,
navigate to the page passing in the query
                if (href && searchPayload) {
                    var queryStart = href.indexOf('?') === -1 ? '?' : '&';

                    // add in the contentType and search parameters
                    // contentType isn't a required URL parameter
                    // Payload contains search string only. No parameter
name.
                    href += queryStart + (contentType ? 'contentType=' +
contentType + '&' : '') + 'q=' + searchPayload;

                    // navigate to the search results page
                    window.location = href;
                 }
            }
        }
    }
}, this));
```

If you expect the same content layout to be used multiple times in the same page, it is better to use the unique ID in the CSS selector rather than the class selector, like `$('.more-from-author').click(…)`.

For example:

```
template.html
    <div id="{{navigateId}}">….</div>

render.js
    content.navigateId = this.scsData.id + 'detailTrigger';
    $('#' + navigateId).click(…)
```

Expand Macros and Render Rich Text

Rich text in a content item can embed a digital image.

To render this rich text correctly in the content layout, rich text fields use `contentClient.expandMacros()` API. This resolves all the references to digital assets inside the rich text.

```
data["starter-blog-post_content"] =
 contentClient.expandMacros(data["starter-blog-post_content"]);
```

If you use Mustache for rendering, `{{{ }}}` should be used to render a rich text value because rich text has HTML. When `{{{ }}}` is used around the variable, Mustache doesn't escape the HTML.

Link to the Details Page

The Details page link is available through `scsData.detailPageLink` .

For example, if you want to navigate to the Details page to display blog details when you click a blog title, you can use the Details page link as follows:

```
<a href="{{scsData.detailPageLink}}">
        <h2 class="post-title">
                {{blogTitle}}
        </h2>
</a>
```

Expand Macros in Content List Queries

In a content list query, you can define values for properties that are calculated when a page is run, to display content that has been recently updated.

Most properties for components within sites are static. The user selects or enters a fixed string or value for one of the properties of the component, and that doesn't change regardless of when or where the page is run. However, you can define values for properties that are calculated when the page is run. This is useful for displaying content that has been recently updated in content queries. Users can enter dates such as "in the last 3 days".

You can insert a Mustache JS expansion to several properties. The values referenced in these strings are derived from a model that is executed when the page is run. An out-of-the-box model handles dates formatted for Content REST API calls. You can extend this model with additional values to meet any user requirements.

An example of the string you can enter for a property follows:

```
Content List component:
      Additional Query String property:
             updatedDate gt "{{#content.date}}today - 3 days{{/
content.date}}"
```

This Mustache entry for the date will be evaluated at runtime so that the returned value changes depending on when it is run (that is, expands to `updatedDate gt "2220181002060000000"`). In this way, the user can build up any complex date string rather than having to enter a predefined value.

Supported Component Properties

The following properties support Mustache JS template syntax:

- Content List
  - Additional Query String
  - For example: `updatedDate gt "{{#content.date}}today - 3 days{{/content.date}}"`
- Title/Paragraph/Text
  - Rich text entered via a rich text editor

  - For example: `"Content REST API format for date: {{#content.date}}now{{/content.date}}"`

> **Note:**
>
> Without a custom model for the Mustache template, the expansion in Title/Paragraph/Text isn't that useful. However, it is very useful for validating what you enter in the Additional Query String because it will be evaluated as you switch between edit and view and be immediately visible.

Supported Component Syntax

The `content.date` object is supported out-of-the-box. This takes in two main parameters, `today` and `now`.

The `today` value takes the current browser time, converts it to midnight tonight, and then converts that value to UTC time.

- `{{#content.date}}today{{/content.date}}` expands to the browser value for midnight tonight, converted to the UTC value and formatted into the Content REST API date format. For example:

  ```
  2220181008065959999
  ```

- It can then be augmented with:

  ```
  today +/-  [day | week | month | year]
  ```

- The `today` value also behaves differently as you add or subtract from it. If you subtract from it, it will use the time in the morning. If you add to it, it will use the time at midnight. For example:
  - `{{#content.date}}today - 1 day{{/content.date}}` expands to yesterday at start of day.
  - `{{#content.date}}today + 2 days{{/content.date}}` expands to the day after tomorrow at midnight.

The `now` value takes the current browser time and converts it to UTC time without any adjustment.

- `{{#content.date}}now{{/content.date}}` expands to the current browser time converted to the UTC value and formatted into the Content REST API date format.
- `now` can also be augmented with `hour`. So you have:

  ```
  now +/-  [hour | day | week | month | year]
  ```

- For example:
  - `{{#content.date}}now + 2 hours{{/content.date}}` - two hours from now converted to UTC time formatted in the Content REST API date format
  - `{{#content.date}}now - 1 day{{/content.date}}` - yesterday at this browser time converted to UTC time formatted to Content REST API date format

Using the Supported Component Syntax

To use the macro expansion in the additional query string, suppose you wanted to return everything in the last 3 weeks, you would enter the following:

```
updatedDate gt "{{#code.date}}today - 3 weeks{{/code.date}}"
```

Only the date is returned, so to work on the Content REST API call, quotes are added when you construct the query string as you would when entering a static value.

MustacheJS

For syntax, reference the Mustache JS template pages at mustache.github.io/mustache.5.html.

One change has been made to the Mustache instance that is run when expanding strings. Mustache provides both a `text` expansion that uses `{{ }}` and an `html` expansion that uses `{{{ }}}`. The difference between these two is that the text expansion does an HTML encode on the string; that is, if the value expanded to `a < b`, then the result would be `a < b`. This is not what you would want to construct strings for URLs. You could tell the user to use the HTML expansion, but that is just an overhead and will generate more issues, like explaining why they need to use `{{{ }}}`.

To avoid this, Mustache has been set up so that it doesn't escape values when using `{{ }}`. This means that both `{{ }}` and `{{{ }}}` behave the same. This also leaves the encoding of any result as an exercise for the user if it's required.

The OOTB Mustache Model

Mustache requires a model to be applied to the template for expansion. In the preceding example, `{{#content.date}}` is already defined out-of-the-box, whereas a new property, such as `{{person}}`, can be added by the developer. If the user enters a value in the Mustache template that isn't in the model, the result will be an empty string. So, in the case of `Hello {{person}}`, it would expand to just `Hello` unless the developer adds `person` to the model.

The model object used is a global object called `SCSMacros`. The developer is free to add any additional entries into this object. The object will be passed to Mustache when the template is evaluated.

The out-of-the-box model object currently supports only the `content.date` object:

```
{
   content: {
     date: <lambda implementation>
   }
}
```

Custom Mustache Model

The supported objects can be enhanced by the developer based on their requirements. So they can introduce a `lastTwoDays` object and simplify the expansion to just `{{lastTwoDays}}`.

To extend the model to support something like `Hello {{person}}` in the preceding example, you would need to add the `person` object to the SCSMacros. You need to do this before the page is run. It can be done within the page layout by adding a script tag to the start of it. For example:

```
<script type="text/javascript">
window.SCSMacros = window.SCSMacros || {};  // define/get the SCSMacros
object
window.SCSMacros.person = "World";
</script>
```

After this change is made, the `Hello {{person}}` template would expand to: `Hello World`.

If you want to pass values to the object (for example, `Hello {{#person}}personId{{/person}}`), then you need to implement a `mustache lambda` and wrap and expand the value within the implementation.

For example:

```
<script type="text/javascript">
window.SCSMacros = window.SCSMacros || {};  // define/get the SCSMacros
object
//implement "person" as a lambda
window.SCSMacros.person = function () {
  var people = { '111': { firstName: 'Small', lastName: 'World'}, '222':
{ firstName: 'Big', lastName: 'Universe'} };
  return function (text, render) {
     var expandedText = render(text);
     var chosenPerson = people[expandedText] || people['111'];
     return chosenPerson.firstName;
  }
};
</script>
```

After this change is made, the `Hello {{#person}}111{{/person}}` template would expand to `Hello Small`, and the `Hello {{#person}}222{{/person}}` template would expand to `Hello Big`.

> **Note:**
>
> Mustache expansion executes synchronously. If you need to retrieve asynchronous values, these will need to be resolved within the model before you attempt to execute the mustache expansion, and this isn't currently supported, although bespoke implementations are possible.

Develop Robust Content Layouts

Content layouts need to be robust to the three types of response data that you get from content REST calls:

- Content item: with the `expand=all` parameter will have references expanded and large text fields.
- Content item: without `expand=all` will not have references expanded but will have large text fields.
- Content queries will not have references expanded or large text fields.

To improve performance, the content REST call that retrieves asset data no longer includes the `expand=all` parameter, as of the 19.2.3 release of Oracle Content Management. The `expand` parameter tells Oracle Content Management to drill down and retrieve all the referenced items as well as the current item in the response. Custom content layouts that rely on data retrieved through the `expand` parameter need to be updated to handle cases where the data retrieved does not contain the referenced field values.

Render Content Items

Content Layouts are used to render content items. They receive the content item data, render it into HTML, and insert it onto the page.

By default, content layouts leverage Mustache templating to render content items, although they can be implemented in any JavaScript technology. For the Mustache template to render, it expects the data to be in a certain format. The content layout `render.js` file needs to ensure that the model it passes to the template matches that format.

Content layouts are rendered in several use cases:

- When used in the Oracle Content Management Asset Management UI, the data can be in an "edited" state for the user to preview changes before saving them.
- When used in an Oracle Content Management site in a content list or content item, the data is augmented with additional information about the site in which it is running.
- When used through the Content SDK's `contentClient.renderLayout()` call, where the user of the Content SDK passes whatever data they want directly to the content layout.

For performance, there is a general trade-off between creating a single query that can return all required data or multiple queries so that the outline renders as fast as possible, with a fast initial query, and areas are subsequently filled in through subsequent queries. Which model you choose depends on your data and use cases.

Also, the data passed to a content layout can vary in format due to how the data was retrieved. For example, if you use a content REST with an expand parameter, field references to other content items, either individually or as a group, can also be returned. If the referenced content items aren't included, you'll need to make additional REST calls.

To handle all cases, the content layout developer should attempt to be flexible about the format of received data. Also, when necessary, the developer can fetch additional data and coerce the data into the format expected by the rendering template.

Standardize the Structure of Data for a Content Layout

The content layout developer needs to standardize the structure of data that the content layout receives.

If all the data is present, content layout can simply render the component. If the data isn't all present, the content layout might need to make additional queries. In all cases, the content layout should never assume a certain data format and instead coerce the data into a format that will render.

You will need to ensure that you have all the data you're expecting. If the data doesn't exist, you'll need to make the additional queries. The following fields will potentially be missing from the data:

- The "`fields`" entry for referenced fields
- Large text fields

Because content layouts are designed for specific content types, the developer of a content layout knows the list of fields needed. For each of these fields, the data needs to be fetched so the content layout can render. You have two options: fetch missing data and then render with complete data, or render immediately and then fetch missing data to fill in the blanks.

Option 1: Fetch Missing Data and Then Render with Complete Data

Create a Promise to retrieve the required data and then continue rendering when all Promises return.

For example, we have the following content types with corresponding fields:

- `starter-blog-author`
  - fields
    - `starter-blog-author_name` -text field
    - `starter-blog-author_bio` - text field
- `starter-blog-post`
  - fields
    - `starter-blog-post_title` - text field
    - `starter-blog-post_content` - large text field
    - `starter-blog-post_author` - reference to a `starter-blog-author` item

The Content Layout has the following template, to render these expected field values:

```
{{#fields}}
<div class="blog_container">
    <div class="blog-post-title">{{starter-blog-post_title}}</div>
    {{#starter-blog-post_author.fields}}
    <div class="blog-author-container">
        <div class="blog-author-details">
            <div class="blog-author-name">{{starter-blog-author_name}}</div>
            <div class="blog-author-bio">{{{starter-blog-author_bio}}}</div>
            <span class="more-from-author">More articles from this author</
span>
        </div>
    </div>
    {{/starter-blog-post_author.fields}}
    <div class="blog-post-content">{{{starter-blog-post_content}}}</div>
</div>
{{/fields}}
```

The Content Layout can be called with data from the following queries:

- Item query with "expand" - all data supplied
  - /content/published/api/v1.1/items/{id}?expand=fields.starter-blog-post_author&channelToken=8dd714be0096ffaf0f7eb08f4ce5630f
  - This is the format of the data that is required to successfully populate all the values in the template. If either of the other queries is used, additional work is required to fetch the data and convert it into this format.
  - ```
    "fields": {
        "starter-blog-post_title": "...",
        "starter-blog-post_summary": "...",
        "starter-blog-post_content": "...",
        "starter-blog-post_author": {
            "id": "CORE386C8733274240D0AB477C62271C2A02",
            "type": "Starter-Blog-Author"
            "fields": {
                "starter-blog-author_bio": "...",
                "starter-blog-author_name": "..."
            }
        }
    }
    ```
- Item query, without "expand" - missing referenced item fields **"starter-blog-post_author.fields"**:
  - /content/published/api/v1.1/items/{id}?channelToken=8dd714be0096ffaf0f7eb08f4ce5630f
  - ```
    "fields": {
        "starter-blog-post_title": "...",
        "starter-blog-post_summary": "...",
        "starter-blog-post_content": "...",
        "starter-blog-post_author": {
            "id": "CORE386C8733274240D0AB477C62271C2A02",
            "type": "Starter-Blog-Author"
        }
    }
    ```
- SCIM query - missing large text field **"starter-blog-post_content"**, missing referenced item fields **"starter-blog-post_author.fields"**:
  - /content/published/api/v1.1/items?q=(type eq "Starter-Blog-Post")&fields=ALL&channelToken=8dd714be0096ffaf0f7eb08f4ce5630f
  - ```
    "fields": {
        "starter-blog-post_title": "...",
        "starter-blog-post_summary": "...",
        "starter-blog-post_author": {
            "id": "CORE386C8733274240D0AB477C62271C2A02",
            "type": "Starter-Blog-Author"
        }
    }
    ```
    ```

To be able to consistently render with any of these queries, the `render.js` from the content layout needs to make sure all the referenced fields are expanded and that the large text fields are present.

If this is not the case,it needs to query these back, fix up the data, and then render with the complete data.

Sample `render()` function:

```
render: function (parentObj) {
    var self = this,
        template,
        contentClient = self.contentClient,
        content = self.contentItemData;

     var getRefItems = function (contentClient, ids) {
        // Calling getItems() with no "ids" returns all items.
        // If no items are requested, just return a resolved Promise.
        if (ids.length === 0) {
            return Promise.resolve({});
        } else {
            return contentClient.getItems({
                "ids": ids
            });
        }
     };

     var fetchIDs = [], // list of items to fetch
         referedFields = ['starter-blog-post_author'], // names of reference
fields
         largeTextFields = ['starter-blog-post_content'], // large text
fields in this asset
         fieldsData = content.fields;
     // See if we need to fetch any referenced fields
     referedFields.forEach(function (fieldName) {
         if(fieldsData[fieldName] && fieldsData[fieldName].fields) {
            // got data already, nothing else to do
         } else {
             // fetch this item
             fetchIDs.push(fieldsData[fieldName].id);
         }
     });

     // See if we need to fetch any large text fields
     for(var i = 0; i < largeTextFields.length; i++) {
        if(!fieldsData[largeTextFields[i]]) {
           // need to fetch this content item directly to get all the large
text fields
            fetchIDs.push(content.id);
            break;
        }
     }
    // now we have the IDs of all the content items we need to fetch, get
them all before continuing
    getRefItems(contentClient, fetchIDs).then(function (referenceData) {
```

```
        var items = referenceData && referenceData.items || [];

        // add the data back in
        items.forEach(function (referencedItem){
            // check if it's the current item
            if(referencedItem.id === content.id) {
               // copy across the large text fields
               largeTextFields.forEach(function (fieldName) {
                   fieldsData[fieldName] =
referencedItem.fields[fieldName];
                });
            } else{
                // check for any referenced fields
                for (var i = 0; i < referedFields.length; i++) {
                    if(referencedItem.id ===
fieldsData[referedFields[i]].id){
                       // copy across the fields values
                       fieldsData[referedFields[i]].fields =
referencedItem.fields;
                       break;
                    }
                }
            }
        });

        // now data is fixed up, we can continue as before
        try{
           // Mustache
           template = Mustache.render(templateHtml, content);

             if(template) {
                $(parentObj).append(template);
             }

        } catch (e) {
            console.error(e.stack);
        }
    });
}
```

Option 2: Render Immediately and Then Fetch Missing Data to Fill in the Blanks

Performance can be improved by separating out the items that might not be present and rendering them in a second pass. This will require two Mustache templates, the first to do the initial render, leaving "holes" that are then filled in with the second render once the data is complete.

This requires setting up the Mustache template to support multiple passes either by having separate templates for the "holes" or having the model return template macros rather than actual values. In either case, you'll need to "hide" these holes until the data is retrieved and then populate them and show them with appropriate UI animation to avoid the page "jumping about" too much.

Create the Sample Blog Template

The BlogTemplate sample demonstrates content layout features.

You can create the template in Developer Cloud Service, examine the content layouts in the template, and test the search capabilities:

1. Create a template of the type `BlogTemplate`:

   ```
   cec create-template MyBlogTemplate -f BlogTemplate
   ```

   A new template, `MyBlogTemplate`, will be created by copying `BlogTemplate`.
2. Open http://localhost:8085/, and then click **Templates**, then **MyBlogTemplate**.
3. You see a list of Blog posts. Click on one of them. It takes you to the Details page, which uses three different Content Layouts to render.
4. Click **More articles from this author**. It takes you to a search page.
5. Click an author name. That content layout raises a trigger and shows the articles on the right side.
6. Go to the Home page again and test the search.

Add Content Layout Mappings to Templates

After you create the content layout, you can add it to local templates in your Developer Cloud Service project by adding a content layout mapping.

Use the `cec add-contentlayout-mapping` command to add a content layout mapping for a template. The content type that the content layout is based on (`-c`) and the template that the mapping is for (`-t`) are required. For example:

```
cec add-contentlayout-mapping Blog-Post-Detail-Layout -c Blog-Post -t
BlogTemplate
```

The default content layout mapping is the Default style for desktop. You can specify the `-s <layoutstyle>` option to specify a different layout style, such as Overview or Details, by name:

```
cec add-contentlayout-mapping Blog-Post-Detail-Layout -c Blog-Post -t
BlogTemplate -s Details
```

You can also set the mapping for mobile with the `-m` option:

```
cec add-contentlayout-mapping Blog-Post-Detail-Layout -c Blog-Post -t
BlogTemplate -m
```

Test Content Layouts with the Local Test Harness

After you add your content types and content layout mappings to a template, you can test your content layouts in the local test harness.

See Test with a local test harness.

Test with a Local Test Harness

Run your custom components, templates, and content layouts in a local test harness before importing them to Oracle Content Management.

To start the local test harness:

1. Enter `cd cec` in a terminal window.
2. Enter `cec develop &` or `cec develop --server <server-name> &`
3. Open a browser at `http://localhost:8085` to see your components, templates, and content layouts running in the local test harness.
4. You can find your components, templates, themes, and so on, in these directories:
   - `cec/src/components`
   - `cec/src/templates`
   - `cec/src/themes`

Import Templates with Content Layouts into Oracle Content Management

After you develop and test your content layouts, you can export the template that contains the content layouts from your Developer Cloud Service project and then import the template into Oracle Content Management.

See Export a Template.

Prepare Content Layouts and Site Pages to Use Consumption Analytics

Consumption analytics lets you track the usage and popularity of assets in sites. This is done automatically at the site level for all standard components, provided the asset consumption feature is enabled for the site.

To track consumption analytics for custom layouts and components, you must prepare content layouts and site pages to collect data within the site or sites where a component is used. The analytic information can then be collected when the site page renders and called in APIs for display in Oracle Content Management.

Several events can be collected for assets used on site pages.

Event Type	Description
load	The asset is referenced on the site page.
view	The asset on the site page is scrolled into the browser viewport.
play	A media asset on the site page has started to play.
download	A download operation is initiated on the asset.

Preparing Pages to Collect Data

There are two ways to prepare content layouts and site pages to collect data. You can augment the rendered asset markup on the page, or manually produce asset events with direct JavaScript calls.

Component Attributes

To automatically record asset consumption events, you can add special data attributes to the rendered asset markup on the page. The `data-asset-operation` attribute instructs the SCSRenderAPI to automatically produce analytic events. The value syntax for the `data-asset-operation` attribute is *<page-event>:<asset-id>:<asset-event>*. Multiple asset operations can be specified per attribute, separated by a semicolon (;).

There are several parameters for the `data-asset-operation` attribute.

Parameter Name	Description
page-event	A page-event normally corresponds to a DOM event, like *click* and *play* events on elements. The supported page events include: • view—the element has scrolled into the browser viewport • play—a <video> element has started to play This corresponds to the DOM *play* event on <video> elements. • click—the element has been clicked This corresponds to the DOM *click* event.
asset-id	The id of the content item or digital asset.
asset-event	One of the supported asset events • load • view • play • download

For example, the final markup of a simple content item might look like this:

```
<div data-asset-operation="view:COREBE60D5159507409B97E9B5CD27937B82:view">
    Hello World!
</div>
```

The `data-asset-operation` attribute in this markup automatically generates a load event for the asset when this markup appears on the page. Additionally, a view event will be recorded when the item is scrolled into view in the browser viewport.

JavaScript API Calls

JavaScript can also be used to record asset events. The SCSRenderAPI object exposes the following function to record asset events:

```
SCSRenderAPI.recordAssetOperation( assetId, assetEvent ) → {Boolean}
```

Name	Type	Required	Description
assetId	String	Yes	The id of the content item or digital asset.

Name	Type	Required	Description
assetEvent	String	Yes	One of the supported asset events • load • view • play • download

This object returns a Boolean value of **true** if the event was accepted for recording, or **false** if not accepted.

For example, the following script records a play event when the button with the specified addetId is clicked.

```
<button
onclick="javascript:SCSRenderAPI.recordAssetOperation('CONTBEAA53457DDE
412B872D21DDC05FED5D', 'play')">Play the Associated Video</button>
```

Once you've prepared your content layout and site pages, you will need to republish your site.

> **Note:**
>
> Consumption analytics must be enabled in site settings before analytic information can be recorded.

Additional Examples

The following is an example of adding `data-asset-operation` markup for digital assets.

```
<img data-asset-
operation="view:CONTBE5A53457DAE412B872C21DDC05FED5D:view" src="https://
samples.mycontentdemo.com/content/published/api/v1.1/assets/
CONTBE5A53457DAE412B872C21DDC05FED5D/Medium/Blog400px.jpg?
channelToken=47c9fb78774d4485bc7090bf7b955632">
```

The following is an example of adding `data-asset-operation` markup for referenced field types.

```
<ul data-asset-
operation="view:COREBE60D5159507409B97E9B5CD27937B82:view">

<li>Name: Joe Bloggs</li>

<li>Age: 39</li>

<li>Photo: <img data-asset-
operation="view:CONTBE5A53457DAE412B872C21DDC05FED5D:view"
src="https://samples.mycontentdemo.com/content/published/api/v1.1/
assets/CONTBE5A53457DAE412B872C21DDC05FED5D/Medium/Blog400px.jpg?
```

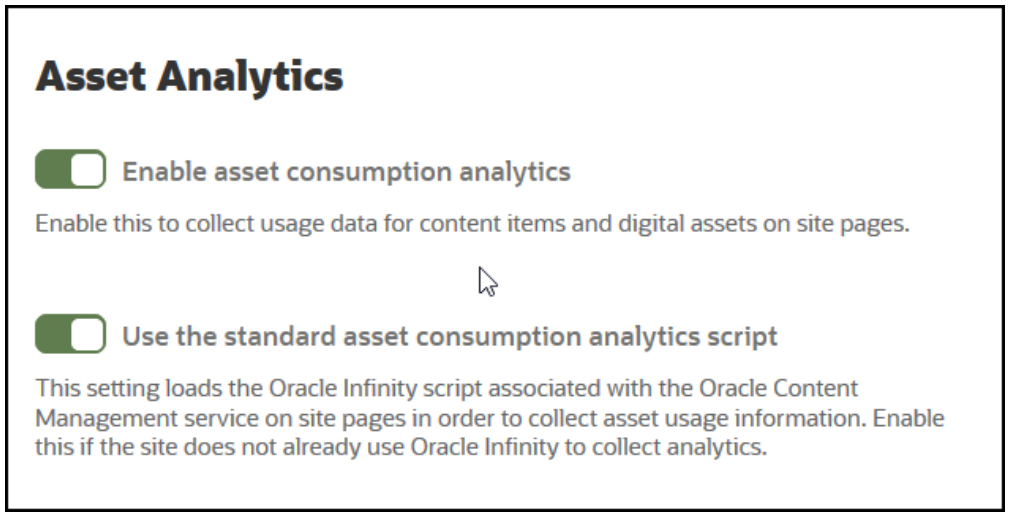
```
channelToken=47c9fb78774d4485bc7090bf7b955632"></li>

<li>Video: <video data-asset-
operation="view:CONTBE5A53457DAE412B872C21DDC05FED5D:view;play:CONTBE5A53457D
AE412B872C21DDC05FED5D:play" src="https://samples.mycontentdemo.com/content/
published/api/v1.1/assets/CONTBE5A53457DAE412B872C21DDC05FED5D/Medium/
Blog400px.jpg?channelToken=47c9fb78774d4485bc7090bf7b955632"></video></li>

</ul>
```

Enable Site-Level Consumption Analytics

Oracle Content Management has the ability to collect analytic information for provided components used in your site out of the box, and any custom content layouts and components you have prepared to provide analytic information. To use this feature, you must enable asset analytics in the analytics section of your site settings in Site Builder.



1. Open the site you want to collect consumption data for and click **Edit**.
2. Select an update to edit or create a new one.
3. Click  to open the setting panel and click **Analytics**.
4. Under**Asset Analytics**, toggle **Enable asset consumption analytics** on.
5. If the site does not already use Oracle Infinity to collect analytics, also toggle **Use the standard asset consumption analytics script** on to load the Oracle Infinity script to collect asset usage data at the site level.
6. Republish your site.

25

Develop Components

You can develop components for use in Oracle Content Management sites.

About Components

A component is a specific type of content that you can add to a page in a site. Components in Oracle Content Management include items such as paragraph, title, image, divider, and so on.

Oracle Content Management supports these types of components:

- Local component: files are stored in Oracle Content Management
- Remote component: files are stored on a remote server

Local components can be set to render directly within the page or render in an inline frame in the page. Remote components are always rendered in an inline frame.

Oracle Content Management provides a default set of components with each template (which also includes themes and sites). You can create new components and also use these default components within your components. Once you've added a component to your site, you can edit the component's property settings to meet your requirements by specifying page content,

fonts and font sizes, image framing and placement, and other styles. What settings you can edit depends on the component type.

These components are included with Oracle Content Management.

Name	Type	ID
Title	scs-title	scs-title
Paragraph	scs-paragragh	scs-paragragh
Text	scs-title	scs-text
Image	scs-image	scs-image
Gallery	scs-gallery	scs-gallery
Gallery Grid	scs-gallerygrid	scs-gallerygrid
Document	scs-document	scs-document
Button	scs-button	scs-button
Map	scs-map	scs-map
Divider	scs-divider	scs-divider
Spacer	scs-spacer	scs-spacer
YouTube	scs-youtube	scs-youtube
Social Bar	scs-socialbar	scs-socialbar
Video	scs-video	scs-video
Dynamic List	scs-dynamiclist	scs-dynamiclist
Recommendation	scs-recommendation	scs-recommendation
Article (custom component)	scs-component	scs-comp-article
Headline (custom component)	scs-component	scs-comp-headline
Image and Text (custom component)	scs-component	scs-comp-image-text
Conversation	scs-app	Conversation
Documents Manager	scs-app	Documents Manager
Project Library	scs-app	Project Library
Conversation List	scs-app	Conversation List
Start Form	scs-app	Start Form
Task List	scs-app	Task List
Task Details	scs-app	Task Details
Folder List	scs-app	Folder List
File List	scs-app	File List
Facebook Like	scs-app	Facebook Like
Facebook Recommend	scs-app	Facebook Recommend
Twitter Follow	scs-app	Twitter Follow
Twitter Share	scs-app	Twitter Share

See Arrange Page Content in *Building Sites with Oracle Content Management*.

About Developing Components

Developing your own custom component enables you to develop compound elements that can be embedded within the site page, using any page technology of choice. This effectively enables you to extend the list of components provided with Oracle Content Management.

Entries for all registered components are stored in the Component Catalog, which is a folder in Oracle Content Management that contains the entries for all registered components.

Don’t use the following names for site templates, themes, components, sites, or site pages: authsite, content, pages, scstemplate_*, _comps, _components, _compsdelivery, _idcservice , _sitescloud, _sitesclouddelivery, _themes, _themesdelivery. Although you can use the following names for site pages, don’t use them for site templates, themes, components, or sites: documents, sites.

Component Types

The Component Catalog supports these types of components:

- Local component
- Local component rendered in an inline frame
- Remote component

The type of component is stored as an extension attribute, `"xScsAppType"`, of the component folder. Valid values follow.

Type	Description
Local component	All dependencies are expected to be local.
Local component rendered in an inline frame	Component is served from the same domain as Oracle Content Management.
Remote component	Component is served from remote location.

Component File Structure

Each of the types of components has a different set of files when created in the Component Catalog, based on how they are implemented.

Local component:

```
/Components/component-name
    appinfo.json
    _folder_icon.jpg
    assets
        settings.html
        render.mjs
```

Local component using inline frame:

```
/Components/component-name
    appinfo.json
    _folder_icon.jpg
    assets
```

```
        settings.html
        render.js
        js
            sites.min.js
            knockout.min.js
            jquery.min.js
```

Remote component:

```
/Components/component-name
    appinfo.json
    _folder_icon.jpg
    keys.json
```

Folder Metadata

The component registration data is stored in the folder metadata. The following properties are used to uniquely identify the component and its type.

Property	Description
app name	Name of the folder which designates the name of the component.
app description	Description of the folder.
app guid	Every component is associated with a GUID and is stored as an extension attribute `xScsItemGUID`. The GUID is generated by the server when the component is created.
app type	Property designates the component type. It is stored as an extension attribute `xScsApType`.

> **Note:**
>
> The property `iconUrl`, which is stored as extension attribute `xScsAppIconUrl`, has been deprecated.

appinfo.json File

The `appinfo.json` registration file for each type of component contains only data that is not available in the folder metadata. Component properties that are defined in the folder metadata are not duplicated over to the `appinfo.json` file. Note that the appInfo.json file differs from the components.json file, which is used for themed components.

Local component:

```
{
        "settingsData":{
                "settingsHeight":80,
                "settingsRenderOption"; "dialog",
                "settingsWidth":300,
                "componentLayouts":[],
```

```
                "triggers":[],
                "actions":[]
        },
        "initialData":{
                "customSettingsData":[],
        }
}
```

Local component rendered in an inline frame:

```
{
        "endpoints": {
              "settings": {
                      "height": "300",
                      "width": "400"
              }
        }
        "initialData": {
                "customSettingsData": {}
        }
}
```

Remote component:

```
{
        "endpoints": {
              "widget": {
                     "url": "http://www.externaldomain.com/app/render.html"
              }
              "settings": {
                      "url": "http://www.externaldomain.com/app/
settings.html",
                      "height": "300",
                      "width": "400"
              }
        }
        "initialData": {
              "customSettingsData": {}
        }
}
```

There are no registered `"url"` values for local components. They use certain files which can be edited for content, but you can't change the location or name.

- Local components use the files `assets/render.mjs` and `assets/settings.html`.
- Local components rendered in an inline frame use the files `assets/render.html` and `assets/settings.html`.
- Remote components use whatever `"url"` values are specified.

Create a Component

You can create customized components for use in Oracle Content Management.

While Oracle Content Management provides many predefined components for use in building a website, as a developer, you can create custom components with access to the same features and capabilities. These component types include local, local rendered in an inline frame, and remote rendered in an inline frame.

Local Component

When you create a local component, you are given a fully functional sample component that you use as a basis for creating your own component. Select **Knockout** to create a Knockout version of the component, or select one of the template versions: **Mustache**, **Preact**, or **React**. See Create Local Components or Layouts in *Building Sites with Oracle Content Management*.

Local Component with an Inline Frame

When you create a local component with an inline frame, you are given a fully functional sample component that you use as a basis for creating your own component. You'll use the same instructions as for creating a local component, and also select the **Knockout (Sandboxed)** option to create an inline frame version of the component that is stored locally. See Components Rendered in Inline Frames.

Remote Component

When you create a remote component, which uses an inline frame, select the **Knockout (Sandboxed)** option.

Copy the created files to your remote server, and register the remote component. Test the component before including it in your published site.

See Components Rendered in Inline Frames and Register a Remote Component.

Develop Custom Components with Developer Cloud Service

Oracle Developer Cloud Service helps you develop templates, themes, and custom components for Oracle Content Management.

The Developer Cloud Service integration with Oracle Content Management provides a template with tools to develop templates and components. It also provides sample unit tests to start with. The integration includes a Git repository and tools, which help develop templates and components, as well as a local test harness for quick, iterative development of templates, themes, and custom components.

Developer Cloud Service can help you do the following tasks:

- Set up your local development environment to use an Oracle Content Management instance for local development and testing of templates, themes, and components
- Create templates and components from samples or starters, run in them in the test harness, explore them, and develop the templates, themes and components in a Developer Cloud Service environment

- Import templates or components that were created from Oracle Content Management into a Developer Cloud Service environment for source management and further development
- Export templates or components from a Developer Cloud Service environment to be imported into Oracle Content Management for use in websites
- Copy an existing component
- Write unit tests
- Optimize components
- Deploy your components to Oracle Content Management

The following topics describe how to use the integration of Developer Cloud Service to develop for Oracle Content Management:

- Develop a Custom Component for Oracle Content Management
- Optimize Components (Minify) for Better Performance
- Run Continuous Integration Jobs

Develop a Custom Component for Oracle Content Management

Use Developer Cloud Service and your local machine to develop a custom component for Oracle Content Management.

The following topics describe the steps for developing and testing a Oracle Content Management component with Developer Cloud Service:

1. Set Up Oracle Content Management Toolkit on Your Local Machine.
2. Sign in to the Developer Cloud Service Console for Oracle Content Management.
3. Create a Project in Developer Cloud Service.
4. Add Oracle Content Management Toolkit to the Project Code in the New Git Repository.
5. Develop Your Custom Component
6. Test with a Local Test Harness
7. Write and Run Unit Tests

Develop Your Custom Component

Use the cec command-line utility to create a new component, develop the component locally, and then export the component to Oracle Content Management.

Create a Component

To create and develop a custom component locally, use the following command:

```
cec create-component <component-name> -f <source>
```

Choose one of the following values for *source*:

- `JET-CCA-Demo-Card`
- `local`

- `local-iframe`
- `Sample-Facebook-Share`
- `Sample-News-API`
- `Sample-Stocks-Embedded`
- `Sample-To-Do`
- `Sample-Text-With-Image`
- `Sample-Weather-Embedded`
- `Sample-Folder-List`
- `Sample-File-List`
- `Sample-Documents-Manager`
- `Sample-Process-Start-Form`
- `Sample-Process-Task-List`
- `Sample-Process-Task-Details`
- `SimpleHTML`

Example:

```
cec create-component MyLocalComponent1 -f local
```

The component is created in your Git repository under `cec-components/src/main/components`.

The `src/main/components` directory is seeded with the `Sample-To-Do` component. All components that you create go in this directory.

Copy a Component

You can copy a component in Developer Cloud Service with the `cec copy-component` command:

```
cec copy-component <source> [<destination>]
```

This command copies an existing component named <*source*> to <*destination*>.

Export or Deploy the Component to Oracle Content Management

Once the component is developed and tested in the local server, you can export the component using the following command. This creates the component zip file. You can manually import this component zip into Oracle Content Management.

```
cec export-component <component name>
```

Alternatively, you can deploy the component directly to Oracle Content Management from Developer Cloud Service using the following command.

```
cec deploy <component name>
```

The `cec deployAll` command will deploy all the components in `src/main/components`.

Import Components into Developer Cloud Service

If you have a component ZIP file created from the Oracle Content Management server, you can import that into Developer Cloud Service for further development. Use the following command:

```
cec import-component <location of the component zip file>
```

> **Important:**
>
> The source code for your components exists in `src/main/components`. You shouldn't modify any files outside `src/main/components` because they are needed for the functioning of the Oracle Content Management local server.

Write and Run Unit Tests

Start with the sample unit test to write unit tests and run them for your custom Oracle Content Management components.

Start with the Sample Unit Test

The Developer Cloud Service samples for Oracle Content Management include the following files to help you write unit tests:

- **src/test/unit**: Contains the unit test for the `Sample-To-Do` component. This serves as a sample for writing unit tests for component JavaScript code, which includes RequireJS modules. The Mocha and Chai frameworks for JavaScript unit testing are used.
- **index.html**: Runs the unit test. It loads Mocha, Chai, and the main test file, `test-main.js`.
- **test-main.js**: Loads the unit test module, `Sample-To-Do-Test`, and runs the Mocha unit tests.
- **Sample-To-Do-Test.js**:
  - Defines the actual tests. It loads the component source code, `components/Sample-To-Do/assets/render.js`, using RequireJS.
  - Tests the `add()`, `delete()`, `title()`, and `placeholder()` methods.

Write Unit Tests for Your Components

To write each of your unit tests for a custom component, follow these steps:

1. Write a unit test similar to `Sample-To-Do-Test.js`.
2. Load the test you wrote into `test-main.js`.

Run a Unit Test Locally

To run your unit test locally, follow these steps:

1. Clone the Git repository locally.

2. In a terminal window on your local machine, enter `npm install`.
3. Enter `npm start`.
4. Either enter `npm test` or open `http://localhost:8085/unit/` in a browser.

Optimize Components (Minify) for Better Performance

Improve the performance of your components and reduce download sizes by minifying the JavaScript code with Developer Cloud Service.

Optimize Your Components

An example of optimizing component source code (optional) is provided for the `Sample-Text-With-Image` component. You can use the RequireJS optimizer to minify and combine source code. Minifying a component removes unwanted space in the JavaScript code, resulting in better performance and reduced download size. Optimizing involves the following actions:

- Minify the JavaScript.
- Compress the CSS.
- Combine the JavaScript, HTML, and CSS into a single file, `render.js`.

This optimization reduces download size and improves performance.

Enable Optimization for Your Component

In the `Sample-Text-With-Image` component, optimization will minify and combine all the files that are marked as dependencies in `render.js` (`template.html`, `data-defaults.js`, and `design.css`) into `render.js` as a single file. Other components that already support optimization follow:

- `Sample-Folder-List`
- `Sample-File-List`
- `Sample-Documents-Manager`
- `Sample-Process-Start-Form`
- `Sample-Process-Task-List`
- `Sample-Process-Task-Details`

To enable optimization for your local component, take the following steps (required):

1. Copy `gulpfile.js` from the `Sample-Text-With-Image` component to your component.
2. Fix the module names in `gulpfile.js` for your `render.js` dependencies. Refer to the comments in `gulpfile.js`.

> **Note:**
>
> If you have an existing component, with a `css` folder under `assets`, rename the `css` folder to `styles` to avoid errors during optimization.

Run Continuous Integration Jobs

You can run continuous integration jobs to keep your site up to date.

Refer to the "Getting Started with CEC Custom Components Development" wiki in Developer Cloud Service. You can configure a build job for the continuous integration.

Develop Translatable Components for Multilingual Sites

Developers of custom components can designate which strings within a custom component should take part in page translations for multilingual (MLS) sites.

To do this, you need to introduce an `nls` property at the top level when saving your data to the `customSettingsData` object.

For example:

```
SitesSDK.setProperty('customSettingsData', {
        'nls': {
            linkText: 'More...'
        }
    });
```

When a translation job is created, Oracle Content Management will check the top-level properties of the `customSettingsData` object and export the entire `nls` object for each custom component instance on the page. Translators will translate these values, which can then be imported back into the site.

Once the site translations have been imported, the correct version of the `nls` object will be returned in the `customSettingsData` object for the translated locale.

For example, if you translated the site to French and then render the page in the French locale, the value of the `nls` object in the `customSettingsData` object passed to the custom component would be updated to:

```
{
     'nls': {
         'linkText': 'Plus...'
     }
 },
```

The format of the `nls` object in `customSettingsData` should be limited to name/value pairs. This aids in translation and ensures that translated values can be applied correctly to the base values in the site when the page is rendered in a translated locale.

Build an H1 Component with a Settings Panel

You can create a minimal Oracle Content Management component that has a simple HTML template and CSS. This H1 component has a simple settings panel and an entry for the theme in `design.json` to allow other Oracle Content Management users to pick from three built-in styles when using the component in an editor.

When you create the new component, you get a set of seeded files that will work out of the box. The seeded files cover most of the functionality of a component within the product. You can change the seeded code to create your own component, which requires only a small subset of seeded code to achieve the end result.

You can build an H1 component with a settings panel in five steps:

1. Create a New Local Component
2. Build the Basic H1 Component
3. Add CSS for Your Component
4. Add a Settings Panel to Change Heading Text
5. Update the Theme for Others to Pick the H1 Component Style

Create a New Local Component

Create a local component with Oracle Content Management that you can immediately drop onto the page. This is the starting point for creating any new component.

To create a new local component:

1. Click **Developer** in the side navigation, then click **View all Components**.
2. Select **Create > Create Local Component**.
3. Enter a name, for example, `H1_Component`, and optionally, a description.
4. Click **Create** to create a new component.

Now that you have successfully created a component, you should see it in the list of component on the component page, as well as in the **Add > Custom** component palette for any site you create. Use the following steps to validate your component creation (Checkpoint 1):

1. Create a new site using any seeded template; for example, create a site named `ComponentTest` using the `StarterTemplate` template.
2. Select the **Edit** option, and create an update for the site to open it in an editor.
3. Edit a page within the site you created.
4. Click the **Add** button (**+**) on the left bar, and select **Custom** for the list of custom components.
5. Select **H1_Component** from the custom component palette and drop it onto the page.

   You should now see a default rendering for the local component you created.
6. Select the context menu for the component.
7. Choose **Settings** from the drop-down menu.

   You can change the settings to see how seeded component rendering will change.

   You can modify seeded files to create a new custom component.

Build the Basic H1 Component

You can remove most of the contents in seeded files to create an H1 component. It displays the heading text that you seed when you create `viewModel`. Later you can provide settings and styles for the component.

To review the structure of your local component:

1. Using the Oracle Content Management Desktop Sync App, locate your component and sync it with the file system.
   - In a recent version of the Desktop Sync App, choose the **Start Sync** or **Select Folders to Sync** option.
   - If you don't have the Desktop Sync App, you can select the component on the **Components** tab of Oracle Content Management and then drill down to see the files.
2. If you list the files under the component, you can see these files:
   - The component files in the `assets` folder:
     - `render.mjs`
     - `settings.html`
   - `appinfo.json`: JSON file with the component description.

     See About Developing Components.
   - `folder_icon.jpg`: Icon that is displayed in the Component Catalog.

To build an H1 Component:

1. Open the `appinfo.json` file and replace its contents with the following lines:

```
{
   "id": "h1-component-id",

   "settingsData": {
             "settingsHeight": 90,
             "settingsWidth": 300,
             "settingsRenderOption": "inline",
             "componentLayouts": [ ],
             "triggers": [ ],
             "actions": [ ]
   },
   "initialData": {
             "componentId": "h1-component-id",
             "customSettingsData": {
                     "headingText": "Heading 1"
             },
             "nestedComponents": [ ]
   }
}
```

2. Open the `render.mjs` file in the `assets` folder in your favorite text editor.

3. Change the contents of `render.mjs` to the following lines:

```
import CommonUtils from './common.mjs';

// The Custom Component class will be the "default" export from the
module
export default class {

    constructor(args) {
        // store the args
        this.mode = args.viewMode;
        this.id = args.id;

        // store the path to the <component>/assets folder
        this.assetsPath =
            import.meta.url.replace('/render.mjs', '');

        // get the OCM environment resources
        this.sitesSDK = args.SitesSDK;
        this.Mustache = SCSRenderAPI.getMustache();

        // add in the event listeners
        this.addEventListeners();
    }

    // add in the listeners for whenever the settings values change
    // in this case, we want to re-render the component on the
screen
    addEventListeners() {
        // listen for settings update

this.sitesSDK.subscribe(this.sitesSDK.MESSAGE_TYPES.SETTINGS_UPDATED
, (props) => {
            if (props.property === 'customSettingsData') {
                this.renderComponent({
                    customSettingsData: props.value
                });
            }
        });
    }

    // insert the component's HTML into the page
    // after it has added the component, it applies any
clickHandlers to elements that were added to the page
    renderComponent(args) {
        Promise.all([SCSRenderAPI.importText(this.assetsPath + '/
template.html'),
            SCSRenderAPI.importCSS(this.assetsPath + '/styles/
design.css')
        ]).then((componentResources) => {
            const componentTemplate = componentResources[0];

            // use the common code to generate the HTML for this
component based on the componentLayout and customSettingsData
            const componentHTML = CommonUtils.createHTML({
```

```
                Mustache: this.Mustache,
                componentLayout:
this.sitesSDK.getProperty('componentLayout'),
                customSettingsData:
this.sitesSDK.getProperty('customSettingsData'),
                id: this.id,
                template: componentTemplate
            }, args);

            // replace the content of the container with the rendered HTML
            this.container.innerHTML = componentHTML;
        });
    }

    // the render method is called to render the component dynamically
onto the page
    render(container) {
        this.container = container;
        this.renderComponent();
    }
}
```

4. In the `assets` folder, create a new file, `render.html`, to be the simple HTML template of the component.
5. Use the following contents in the `template.html` file:

```
<h1>{{headingText}}</h1>
```

6. Also in the assets folder, create a new file called `common.mjs` and add the following code:

```
// Common Utilities Class
export default class {
    constructor() {}

    static createHTML(context, args) {
        // extract all the required dependencies from the context
        const Mustache = context.Mustache,
            customSettingsData = context.customSettingsData,
            id = context.id,
            template = context.template;

        // extract the original values (or apply deafult values)
        const customData = (args && args.customSettingsData) ||
customSettingsData || {};
        customData.nls = customData.nls || {};

        // create the model
        const model = {
            headingText: customData.nls.headingText || ''
        };

        // render the component
        try {
            return Mustache.render(template, model);
```

```
            } catch (e) {
                console.log('Failed to expand Mustache template.', e);
                return '';
            }
        }
    }
```

This file is called from the `render.mjs` file.

The component `assets` folder now contains the following files.

- template.html
- render.mjs
- settings.html
- common.mjs

Add the new H1 component to your page (Checkpoint 2).

Add CSS for Your Component

You can add a CSS that will provide a default style for your component.

To add a CSS:

1. Add a `design.css` file to the `assets` folder of your component, with the following contents:

```
.h1-component-default-style .scs-component-content {
 font-family: "Helvetica Neue", "Helvetica", "Arial", sans-serif;
 font-size: 24px;
 color:red;
 font-weight: normal; }
```

2. Add to `appinfo.json` to declare the style class prefix that will be used to style your component. If a `styleClassName` of `h1-component` is added, when your component is dropped onto the page, the default style will be `h1-component-default-style`. The new contents of `appinfo.json` follows:

```
{
   "id": "h1-component-id",

   "settingsData": {
             "settingsHeight": 90,
             "settingsWidth": 300,
             "settingsRenderOption": "inline",
             "componentLayouts": [ ],
             "triggers": [ ],
             "actions": [ ]
   },
   "initialData": {
             "componentId": "h1-component-id",
             "styleClassName":"h1-component",
             "customSettingsData": {
                    "headingText": "Heading 1"
```

```
            },
            "nestedComponents": [ ]
    }
}
```

Verify that your component will now pick up its default data from the `appinfo.json` file.

Add a Settings Panel to Change Heading Text

Update the `settings.html` file to provide a settings panel that can be used to set the text of the H1 component.

To add a settings panel to change heading text:

1. Update the `settings.html` file to have the following contents:

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

    <title>Sample Custom Component</title>

    <!-- include sample apps styling -->
    <link href="/_sitescloud/renderer/app/sdk/css/app-styles.css"
rel="stylesheet">

    <!-- include supporting files -->
    <script type="text/javascript" src="/_sitescloud/renderer/app/apps/js/
mustache.min.js"></script>
    <script type="text/javascript" src="/_sitescloud/renderer/app/apps/js/
jquery.min.js"></script>

    <!-- include the Sites SDK -->
    <script type="text/javascript" src="/_sitescloud/renderer/app/sdk/js/
sites.min.js"></script>

    <template id="settingsTemplate">
        <div>
            <!-- H1 heading Text -->
            <label id="headingTextLabel" for="headingText"
class="settings-heading">Title Text</label>
            <input id="headingText" value="{{nls.headingText}}"
placeholder="example: Template Component" class="settings-text-box"
style="margin-bottom: 10px;">
        </div>
    </template>
</head>

<body style="display:none; margin:0px; padding-
left:3px;background:transparent;background-image:none;">

    <!-- conatiner for the settings panel -->
    <div id="settingsContainer" class="scs-component-settings">
```

```
        <div data-bind="text: 'waiting for initialization to
complete'" class="settings-heading"></div>
    </div>

    <script type="text/javascript">
        // get the Mustache template for the settings panel
        var template = $('#settingsTemplate').html();

        // define the viewModel object
        var model = {};

        // save whenever any updates occur
        var lastState = '';
        var saveUpdates = function () {
            var saveConfig = {
                    nls: {
                        headingText: $('#headingText').val(),
                    }
                },
                newState = JSON.stringify(saveConfig);

            // if model has changed, save it
            if (newState !== lastState) {
                // update the last saved state
                lastState = newState;

                SitesSDK.setProperty('customSettingsData',
saveConfig);
            }
        };

        // Get custom settings and render the settings
        SitesSDK.getProperty('customSettingsData', function (data) {
            // update the model with the current values
            model.nls = data.nls || {};
            model.headingText = model.nls.headingText;

            // initialize the last saved state to the fetched model
            lastState = JSON.stringify(model);

            // apply the model to the template
            var html = Mustache.render(template, model);

            // add the rendered HTML to the page
            $('#settingsContainer').html(html);

            // save updates whenever something changes
            $('#headingText').on('blur', saveUpdates);

            // show the body
            $('body').show();
        });
    </script>
</body>
```

```
</html>
```

2. Select your component in Site Builder, and click **Settings**.

   The settings panel is small, and is set to be embedded at the top of the component settings panel. Look for the section titled "Heading Text".

To verify that you can change heading text (Checkpoint 4):

When you change heading text in the settings panel, the component updates to show the new text.

Update the Theme for Others to Pick the H1 Component Style

You can register styles for your component with the theme, so that other users can switch between the styles you provide for your component from the settings panel or **Style** tab.

To update the theme for other users to pick the component style:

1. Add some more styles to your component `design.css` file. Prefix each style with the component's registered `styleClassName` as defined in `appinfo.json`. For this component, that prefix is `h1-component`.

   Two more styles, `h1-component-gothic-style` and `h1-component-courier-style` have been added.

   The new contents of `design.css` will now be as follows:

```
.h1-component-default-style .scs-component-content {
 font-family: "Helvetica Neue", "Helvetica", "Arial", sans-serif;
 font-size: 24px;
 color:red;
 font-weight: normal; }
.h1-component-gothic-style .scs-component-content {
 font-family: "Century Gothic","CenturyGothic","AppleGothic",sans-serif;
 font-size: 32px;
 font-weight: bold; }
.h1-component-courier-style .scs-component-content {
 font-family: "Courier";
 font-size: 32px;
 font-weight: bold; }
```

2. Register your styles in the theme's `design.json` file. You can find this file in the theme the site is using. Drill down to the theme files in the designs folder, then to the defaults folder, and add a section for your component to `design.json`.

   The **bold** text that follows is an example of what to add:

```
          "news-article": {
                  "styles": [{
                                          "name": "News Article 1",
                                          "class": "news-article-default-
style"
                                          },
                          {
                                          "name": "News Article 2",
```

```
                                        "class": "news-article-
style-1"
                                        }
                                    ]
            },
            "h1-component": {
     "styles": [{
       "name": "Plain",
       "class": "h1-component-default-style"
    }, {
             "name": "Courier",
             "class": "h1-component-courier-style"
    }, { "name": "Gothic",
         "class": "h1-component-gothic-style"
    }]
   }
   },
   "componentIcons": {
          "scs-socialbar": {
                    "icons": [
```

The names in the `design.json` snippet that was added (`"Plain", "Courier", "Gothic"`) will appear in the settings panel for your component on the style tab as below. When selected, they will apply the corresponding styles ("*h1-component-default-style*", "*h1-component-courier-style*", "*h1-component-gothic-style*") respectively, on your component.

Compare Local Components to Remote Components

Local components and remote components are implemented differently.

The following table lists differences in how components can be implemented. It may help you determine whether you want to use a local component or a remote component rendered in an inline frame.

Local Component Implementation	Remote Component Implementation
Integrates complex content-centric user interface into Oracle Content Management	Integrates application logic into Oracle Content Management
Executes JavaScript to render content in the page	Uses the `<iframe>` HTML tag to render content in the page
Uses any JavaScript technology stack	Can use any technology, not just JavaScript
Leverages the Oracle Content Management JavaScript stack and can re-use Oracle Content Management components	Doesn't integrate with the Oracle Content Management JavaScript stack
Adds assets dependencies from `/assets` documents stored in the Oracle Content Management server	Requires HTTPS URLs and corresponding certificates for all asset access
Is hosted with a template and published independently of a theme by Oracle Content Management	Requires a hosted middle-tier server for the URL endpoints

Local Component Implementation	Remote Component Implementation
Because code executes within the page, if the component code breaks, it may break the page	The page renders independently to the content of inline frames, so the page won't break if the component fails
May slow down the rendering of the page if the component is slow to render	The page loads independently to inline frame content, so the page will load as fast as it can and then load the content of any inline frame

> **Note:**
>
> While the rendering of local components and remote components differ, the Settings panel implementation in Site Builder is the same. The Settings panel for both types of components is rendered using inline frames, and it uses the same JavaScript SDK to allow for both cross-boundary and cross-domain communication.

Render Component Settings

To render component settings, you can use a component settings URL and component settings rendering options.

Component Settings URL

A component settings URL is rendered using an inline frame and called with parameters to allow specific settings of an actual component dropped onto a page. The settings URL has this format:

```
{Component Settings URL}?instance=<app-
instance>&width=<width>&currCompId=<id of the app associated with the
settings panel>&locale=<locale>
```

Name	Type	Description
`Component Settings URL`	URL	Component Settings URL of a component
`width`	Number	Width of the Settings inline frame in pixels
`currCompId`	String	Current component ID of the component edited by the Settings panel
`locale`	String	Current locale of the host site (Site Builder). Format is `<language>_<dialect>`. Example: `En_us`.

Local Component Settings Rendering Options

You have three options for rendering the inline frame in the Settings panel for a local component, based on the size and complexity of the inline frame. Each option is specified in the `settingsRenderOption` property.

> **Note:**
>
> These options are available only to local components. Remote components Settings panels always render in a dialog.

- `inline`
  - Use this option only if you have a few small properties for the user to enter.
  - The inline frame will be inserted onto the **General** tab in place of the usual button to navigate to the Settings panel.
  - Provides the most integrated solution, requiring the least amount of clicks for a user, but it has limited space.
- `panel`
  - Use this option when you have a longer list of properties, but they can still be reasonably displayed within the 300 pixels of the standard Settings panel.
  - The inline frame will slide into view and a Back button will appear to return the user to the General tab.
  - Provides an integrated solution where you can interact with the inline frame on the page.
- `dialog`
  - The default mode for handling more general settings layouts requiring a complex user interface.
  - Displays the inline frame in a modal dialog on the page.

Local Component Implementation

The component exports a default class export from the JavaScript module. The JavaScript module is dynamically imported into the page and then a new instance of the class is created.

For example, in render.mjs, the custom component class will be the default export from the module export.

```
export default class {
      constructor(args) {...}
      render(container) {...}
}
```

The following settings are contained in *`args`*:

- `SitesSDK`: The Oracle Content Management Sites SDK.
- `id`: The unique ID (`GUID`) for the component added to the page.
- `viewMode`: The current mode the page is rendering. When a page is being edited, it's "Edit". When a page is previewed, it's "Navigate". At runtime, which is when the site is published, the value is undefined. You can provide different implementations based on what functionality should be exposed for each mode. For example, links shouldn't be active when the page is running in Edit mode.

> **Note:**
>
> It isn't a requirement to use JQuery or Knockout for your component, but if you want to leverage Oracle Content Management features such as nested components, you must use the version of Knockout provided by Oracle Content Management. This version of Knockout has extended component registration and handlers, which wouldn't be available to you otherwise.

For the component itself, the SDK is passed in when the component is instantiated so that the component can communicate with the page lifecycle. The page lifecycle functions must be implemented by the component and are called by Oracle Content Management to render the component on the page.

Mandatory and optional APIs are provided to implement a component.

Mandatory APIs

customComponent`.render(container)`: Asks the component to insert itself into the provided DOM container element.

- `container`: DOM container element for the custom component HTML.

Optional APIs

`customComponent.dispose()`: Called when the component is being removed from the page. Provides an opportunity for the component to remove any resources that are no longer required.

Style Classes for Components

You can create a defined list of styles that can be applied to your component by users.

Having a predefined list of styles for your component follows the same model as defining style classes for components provided by Oracle Content Management through a theme's design files. You name your custom style classes in the `appinfo.json` file for the component.

You define additional styles in the `design.css` and `design.json` files. The `json` file provides a mapping from the name that will appear in the user interface to the actual underlying `css` class name, and the `css` file provides the details for each style class.

The `design.json` file has the following structure for components:

```
{
    "componentStyles": {
        "scs-image": {
            "styles": []
        },
        "scs-map": {
            "styles": []
        },
        "scs-title": {
            "styles": []
        },
```

```
        "scs-paragraph": {
            "styles": []
        },
        "scs-txt": {
            "styles": []
        },
        "scs-divider": {
            "styles": []
        },
        "scs-button": {
            "styles": []
        },
        "scs-app": {
            "styles": []
        },
        "scs-spacer": {
        },
        "scs-gallery": {
            "styles": []
        },
        "scs-youtube": {
            "styles": []
        },
        "scs-socialbar": {
            "styles": []
        },
        "scs-document": {
            "styles": []
        }
    }
}
```

Each of the `"styles":[]` entries can contain a list of styles for that particular component. The `"name"` can either be a reference to a built-in localized string, or a specified value to use. For example, the Title component provides these default styles:

```
{
    "styles": [{
                    "name": "COMP_STYLE_FLAT",
                    "class": "scs-title-default-style"
                },
                {
                     "name": "COMP_STYLE_HIGHLIGHT",
                     "class": "scs-title-style-2"
                },
                {
                     "name": "COMP_STYLE_DIVIDER",
                     "class": "scs-title-style-3"
                }
]
}
```

The name values are mapped to the actual words to display in the user interface, like this:

```
"COMP_STYLE_FLAT": "Flat",
"COMP_STYLE_HIGHLIGHT": "Highlight",
"COMP_STYLE_DIVIDER": "Divider",
```

The `css` file provides the definitions for the class values:

```
.scs-title-default-style {
  color: #333333;
  display: block;
  font-family: "Helvetica Neue", "Helvetica", "Arial", sans-serif;
  font-size: 24px;
  font-weight: normal;
                          }
```

As an example, in the theme's `design.json` file, you can add entries for your component based on the `initialData.compomentId` value you define in the `components.json` file:

```
"componentId": "news-article"
```

The corresponding entries in the `design.json` file would be these:

```
"componentStyles": {
        "news-article": {
                "styles": [{
                        "name": "News Article 1",
                        "class": "news-article-default-style"
                  },
                  {
                        "name": "News Article 2",
                        "class": "news-article-style-1"
                  }]
        },
```

The corresponding entries in the `design.css` file would be these:

```
.news-article-default-style .scs-image {…}
.news-article-style-1 .scs-image {…}
```

How to Style Built-In Components

Create your own look and feel to style built-in components in Oracle Content Management by overriding and extending built-in styles.

Built-in components get their visual styling from two places:

- `comp.css`, a built-in CSS file that specifies the *base* look of each component
- `design.css`, a CSS file that is part of the theme that your site is using

In the `design.css` file, you can override and extend the built-in `comp.css` styles to create your own look and feel. In a theme, the `design.css` file is in the `designs/default` directory.

The following topics describe classes in the `comp.css` file are common to all built-in components and provide an overview of defining a theme:

- Component Styling Basics
- Component-Specific Styling
- Set Component Properties

Component Styling Basics

All built-in Sites components share a similar CSS class structure.

Each component has the following three CSS classes applied to its outermost `<div>` element:

```
scs-component scs-type design-style
```

The *`type`* is the component type (such as `image`, `gallery`, or `divider`). The *`design-style`* is the chosen style class for a component, as defined in the theme's file.

Theme Styles Basics

A theme's `design.json` file lists all the styles (`frame`, `shadow`, `highlighted`, and so on) that can be applied to each type of component (such as `button` or `image`). Each style has both a `display name` and a `class name`. The display name is shown on the **Style** tab of the **Settings** panel. The class name refers to a CSS selector in the theme's `design.css` file. For example, the entry for the `button` component follows:

```
"scs-button": {
         "styles": [{
                        "name": "COMP_STYLE_ALTA_SMALL",
                        "class": "scs-button-default-style"
                        },
                    {
                        "name": "COMP_STYLE_ALTA_LARGE",
                        "class": "scs-button-style-2"
                        },
                    {
                        "name": "COMP_STYLE_SIMPLE",
                        "class": "scs-button-style-3"
                        }
                    ]
},
```

Names of built-in components are translated, so you see a key to get style name from the resource bundle. If you add a `button` component to a page and then choose the **Simple** style in the **Settings > Style** panel, the `design.json` file associates the display

name **Simple** (key `COMP_STYLE_SIMPLE`) with the class name `scs-button-style-3`. The `button` will be rendered with the following classes:

```
scs-component scs-button scs-button-style-3
```

If no style is chosen for a given component, then the default style, `scs-type-default-style`, is used. In the preceding example, the button will be rendered with the following classes:

```
scs-component scs-button scs-button-default-style
```

The scs-component-content Style

For every built-in component, inside the `scs-component <div>` mentioned previously, there is a `content <div>` with the CSS class `scs-component-content`. In other words:

```
scs-component scs-type design-style
scs-component-content
```

In the `design.css` file, the `scs-component-content` class is often used to style the "box" around the component (for example, to apply a border or shadow).

It's worth noting that in the built-in `comp.css` file, the common `scs-component-content`class is defined with `position:relative` and `display:inline-block`, among other CSS properties.

While `scs-component-content` is useful for styling the "box" around each component, component-specific classes are needed to fully style a component. See Component-Specific Styling.

Component-Specific Styling

You can apply specific styles to images, buttons, documents, paragraphs, titles, maps, and other components.

Image Component

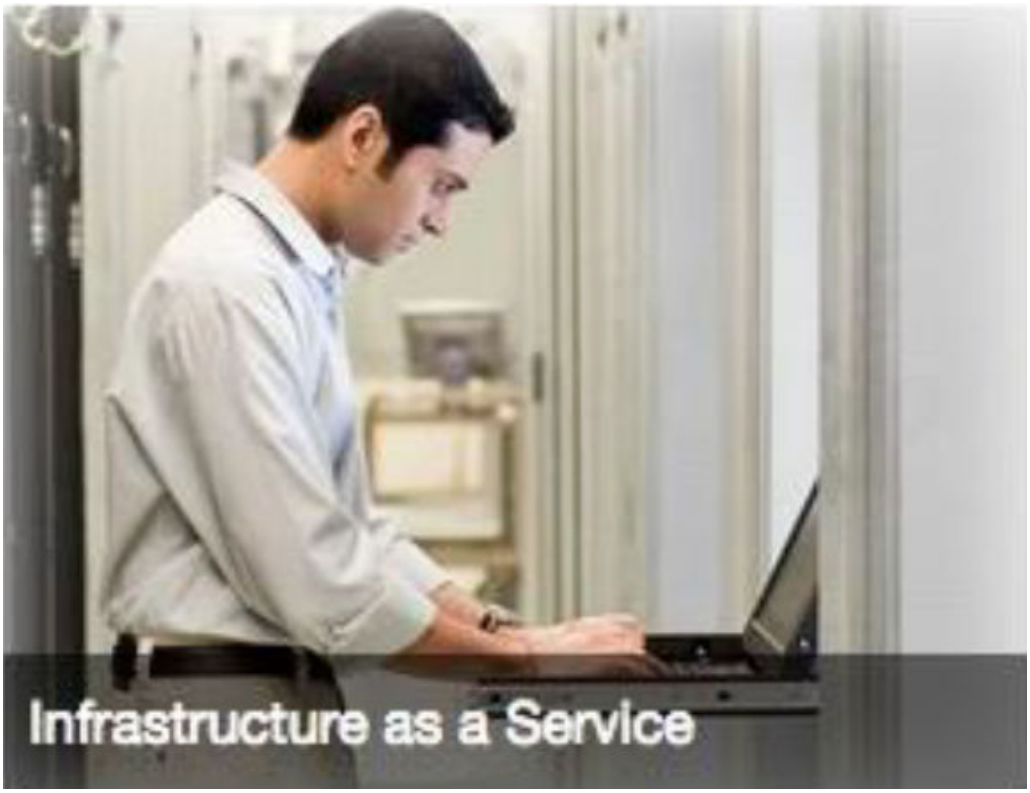
The Image component has the following CSS class structure below the `scs-component-content` class:

```
scs-image-container
scs-image-link
scs-image-image
scs-image-caption
```

The `scs-image-image` class is applied to the `<img>` tag itself. The `scs-image-caption` class is used to style the caption, if the caption is present.

The `scs-image-link` class is present only if the image has a link attached. Neither it nor the `scs-image-container` class typically requires custom styling.

By default, the image caption is rendered as a semitransparent overlay stretched across bottom of the image.




```
.scs-image .scs-image-caption {
 position: absolute;
 left: 0px;
 bottom: 0px;
 right: 0px;
 background-color: rgba(0, 0, 0, 0.54);
 padding: 0.5em;
 color: #FFFFFF;
}
```

To place the captions at the top of the image and change colors, add extra style for the Image component in the `design.json` file and then define CSS for it in the `design.css` file.


```
.scs-image-style-17 .scs-image-caption {
 position: absolute;
 top: 0px;
 height: 35px;
 font-weight: bold;
 background-color: rgba(122, 213, 256, 0.54);
 color: #515151;
}
```

Button Component

The Button component has the following class structure:

```
scs-button-button
    scs-button-text
```

The `scs-button-button` class is the clickable `<div>`, styled to look like a button. The `scs-button-text` class is used to style the text inside the button.

For example, test changing the look and feel of the Button component by adding extra style for it in the `design.json` file, and then define CSS for it in the `design.css` file.


```
.design-style .scs-button-button {
 background-image: linear-gradient(
  to top, #E3E7E9 0%, #E7EBED 50%, #F1F3F3 100%);
 border: 1px solid #c4ced7;
 color: #000000;
}

.design-style .scs-button-button:hover {
background: #f7f8f9;
border: 1px solid #c4ced7;
 color: #0572ce;
}

.design-style .scs-button-button:active {
background: #0572ce;
border: 1px solid #0572ce;
color: #ffffff;
}
```


```
.scs-button-style-4 .scs-button-button {
 background-image: radial-gradient(
  red, yellow, green
 );
border: 1px solid #c4ced7;
color: #000000;
}

.scs-button-style-4 .scs-button-button:hover {
 background: #f7f8f9;
 border: 1px solid #c4ced7;
 color: #0572ce;
}

.scs-button-style-4 .scs-button-button:active {
 background: #0572ce;
 border: 1px solid #0572ce;
 color: #ffffff;
}
```

Document

The Document component has the following class structure:

```
scs-document-container
 scs-document-cap
  scs-document-title
  scs-document-desc
```

The `scs-document-container` class wraps the document viewer, and is not normally styled.

Gallery

The Gallery component has a single class wrapping the underlying JSSOR slider component:

```
scs-gallery-container
```

The JSSOR slider uses several classes that you can style as well:

```
jssorb14 (navigator)
jssora02l (left arrow)
jssora02r (right arrow)
jssort07 (thumbnails)
```

Gallery Grid

The classes used for the Gallery Grid component depend on the layout and cropping selected in the **Settings** panel:

```
scs-gallerygrid-container scs-gallerygrid-layout
  scs-gallerygrid-cell
    scs-image (multiple)
```

Depending on the cropping and layout settings selected for Gallery Grid, the value of `layout` \will be stretch, crop, fit, or flowing.

The `scs-gallerygrid-cell` class is present only for Column layouts.

Social Bar

The Social Bar component has the following class structure:

```
scs-socialbar-container
  scs-socialbar-icon
```

The `scs-socialbar-icon` class is applied to each `<img>` tag in the social bar.

Paragraph

The Paragraph component has only a single class wrapping the actual paragraph text:

```
scs-paragraph-text
```

For example, to make text that you contribute in the Paragraph component to have an engraved-text-on-metal effect, add an extra style class in the `design.json` file and then define CSS for it in the `design.css` file.


```
.scs-paragraph-style-7 {
 font-size: 24px;
 font-family: Arial, Helvetica, sans-serif;
 font-weight: 700;
 padding: .3em;
 color: #000000;
 background: #666666;
 text-shadow: 0px 1px 1px #ffffff;
}
```

Or if you want to get fancy, use something like the next example.


```
.scs-paragraph-style-8 {
 padding: 20px;
 margin: 10px;
 background: #ff0030;
 color: #fff;
 font-size: 21px;
 font-weight: bold;
 line-height: 1.3em;
 border: 2px dashed #fff;
 border-radius: 10px;
 box-shadow: 0 0 0 4px #ff0030, 2px 1px 6px 4px rgba(10, 10, 0, 0.5);
 text-shadow: -1px -1px #aa3030;
```

```
 font-weight: normal;
}
```

Title

The Title component also has only a single class wrapping the actual text:

```
scs-title-text
```

Map

The Map component has a single class wrapping the map rendition:

```
scs-map-content
```

This class is not normally styled.

Custom Local Component

The Custom Local Component has only a single class wrapping the actual component:

```
scs-custom-component-wrapper
```

You have full control of the CSS styles that you need to use to render custom view for your custom local component. A local component is rendered inline; that is, you can directly apply CSS styles defined in your theme or in the `design.css` file.

Custom Remote Component

The Custom Remote Component has only a single class wrapping its iframe:

```
scs-app-iframe-wrapper
```

In addition to applying CSS styles defined in your custom remote component, you can leverage Sites SDK to fetch a `design.css` file from the host site.

```
// fetch current theme design from host site and then add it to the page
SitesSDK.getSiteProperty('theme',function(data){
 // check if we got a url back
 if ( data.url && typeof data.url === 'string' ) {
  if ( data.url !== '') {
   // theme is loaded, so dynamically inject theme
   SitesSDK.Utils.addSiteThemeDesign(data.url);
  }
 }
});
```

Thus, you can make your component inherit styling from the host style.

Divider

Although there are no component-specific classes for the Divider component, the `<hr>` tag itself can be styled.

For example, you can create a dotted divider:

```
.design-style .scs-divider hr {
border-top: 1px dotted #333333;
}
```

Video, YouTube, Spacer

There are no component-specific classes for Video, YouTube, or Spacer components.

Set Component Properties

You can configure component properties for use in a site.

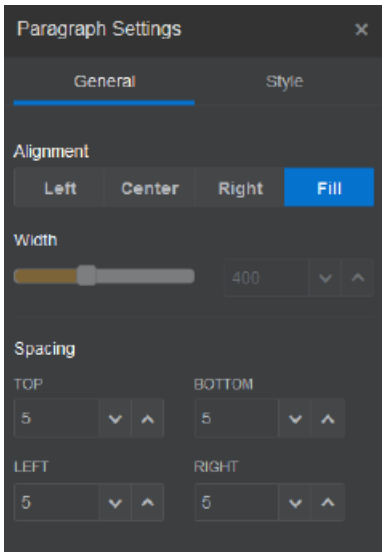
Components are the individual parts of a web page, which include text, titles, images, buttons, dividers, maps, galleries, videos, and so on. When you create a design for a theme, you also must specify the default settings for each type of component. Each component has settings — such as size, alignment, spacing, color, and borders — that define how the component looks and behaves. The settings vary based on the component. You also can choose whether a component properties can be changed by users once it is available in a site in a new theme.

As an example, these steps illustrate how you can configure settings for a Paragraph component:

1. With your development site open in Edit mode, select a page that has a Paragraph component in it, or add a Paragraph component.
2. Click the Paragraph component, then click  in the corner of the component, and then choose **Settings**.

   You'll see a **Settings** panel displayed where you can make selections for the component.

> **Note:**
>
> The settings options are specific to each type of component. If working with local or remote custom components, you'll see a **Custom Settings** link.



3. For example, you can change the settings for alignment, width, spacing, fonts, style, color, size, and so on.

   The component appearance changes to the new settings.

4. When you're finished, click  . Your settings are applied to the page.

Components Rendered in Inline Frames

Components that are rendered in inline frames can be specified in Oracle Content Management pages by registering and adding components from external servers (referred to as **remote components**), and also by selecting the inline frame option when creating a local component. This type of component can extend functionality for sites, such as adding a social component or a check-out cart component.

For a remote component, you must specify the endpoint URL. For a local component rendered in an inline frame, the URLs are derived from the name of the component in the Component Catalog.

- Endpoint URL: The content of the widget is fetched from this URL and embedded within an inline frame.
- Settings URL: This URL is rendered in an inline frame to configure the component once it is dropped onto a page.

The component can render static or dynamic data visualizations and display a form or other interactive user interface that extends site functionality. See Render Component Settings.

Components Provided by Oracle Content Management

A set of components that are rendered in inline frames is provided with Oracle Content Management.

Name	Type	Id
Folder List	scs-app	Folder List
File List	scs-app	File List

Name	Type	Id
Documents Manager	scs-app	Documents Manager
Facebook Like	scs-app	Facebook Like
Twitter Follow	scs-app	Twitter Follow
Twitter Share	scs-app	Twitter Share
Facebook Recommend	scs-app	Facebook Recommend

Component Registration

Before a remote component rendered in inline frames can be used in a site, the endpoint URL must be registered using HTTPS. This information is stored in the Component Catalog. The endpoint must allow the URL to display in an inline frame; don't set `X-Frame-Options="sameorigin"` in the header. For a local component, because the files are stored on the Oracle Content Management server, this endpoint criteria is automatically met.

When a component rendered in an inline frames is registered, a new GUID is generated that represents the component. If such a component is registered in an Oracle Content Management instance multiple times, it will get multiple GUIDs because they represent each registration of that component. When the component is registered, the description is all that is stored in Oracle Content Management against the GUID; the component will still run from its remote endpoint. See Register a Remote Component.

Once a component rendered in an inline frame is registered, an instance ID is also generated. This instance ID represents a component registered with a certain Oracle Content Management tenant. The same component, registered more than once within the same Oracle Content Management tenant or with a different Oracle Content Management tenant, will have different instance IDs.

Remote Component Settings Persistence

When the Settings URL for a component rendered in an inline frame is rendered in a Settings panel dialog, the instance ID and component ID are both provided. This allows the component to choose to persist any settings itself in its own server, indexed by instance ID and component ID. Alternatively, the Oracle Content Management Sites SDK can be used to allow up to 1.5 KB of JSON data to be stored in the site page's page model against the component ID.

Using the Sites SDK to persist settings has two benefits for components rendered in inline frames:

- The component can easily participate in page versions, page updates, and the site publishing model.
- The component can comprise HTML endpoints that execute in the browser, as opposed to executing in a back-end system.

See Oracle Content Management SDKs.

About the Instance ID and Structure for Components Rendered in Inline Frames

The component Instance ID is the unique identifier for a component rendered in an inline frame within a site.

When a user drags and drops a component rendered in an inline frame from the Component Catalog onto a site page, a provisioning call is made to Oracle Sites Cloud Service to generate a new unique component instance ID. This ID is guaranteed to be unique and all such component instances provisioned on that service will get the same instance ID.

A component instance ID contains additional information that can be used to secure the settings and use of a component rendered in an inline frame, so that the component can be sure that the Settings update is coming from a trusted place.

The instance ID parameter enables developers to identify the site and authenticate the calling party. The caller is authenticated by verifying a digital signature that is generated using the component secret key. The secret key is generated during the component registration process.

The component instance consists of two parts separated by a '.' delimiter: data and structure.

Component Instance: Data

The data portion of the instance for a component rendered in an inline frame is a Base64 JSON encoded string. Here's the structure of the JSON string:

```
{
  "instanceid": "BBDC7614F693B75110D811E6C0B77C935FAEC5112E5E",
  "permissions": "",
  "entitlements": "",
  "signdate": "1435426735293",
  "sitedomain": "service1-tenant4.localhost"
}
```

Field Name	Description
`instanceid`	Unique identifier of a component rendered in an inline frame for an Oracle Content Management tenant.
`signdate`	Signature generation date.
`sitedomain`	Domain name of the Oracle Content Management instance.
`permissions`	Set of permissions of the site member. In editing mode, it will have the value `"SITE_OWNER"`; otherwise, it will have no value.
`entitlements`	List of premium features purchased by the site owner.

Component Instance: Signature

The data portion of the component instance is serialized before being signed by an `APP_SECRET_KEY`. This secret key must be generated and shown to the developer while registering the component. The signature is calculated by generating a hash of the data

portion of the component instance (a serialized JSON structure) with the secret key as shown here:

```
$signature = HMAC (serialized JSON structure, APP_SECRET_KEY)
```

The hash algorithm used in generating the signature is SHA256. The token is then the concatenation of the serialized JSON structure and the generated signature component as shown here:

```
$instance = {base64encoded serialized JSON structure}.
{base64encoded $signature}
```

Example:

```
//base64 encoded serialized object           //signature
eyJpbnN0YW5jZWlkIjoiQTRGOTE3REY5OTZEN0Q3ODBCMjUzODZFOTFEMDA3ODJGMjVBRjY
2Rjc3OTIiLCJzaWduZGF0ZSI6IjE0NDU2MzcwNTk5MTciLCJzaXRlZG9tYWluIjoic2Vydm
ljZTEtdGVuYW50MS51cy5vcmFjbGUuY29tIiwicGVybWlzc2lvbnMiOiJTSVRFX09XTkVSI
iwiZW50aXRsZW1lbnRzIjoiIn0=.5p3of7t11OwuysF3zpm+YgICSHH8C/
BHczdbVZx2VH8=
```

Security for Remote Components

Oracle Content Management enables third-party developers to integrate their custom components into the Oracle Content Management platform but have them stored on a remote server.

Each remote component must have registered settings and rendering endpoints with Oracle Content Management. In addition to endpoints, developers also need to provide a secret key unique to the registered component.

Oracle Content Management invokes registered component's endpoints to realize the content in a site page. Because these endpoints are exposed to public Internet, developers should verify that the endpoints of a registered remote component are being invoked from Oracle Content Management. For verifying the authenticity of the caller, a signed token is delivered to the registered endpoints of an URL. The calling party is authenticated by verifying the digital signature embedded in the signed token with the secret key of the remote component that was provided during the registration process.

The format of the token is:

```
{base64 encoded serialized JSON data}.{base64 encoded signature}
```

A sample token passed to the registered app endpoints follows:

```
eyJpbnN0YW5jZWlkIjoiQTRGOTE3REY5OTZEN0Q3ODBCMjUzODZFOTFEMDA3ODJGMjVBRjY
2Rjc3OTIiLCJzaWduZGF0ZSI6IjE0NDU2MzcwNTk5MTciLCJzaXRlZG9tYWluIjoic2Vydm
ljZTEtdGVuYW50MS51cy5vcmFjbGUuY29tIiwicGVybWlzc2lvbnMiOiJTSVRFX09XTkVSI
iwiZW50aXRsZW1lbnRzIjoiIn0=.5p3of7t11OwuysF3zpm+YgICSHH8C/
BHczdbVZx2VH8=
```

The token consists of two distinct parts: data and signature separated by a '.' delimiter.

As a general guideline, developers should always authenticate the token in Edit or Preview mode before granting access to registered endpoints of a component. In addition, while authenticating the calling party in the settings endpoint, developers should always take care to look for a `SITE_OWNER` value in the **permissions** field of the token. The **permissions** field of the token shows the `SITE_OWNER` value only in Edit mode. A token generated during an editing session is never persisted back to the page model and is switched out with a runtime token that has a `NULL` value in the **permissions** field.

Data

The data portion of the instance is a Base64 JSON encoded string. Here's the structure of the JSON string:

```
{
  "instanceid": "BBDC7614F693B75110D811E6C0B77C935FAEC5112E5E",
  "permissions": "",
  "entitlements": "",
  "signdate": "1435426735293",
  "sitedomain": "service1-tenant4.localhost"
}
```

Field Name	Description
`instanceid`	Unique identifier of a component for an Oracle Content Management tenant.
`signdate`	Signature generation date.
`sitedomain`	Domain name of the Oracle Content Management instance.
`permissions`	Set of permissions of the site member. In Edit mode, it will have the value `"SITE_OWNER"`; otherwise, it will have no value.
`entitlements`	List of premium features purchased by the site owner.

Signature

The data portion of the remote component instance is serialized before being signed by an `APP_SECRET_KEY`. This secret key must be generated and shown to the developer while registering the component. The signature is calculated by generating a hash of the data portion of the component instance (a serialized JSON structure) with the secret key, as shown here:

```
$signature = HMAC (serialized JSON structure, APP_SECRET_KEY)
```

The hash algorithm used in generating the signature is SHA256. The token is then the concatenation of the serialized JSON structure and the generated signature component as shown here:

```
$instance = {base64encoded serialized JSON structure}.
{base64encoded $signature}
```

Register a Remote Component

Before you can use a remote component in a site, it must be registered in Oracle Content Management.

You can register third-party remote components and those that you have developed yourself.

To register a remote component for use in your Oracle Content Management instance:

1. Click **Developer** and then click **View All Components**.
2. Click **Create** and choose **Register Remote Component**.
3. In the Register Remote Component window, enter or select information including:
   - **Name**: Name of the component that users will see.
   - **Description**: Description of the component that users will see.
   - **Component URL**: The end point used in an iframe to render component content in a page. It must be HTTPS.
   - **Settings URL**: The end point used in an iframe to render the settings of a remote component added to a page. It must be HTTPS.
   - **Settings Width**: Sets the default width of the component settings panel in pixels.
   - **Settings Height**: Sets the default height of the component settings panel in pixels.
   - **Key**: A 192–bit AES key associated with the remote component and used to create a signed hash token when the component is provisioned. It's used to encrypt and ensure component settings are read and written securely.
4. Click **Register**.

   When the remote component is created, the name appears in the list of components. You can explore the files used to register the component by clicking the component name in the list of components.

The component registration information is stored in the catalog used by sites created in the same Oracle Content Management instance, but the component remains a remote service.

As the component owner, the component icon is added to the Custom Components panel in Site Builder with the name you assigned to the component. You can share the component with other users and they will see the component in the Custom Components panel in Site Builder.

Delete a Component

If you have the appropriate permissions, you can delete a component from the component manager so it's no longer available for use. When you delete a component, the component folder and all its associated folders and files are moved to the trash.

You can delete a component from the component manager if you created the component (you're the component owner) or if someone has shared a component with you and has given you a manager role.

> **Note:**
>
> You can't delete a component if it's in use by any site or update, including sites or updates that are in the trash.

To delete a component:

1. On the home page, click **Developer**.
2. Click **View all Components**.
   Any currently registered components are displayed.
3. Select a component name and choose **Delete** in the right-click menu or click  in the actions bar.
   You're prompted to confirm your action.
4. To confirm the delete action, click **Yes**. To stop the delete action, click **No**.
   If you confirm the delete, the component and all its associated folders and files are moved to the trash.

A deleted component folder stays in the trash until:

- You restore the folder.
- You permanently delete the folder.
- Your trash quota is reached.
- The trash is automatically emptied based on the interval set by your service administrator. The default value is 90 days.

Sites SDK

Components developed for Oracle Content Management are rendered as a component in a site and can be dragged and dropped anywhere on a specific page of a site.

The Sites SDK handles all communication between the component and the page.

- `Sites.Settings.getProperty(propertyName, callbackFunction)`: Provides a callback to retrieve the requested property for the custom component instance.
- `Sites.Settings.setProperty(propertyName, propertyValue)`: Stores the requested property against the custom component instance.

Use `Sites.Settings.getProperty` or `Sites.Settings.setProperty` for all custom component properties, then use `SitesSDK.publish` and `SitesSDK.subscribe` for listening to message events.

See Sites SDK.

26

Customize the Controller File

In Oracle Content Management sites, the controller file is used to display each page in a site. When the browser sends a request for a web page, the server responds with a copy of the controller file. For every page requested, the same copy of the controller file is delivered from the server.

About the Controller File

The controller file is a small HTML page that dynamically initiates the rendering sequence for the remainder of the page. It is the first place that customizations can be applied to affect the behavior of every page of a site.

The primary task of the controller file is to provide and host an execution environment for the controller JavaScript. The controller JavaScript subsequently loads and displays the page. Customizations in the controller file provide the ability to override and influence the operation of the controller JavaScript.

Default Controller File

When an Oracle Content Management site is first created, a default controller file, controller.html, is associated with the site.

```
<!DOCTYPE html>
<html>
<head>
<!-- The following meta tag is used for Internet Explorer browsers. It
indicates that the browser should use the latest rendering mode to display
    the web page. -->
<meta http-equiv="X-UA-Compatible" content="IE=edge">

<!-- The following meta tag is used for browsers on mobile devices to set
the initial viewport scale to the full page. -->
<meta name="viewport" content="initial-scale=1">

<!-- The following script initially defines the SCS object. The SCS object
must be present, and this variable name is reserved for use by Oracle
Content Management.. -->
<script type="text/javascript">
```

```
var SCS = { sitePrefix: '/SampleSite/' };
</script>

<!-- The following script loads the full controller JavaScript, which
is used to display the web page. -->
<script src="/SampleSite/_sitescloudddelivery/renderer/controller.js"></
script>
</head>

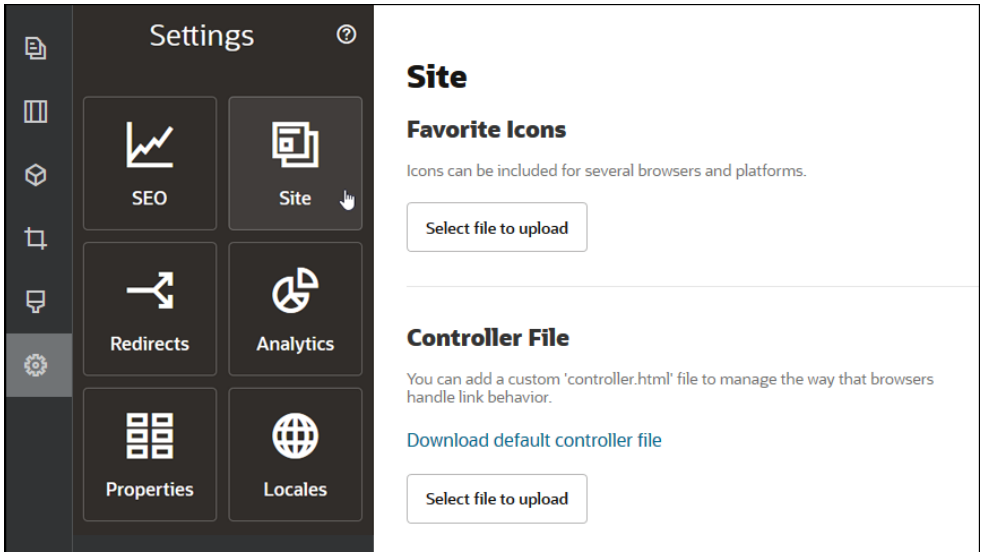
<!-- The body tag of the controller must have the id
scsControllerBody. This identifier is used by the Controller
JavaScript. If JavaScript is not enabled on the browser, the noscript
tag content is displayed.-->
<body id="scsControllerBody"><noscript>This site requires JavaScript
to be enabled.</noscript>

<!-- The following image tag displays an animated circle by default if
it takes too long for a page to display. The wait image  must have the
id scsWaitImage. -->
<img id="scsWaitImage" style="display: none; margin-top: 5%; margin-
left: auto; margin-right: auto;" src="..." />
</body></html>
```

Modify the Default controller.html File

Download a site's controller.html file and modify it to affect the behavior of each site page. To download a site's controller.html file:

1. Open the site you want to modify in Site Builder and set it to **Edit**.
2. Select an existing update or create a new one.
3. Click **Settings** in the side navigation menu and then **Site**.



4. In the Controller File section, click **Download default controller file**.
5. Save to your local drive and customize the controller.html file with your edits.

6. When done editing, return to the settings page of the site and in the Controller File section, click **Select file to upload**.
7. Navigate to the modified controller file and upload it.

> **Note:**
>
> Customizations to the controller file are only used in online sites. They will not be used when editing a site or in site previews. For the changes to take effect after uploading a custom controller.html file for a site, the site update must be committed and the site must be published.

About the SCS Object

The controller JavaScript uses variables and properties defined in the global SCS object in order to render the web page. The global JavaScript object must be defined in the controller file before the inclusion of the controller.js file. There are two primary properties in the SCS object that can be used for customization:

- SCS.sitePrefix
- SCS.preInitRendering
- SCS.getDeviceInfo

SCS.sitePrefix

The `SCS.sitePrefix` variable defines the path prefix of the online site. Normally this defaults to `site/<siteName>/`.

The controller JavaScript uses this value to determine what web page to display. For example, if the browser is requesting the page at `/site/SampleSite/products/index.html`, the sitePrefix `/site/SampleSite/` allows the JavaScript to compute that the `products/index.html` page in the site should be displayed. Note that the sitePrefix must be a string value that begins and ends with a "/" character.

If a proxy or other infrastructure such as Akamai and URL mapping rules are in place, modifying this variable allows the site prefix to be customized. For example, setting the sitePrefix to `/intranet/ExampleSite/` would allow the web site to be delivered with that pathname in the browser instead of the default `/site/SampleSite/` path prefix.

> **Note:**
>
> Internally the default controller JavaScript uses "/" as the sitePrefix if the URL in the browser does not match the site prefix. This allows the site to be served using a vanity domain without customization.

SCS.preInitRendering

If defined, the `SCS.preInitRendering` function is called by the controller JavaScript code before any of its computational logic executes. This is a useful function to define in custom controller files to override basic operations of the controller JavaScript.

If defined, `SCS.preInitRendering` must be a function. It is called without any arguments, and no return value is expected or processed.

SCS.getDeviceInfo

The `SCS.getDeviceInfo` function allows customization of the device detection logic inside the controller JavaScript. This detection is used to determine if a mobile or responsive version of a page should be delivered.

The `SCS.getDeviceInfo` function does not take any arguments, and it returns a JavaScript Object having two properties:

- **isMobile**—a Boolean property indicating that the current device is a mobile client, like a smart phone.
- **isIOS**—a Boolean property indicating that the current device is running on an iOS-based operating system.

If the custom controller does not override `SCS.getDeviceInfo`, the built-in default implementation is used.

Controller File Sections That Should Not Be Customized

Certain sections of the controller file are processed by the Oracle Content Management server in order to use the CDN and to define URL caching segments. These are noted inline in the sample below.

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="initial-scale=1">
<script type="text/javascript">
<!-- The global variable SCSCacheKeys will be inserted at this
location -->


var SCS = { sitePrefix: '/SampleSite/' };
<!-- A variable denoting the location of the CDN will be injected at
this location. -->
<!-- Additionally, the prefix "/site" will be inserted before the site
name segment. This is to support legacy controller files. -->


</script>
<script src="/SampleSite/_sitesclouddelivery/renderer/controller.js"></
script>
<!-- The src value will be updated to use the CDN if this syntax is
```

```
used. -->


</head>
<body id="scsControllerBody"><noscript>This site requires JavaScript to be
enabled.</noscript>
<!-- The id of the <body> should not be changed. -->


<img id="scsWaitImage" style="display: none; margin-top: 5%; margin-left:
auto; margin-right: auto;" src="..." />
<!-- The id of the wait image tag should not be changed -->


</body></html>
```

Use Tokens to Allow Custom Controller File Portability

When a site is created, the name of the site is used in the controller file. This ties the controller file to the named site and makes the controller file unable to be used between sites.

Using a controller file on a site with a different name than what is used in the controller file will not work. This also applies to sites created from a site template having a custom controller file. The following dynamically evaluated tokens allow the controller file to be used with multiple sites.

Token	Description

[!--$SCS_SITE_PREFIX--]	This evaluates to the site prefix for the current site. Value examples might include:
	
	`/site/MySite/`
	
	or
	
	`/site/authsite/MySecureSite/`

[!--$SCS_SITE_PATH--]	This evaluates to the current product CDN location. The value will not have a trailing '/' character. Value examples might include:
	
	`/site/MySite/_cache_0000`
	
	or
	
	`/site/authsite/MySecureSite`

Token	Description

[!--$SCS_PRODUCT_PATH--]	This evaluates to the current product URL, including cache key, if applicable. The value will not have a trailing '/' character. Value examples might include:
	
	`https://www.example.com/cdn/cec/v21.1.2.23`
	
	or
	
	`https://www.example.com/cdn/cec/v21.1.3.18`

Custom Controller File Samples

Samples are provided for the following custom control files:

- Changing the Site Prefix
- Customizing the Wait Graphic
- Customizing Favicons
- Customizing <noscript> and <meta> Tags for Non-JavaScript Crawlers
- Prefetching JavaScript Files
- Verifying Site Ownership with Additional Markup
- Augmenting Device Detection
- Using Tokens to Enhance controller.htm Portability

Changing the Site Prefix

The following sample controller file defines a `preInitRendering` function to allow the site to be delivered on multiple prefixes.

> **Note:**
>
> To make use of this sample, a CDN or other proxy must be configured to respond to the defined prefixes. Also, if none of the additional prefixes defined in the function match the browser URL, the default behavior is used.

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="initial-scale=1">
<script type="text/javascript">
var SCS = { sitePrefix: '/SampleSite/' };

SCS.preInitRendering = function() {
```

```
     // List additional site prefixes here. All sitePrefix values MUST start
and end with a '/' character.
     var additionalSitePrefixes = [
           '/corporate/intranet/SampleSite/',
           '/marketing/preflight/',
           '/qa/'
     ];

     // Determine if the actual browser URL matches one of the additional
site prefixes
     var pageUrl = decodeURI(window.location.pathname);
     var i, prefix;
     for (i = 0; i < additionalSitePrefixes.length; i++) {
           prefix = additionalSitePrefixes[i];
           if (pageUrl.startsWith(prefix) || (pageUrl === prefix.slice(0,
-1))) {
                // If we find a match, set the global site prefix variable
                SCS.sitePrefix = prefix;
                break;
           }
     }
};
</script>
<script src="/SampleSite/_sitesclouddelivery/renderer/controller.js"></
script>
</head>
<body id="scsControllerBody"><noscript>This site requires JavaScript to be
enabled.</noscript>
<img id="scsWaitImage" style="display: none; margin-top: 5%; margin-left:
auto; margin-right: auto;" src="..." />
</body></html>
```

Customizing the Wait Graphic

The following sample controller file customizes the wait graphic that is displayed for when rendering is delayed due to network delays.

> **Note:**
>
> This sample uses a data URL for the image instead of incurring a separate request during the rendering.

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="initial-scale=1">
<script type="text/javascript">
var SCS = { sitePrefix: '/SampleSite/' };
</script>
<script src="/SampleSite/_sitesclouddelivery/renderer/controller.js"></
script>
```

```
</head>
<body id="scsControllerBody"><noscript>This site requires JavaScript
to be enabled.</noscript>
<img id="scsWaitImage" style="display: none; margin-top: 5%; margin-
left: auto; margin-right: auto;" src="
M//MAP+Z//+ZzP+Zmf+ZZv+ZM/+ZAP9m//9mzP9mmf9mZv9mM/9mAP8z//
8zzP8zmf8zZv8zM/8zAP8A//8AzP8Amf8AZv8AM/8AAMz//8z/zMz/mcz/Zsz/M8z/AMzM/
8zMzMzMmczMZszMM8zMAMyZ/8yZzMyZmcyZZsyZM8yZAMxm/
8xmzMxmmcxmZsxmM8xmAMwz/8wzzMwzmcwzZswzM8wzAMwA/
8wAzMwAmcwAZswAM8wAAJn//5n/zJn/mZn/Zpn/M5n/AJnM/
5nMzJnMmZnMZpnMM5nMAJmZ/5mZzJmZmZmZZpmZM5mZAJlm/
5lmzJlmmZlmZplmM5lmAJkz/5kzzJkzmZkzZpkzM5kzAJkA/
5kAzJkAmZkAZpkAM5kAAGb//2b/zGb/mWb/Zmb/M2b/AGbM/
2bMzGbMmWbMZmbMM2bMAGaZ/2aZzGaZmWaZZmaZM2aZAGZm/
2ZmzGZmmWZmZmZmM2ZmAGYz/2YzzGYzmWYzZmYzM2YzAGYA/
2YAzGYAmWYAZmYAM2YAADP//zP/zDP/mTP/ZjP/MzP/ADPM/
zPMzDPMmTPMZjPMMzPMADOZ/zOZzDOZmTOZZjOZMzOZADNm/
zNmzDNmmTNmZjNmMzNmADMz/zMzzDMzmTMzZjMzMzMzADMA/
zMAzDMAmTMAZjMAMzMAAAD//wD/zAD/mQD/ZgD/MwD/AADM/
wDMzADMmQDMZgDMMwDMAACZ/wCZzACZmQCZZgCZMwCZAABm/
wBmzABmmQBmZgBmMwBmAAAz/wAzzAAzmQAzZgAzMwAzAAAA/
wAAzAAAmQAAZgAAM+4AAN0AALsAAKoAAIgAAHcAAFUAAEQAACIAABEAAADuAADdAAC7AACq
AACIAAB3AABVAABEAAAiAAARAAAA7gAA3QAAuwAAqgAAiAAAdwAAVQAARAAAIgAAEe7u7t3
d3bu7u6qqqoiIiHd3d1VVVURERCIiIhEREQAAACH/C05FVFNDQVBFMi4wAwEAAAAh/
h9HaWZCdWlsZGVyIDAuMy4yIGJ5IFl2ZXMgUGlndWV0ACH5BAQAAP8ALAAAAABKACAAAAj9
AFcJHEiwoMGDCBMqXMiwocOHECNKnEixosWLqwBh3HgRkEeOICN6HOkvpEmGI1f9K3my5UG
N/v79c0mTYMyVGWvqjJnxo86aIz2y/
NkyJU6iRVfdfKjRYNOCTwlGHThVYNWcNmWWvOrUJ1WvVsH2nJpSqtiyA3miRRgUEMu2b4PG
JRmW7lihdfGatcv241Gjc1UG/
ut3sGGohRXClElVKWOrjmdCXjr5cU7Kly03xnzw5tHInz0PFZ1WZmjTo033wyqQtMKYQ1t/
lh3bcW3YBXFn3QpWN9KOcn+fBCzc5GLJxStGdZ184lrfzUUGj46ROPWOka9fZK6dIvTu4AT
DMwwIACH5BAQKAP8ALAAAAAARACAAAAizAFcJHEiwoMGDCBMqXMiwYcN9q/
ZJlBjRIMSJGClCJJhxX6CMHCUG0keS5MeJAyWSvMKSJUmMAvetbGnFpT6Uq0a2vNJip76TE
PW1ZBEAQAAWNvUBCtqyKACjO/
O9FMrzqVWaUgEBonrFKVQsLKXeDDS0aIAWWARdGXmzn86dLAUJ+rmPFbh/
MrleSTv35j5+dlXqLYkzoki2P4FuNIxRMcWCHWFaNFxxscPLmDNr3syZYEAAIfkEBAoA/
wAsAAAAABkAIAAACMQAVwkcSLCgwYMIEypcyLChw4cQI0p8uK+ixX0QKwbayNFiQ40b9YkM
pC9QxYUg9V1ZyXLjSYQgWcpc6RLjQY0zc5q0WXAfoJIsWQQAEIBFS32AeAr0CSifzKEAiMr
MBygpwapN8wm60iKqV5b5qFYlyNTp1itQiWIBO7YnoECC4l4RSrQFlq0lk/
YzyKpVK7kz45YMxOrfP4P8+gpSKVOwy8SHDcbUuRPlPo4iB1dmeJHjTqWcL656ObG06dOoU
6tenTogACH5BAQKAP8ALAQAAAAdACAAAAjpAFcJHEiwoMGDCBMqXMiwocOHECNKjLhv1b6L
Fy1KrIixY8aKDjleDESSZMeHGEnqW6mvJMaGKQNdmUnzSsuXC0fKrFnzJsiEKXny9AlyHyC
DRlfSZBEAQAAWNFcC4gio6k+jgPLVbArAac18VquK9SdQbL58gmZ2XUvzrNhVYvn9I4tVa9
orXL22tWoRED9///4JTCqo8BWmTqGmbTm1n0DAgckKZNWqlWGhaUmy6if48dyB+/
hRFqSPZ+GWgUJzzrkvUGmhrjMy7FiSpUvZIXXe5ojSoseXPyHy5j2xuPHjyJMrX85cYEAAI
fkEBAoA/wAsDAAAAB0AIAAACOgAVwkcSLCgwYMIEypcyLChw4cQI0qMuG/
VvosXLUqsiLFjxooOOV4MRJJkx4cYS+pbWRJjw5SBrsiceUVfIJcLR8akSdNmRoUpefL0+X
MfIJADjdqcySIAgAAsZto8KhCQVaRGAeWj6RTAU5r5rlod66+q1Xz5BMn0ynYm2rFZAfH7V
zbrVrVXun51e9WiXH///
gk0GkiQ4StNn0ZVO3VfP4GAA5cVyKpVq8NC1d5k1U8wZLpJ+VUWpI+nYZs3+XUGOrK0UJNI
c+oMtBL1zZ8vYbb8SNGjy9ghLf6eSLy48ePIkytfPjAgACH5BAQKAP8ALBQAAAAdACAAAAj
oAFcJHEiwoMGDCBMqXMiwocOHECNKjLhv1b6LFy1KrIixY8aKDjleDESSZMeHGEvqW1kSY8
OUga7InHlFXyCXC0fGpEnTZkaFKXny9PlzHyCQA43anMkiAIAALGbaPCoQkFWkRgHlo+kUw
FOa+a5aHeuvqtV8+QTJ9Mp2JtqxWQHx+1c261a1V7p+dXvVolx///
```

```
4JNBpIkOErTZ9GVTt1Xz+BgAOXFciqVavDQtXeZNVPMGS6SflVFqSPp2GbN/
l1BjqytFCTSHPqDLQS9c2fL2G2/EjRo8vYIS3+nki8uPHjyJMrXz4wIAAh+QQECgD/
ACwcAAAAHQAgAAAI5gBXCRxIsKDBgwgTKlzIsKHDhIAeMgREUSJEihEtIsRoMdAqjwYzNgxEsiTJg
v7+jSypr6VJgv4YlrxCs+aVkhJJ2rxipeZJhzp37vw5UOTAQID0BarJIgCAACx86gMEkiNBQPny2X
QK4KnNrBExUoy5CmM+K4Jodl1b00o+jhj5/YuJNOuVtFe4eq0Jtii/lCo/
UhVE+ErTvWlJRuwnEPDcgaxatSosNHEgVv8Cr0pJViC/
yIL07SSslORnzQlnCsWZk6VL1hpNwtZ49ONBo7SvVsytezfvor5/
C8QtPKVwlMeTJwwIACH5BAQKAP8ALCQAAAAdACAAAAj9AFcJHEiwoMGDCBMqXMiwocNVgPY9VLgPk
EWJEw9a3OgvI0GJgTbu+9fP476TJ0MGYtWv5EOU+wLJDMSvVT9+FQM5RDkz0JVAgviFtKiT4qqYMq
8oXRrIipWNLhOm9Lm06hWnEf91RDjVqtd8gVr5+3cUkEGVVK+wCAAgAIulgmyykihyYEVA+aqyBdC
WKb9+dzcC2roxnxVBSvkqZirTIkTHWsvivYL4yt6+SmXGNAtx1Viyq1QKGq2WbQCliDWfFPg58lGW
/
PhVtooYLMaBY7cKjN1KUFrUggznAxToNlekvzPnG25RN8WePnuKdC31KHLoIff1+wfaKEyVqyO3c3
e+EOTGovxyexQ82GPBwKuouy/rmft8gq3Jz1d/v//8gAAh+QQEDwD/ACwsAAAAHgAgAAAI/
QBXCRS4r+A+fQgRHiw4sKFDhwYTSlTI8KHFiAmvaLwi0aBFiAcR5rMHwB6LfBwT7lu18iPBkPpIAi
iZr2ZKfRVdFkR4ZaZPjTZV5ny4E6HMkliAolTJ8mNRjSTtacTSAujNoQ2f2sRCNSqLKzWZtszKUiR
KqD6lbqTYFCTPsz19AlCaEuvLtyhbHFULVOjFkBuvsIha9UrhugP3AWq5M7Dhw3pLfqW4EpDllY0F
b2zBOfJMqUwti+7Hb59NFoU7t5A7l+0q0ftasTqo+UpS1XsFCVJIEBC/
VsBpg7X9mPPgkld04xzbD7hsmIK4RjbJ2bbu3UObz+YZeO9h5XZkMVpdLXfj0uUXM571jBRsPrZjX
9KG2+K4VCyC3rvO+vSK18JcKbefW/
r4l1ZgAqLnFGCs0UWZSwQllNtwfumE0Ug06XZVfDpNdJ2AdlkIkz7X4aQghPwt1FGIKLJk0IsMcfh
QQAA7"/>
</body></html>
```

Customizing Favicons

The following sample controller file defines a custom favicon for the site. This is useful when browsers do not dynamically load the favicon from the rendered page's Document Object Model.

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="initial-scale=1">
<link rel="shortcut icon" href="/SampleSite/favicon.ico" />
<script type="text/javascript">
var SCS = { sitePrefix: '/SampleSite/' };
</script>
<script src="/SampleSite/_sitesclouddelivery/renderer/controller.js"></
script>
</head>
<body id="scsControllerBody"><noscript>This site requires JavaScript to be
enabled.</noscript>
<img id="scsWaitImage" style="display: none; margin-top: 5%; margin-left:
auto; margin-right: auto;" src="..." />
</body></html>
```

Customizing <noscript> and <meta> Tags for Non-JavaScript Crawlers

The following sample controller file customizes the `<noscript>` message and `<meta>` tags for crawlers that do not process JavaScript.

> **Note:**
>
> Because the controller file is served for each page of the web site, the same meta tag would appear on every page.

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="initial-scale=1">
<meta name="description" content="A site with interesting content">
<script type="text/javascript">
var SCS = { sitePrefix: '/SampleSite/' };
</script>
<script src="/SampleSite/_sitesclouddelivery/renderer/controller.js"></
script>
</head>
<body id="scsControllerBody"><noscript>Please enable JavaScript to
view this site properly.</noscript>
<img id="scsWaitImage" style="display: none; margin-top: 5%; margin-
left: auto; margin-right: auto;" src="..." />
</body></html>
```

Prefetching JavaScript Files

Some browsers allow the declaration of resources needed in current or subsequent navigations. Markup in a controller file can declare resources that should be prefetched or preloaded.

> **Note:**
>
> Because every OCE web page uses require.js and renderer.js, these are good candidates for the preloading and prefetching techniques. Also, when the `href` is listed first in the `<link>` tag, its value is subject to fixup by the OCE server to be delivered from a CDN.

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="initial-scale=1">
<link href="/SampleSite/_sitesclouddelivery/renderer/require.js"
rel="preload" as="script">
<link href="/SampleSite/_sitesclouddelivery/renderer/renderer.js"
rel="preload" as="script">
<script type="text/javascript">
var SCS = { sitePrefix: '/SampleSite/' };
</script>
<script src="/SampleSite/_sitesclouddelivery/renderer/controller.js"></
```

```
script>
</head>
<body id="scsControllerBody"><noscript>Please enable JavaScript to view this
site properly.</noscript>
<img id="scsWaitImage" style="display: none; margin-top: 5%; margin-left:
auto; margin-right: auto;" src="..." />
</body></html>
```

Verifying Site Ownership with Additional Markup

Third Party crawlers and search engines may require additional markup in order to verify site ownership. This can be injected into the controller file as needed.

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="initial-scale=1">
<meta name="search-site-verification" content="your verification string">
<script type="text/javascript">
var SCS = { sitePrefix: '/SampleSite/' };
</script>
<script src="/SampleSite/_sitesclouddelivery/renderer/controller.js"></
script>
</head>
<body id="scsControllerBody"><noscript>Please enable JavaScript to view this
site properly.</noscript>
<img id="scsWaitImage" style="display: none; margin-top: 5%; margin-left:
auto; margin-right: auto;" src="..." />
</body></html>
```

Augmenting Device Detection

The recognition of new mobile devices or customized device user-agent identifiers can be accomplished by customizing the device detection code in the controller.

Note:

- The `getDeviceInfo` function is called by the controller to determine if a mobile device is being used. This sample overrides the default getDeviceInfo call.
- The isMobile property determines which layout will be used when rendering the page.
- The isIOS property determines which provider is used with the Map component.
- The built-in default `isMobile` test is

  ```
  /Mobi|iPhone|iPod|BlackBerry|IEMobile|Opera Mini/
  i.test(userAgent) && !/iPad/i.test(userAgent)
  ```

- The built-in default `isIOS` test is

  ```
  /iPad|iPhone|iPod/i.test(userAgent) && !window.MSStream
  ```

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="initial-scale=1">
<meta name="search-site-verification" content="your verification
string">
<script type="text/javascript">
var SCS = { sitePrefix: '/SampleSite/' };


SCS.getDeviceInfo = function() {
    // Return an object with two Boolean properties, isMobile and
isIOS.
    var userAgent = navigator.userAgent;
    return {
        isMobile: /Mobi|iPhone/i.test(userAgent) && !/iPad/
i.test(userAgent),
        isIOS: /iPad|iPhone|iPod/i.test(userAgent)
    };
};
</script>
<script src="/SampleSite/_sitesclouddelivery/renderer/controller.js"></
script>
</head>
<body id="scsControllerBody"><noscript>Please enable JavaScript to
view this site properly.</noscript>
<img id="scsWaitImage" style="display: none; margin-top: 5%; margin-
left: auto; margin-right: auto;" src="..." />
</body></html>
```

Using Tokens to Enhance controller.htm Portability

By default the site name is hard-coded into the controller.html files, making it difficult to rename a site or to reuse identical custom controllers across multiple sites. Tokens can be used to alleviate this issue. The following sample shows how portability tokens can be used.

> **Note:**
>
> The the `<link>` tags in the sample are for demonstration only. The baseline controller.html would not include these.

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="initial-scale=1">
<script type="text/javascript">
var SCS = { sitePrefix: '[!--$SCS_SITE_PREFIX--]' };
</script>
<script src="[!--$SCS_PRODUCT_PATH--]/_sitesclouddelivery/renderer/
controller.js"></script>
<link rel="shortcut icon" href="[!--$SCS_SITE_PATH--]/favicon.ico">
<link rel="preload" href="[!--$SCS_PRODUCT_PATH--]/_sitesclouddelivery/
renderer/require.js" as="script">
<link rel="preload" href="[!--$SCS_PRODUCT_PATH--]/_sitesclouddelivery/
renderer/renderer.js" as="script">
</head>
<body id="scsControllerBody"><noscript>Please enable JavaScript to view this
site properly.</noscript>
<img id="scsWaitImage" style="display: none; margin-top: 5%; margin-left:
auto; margin-right: auto;" src="..." />
</body></html>
```

Part V

Developing for Sites with Other Tools

Oracle Content Management provides several ways to interact with applications and experiences developed either inside Oracle Content Management or outside of Oracle Content Management using other 3rd party tools.

Using Oracle Content Management Toolkit

The Oracle Content Management Toolkit and SDKs help you develop custom applications that consume content that is managed in the Oracle Content Management repository. These applications can be developed in Oracle Content Management or using 3rd party tools.

Content Toolkit can help you do the following tasks:

- Set up your local development environment to use an Oracle Content Management instance for local development and testing of components, templates, themes, and content layouts
- Create components, site templates, and content layouts from samples, run them in the test harness, explore them, and develop the components, templates, themes, and content layouts in a Developer Cloud Service environment
- Import components and site templates that were created in Oracle Content Management into a Developer Cloud Service project and environment for source management and further development
- Export components, templates, and content layouts from a Developer Cloud Service environment for use in Oracle Content Management
- Copy an existing component, template, or content layout
- Write unit tests
- Optimize components
- Deploy your components and templates to Oracle Content Management

Additional information can be found at https://github.com/oracle/content-and-experience-toolkit#readme.

Experience Orchestration

If you use tools other than Oracle Content Management to create experiences, you can connect Oracle Content Management repositories to these experiences so content creators can preview site changes as they work and automatic builds can be triggered when content changes or is published. This experience orchestration automates the workflow process between content providers and site developers to simplify experience management and publication.

27

Develop with Oracle Content Management Toolkit

Oracle Content Management Toolkit helps you develop site templates, themes, custom components, and content layouts for Oracle Content Management.

With Content Toolkit, you work in your own development environment and can use asset repositories, files, and folders in Oracle Content Management. Content Toolkit has tools to create and develop custom components and site templates, including themes and content layouts. It includes a local test harness for quick, iterative development and sample unit tests to start with.

The following topics describe how to set up Content Toolkit and develop with it on your local machine or as a Developer Cloud Service project:

Set Up Oracle Content Management Toolkit on Your Local Machine

To set up the Oracle Content Toolkit on your local machine, you'll need to make sure you have Node.js and Node packet manager (npm) installed. Then, follow the set up instructions to set your path, download and unpack the Content Toolkit from GitHub, install dependencies, and create a local source folder outside of the Content Toolkit hierarchy to store your work. For more information, review the README.md file on GitHub.

Complete the setup and prepare to use Content Toolkit:

1. Install Dependencies Through Node Package Manager
2. Use the cec Command-Line Utility
3. Test with a Local Test Harness

Install Dependencies Through Node Package Manager

After you've set your paths and installed Content Toolkit from GitHub, use node package manager (npm) to install sites dependencies.

1. Navigate to the sites directory of your Oracle Content Management Toolkit installation.
2. Run `npm install`

If you are using a proxy to access the internet, set the proxy for npm with the `npm config` command. See https://docs.npmjs.com/misc/config. To set the proxy for bower, see https://bower.io/docs/config/.

> **Note:**
>
> Ensure that you have Node.js 14.0.0 or later (https://nodejs.org/) installed on your local computer.

Use the cec Command-Line Utility

The Oracle Content Toolkit comes with a cross-platform `cec` command-line utility, which allows you to execute commands for many aspects of Oracle Content Management.

Before you use the `cec` command-line utility, you must create a **local source folder** in a different location from where you installed the Oracle Content Management Toolkit. This source folder is where all of your local work will be stored, and where you will run `cec` commands. The source folder can be named anything you want, but for most examples in this guide, the source folder is named `cec`. For example:

```
mkdir cec
cd cec
cec install
```

For a complete list of all cec commands, along with descriptions and examples, see Oracle Content Toolkit Command-Line Utility in the Oracle Help Center.

Alternatively, you can also open a terminal window, go to your source directory, and type `cec` to see a list of all commands or `cec <command> -h` to see help for specific commands.

> **Note:**
>
> When using a private instance the Oracle Content Toolkit must be installed and updated from a location with open access to the internet as various required dependencies need to be downloaded from domains outside of Oracle Content Management. After the Oracle Content Toolkit has been installed or updated you can run it from a private instance.

Test with a Local Test Harness

Run your custom components, templates, and content layouts in a local test harness before importing them to Oracle Content Management.

To start the local test harness:

1. Enter `cd cec` in a terminal window.
2. Enter `cec develop &` or `cec develop --server <server-name> &`
3. Open a browser at `http://localhost:8085` to see your components, templates, and content layouts running in the local test harness.
4. You can find your components, templates, themes, and so on, in these directories:
   - `cec/src/components`
   - `cec/src/templates`
   - `cec/src/themes`

Upgrade to jQuery 3.5.x

As a Developer, you can use Oracle Content Management Toolkit to identify improperly closed html tags.

Upgrade to jQuery 3.5.x to pick up a security fix in the HTML Parser.

https://blog.jquery.com/2020/04/10/jquery-3-5-0-released

Parsing certain HTML strings in jQuery 3.5.x will give different results than parsing the same strings in 3.4.x. These strings involve tags that are self-closed, in violation of the HTML standard; for example: `"<div />"`. Such strings, especially when part of a larger sequence of tags, might be parsed differently in 3.5.x compared to 3.4.x.

You can run `cec create-asset-report <site>` to find improperly closed html tags in page JSON files and component HTML and JS files.

Develop for Oracle Content Management with Developer Cloud Service

Oracle Content Management Toolkit, the Developer Cloud Service integration, helps you develop site templates, themes, custom components, and content layouts for Oracle Content Management.

With Content Toolkit, you can use asset repositories, files, and folders in Oracle Content Management. Content Toolkit has tools to create and develop custom components and site templates, including themes and content layouts. It includes a Git repository as well as a local test harness for quick, iterative development, and sample unit tests to start with.

Content Toolkit can help you do the following tasks:

- Set up your local development environment to use an Oracle Content Management instance for local development and testing of components, templates, themes, and content layouts
- Create components, site templates, and content layouts from samples, run them in the test harness, explore them, and develop the components, templates, themes, and content layouts in a Developer Cloud Service environment
- Import components and site templates that were created in Oracle Content Management into a Developer Cloud Service environment for source management and further development
- Export components, templates, and content layouts from a Developer Cloud Service environment for use in Oracle Content Management
- Copy an existing component, template, or content layout
- Write unit tests
- Optimize components
- Deploy your components and templates to Oracle Content Management

The following topics describe how to set up Developer Cloud Service for developing custom components, site templates and themes, and content layouts:

1. About Using Developer Cloud Service
2. Sign In to the Developer Cloud Service Console for Oracle Content Management
3. Create a Project in Developer Cloud Service
4. Add Oracle Content Management Toolkit to the Project Code in the New Git Repository
5. Test Custom Components, Templates, and Content Layouts in a Local Test Harness
6. Merge Changes

The following topics provide information about using the Oracle Content Management Toolkit:

- Use the cec Command-Line Utility
- Develop Custom Components with Developer Cloud Service
- Develop Templates with Developer Cloud Service
- Develop Content Layouts

About Using Developer Cloud Service

Oracle Developer Cloud Service is a cloud-based software development Platform as a Service (PaaS) and a hosted environment for your application development

infrastructure. It provides an open source standards-based integration to develop, collaborate, and deploy applications within Oracle Cloud.

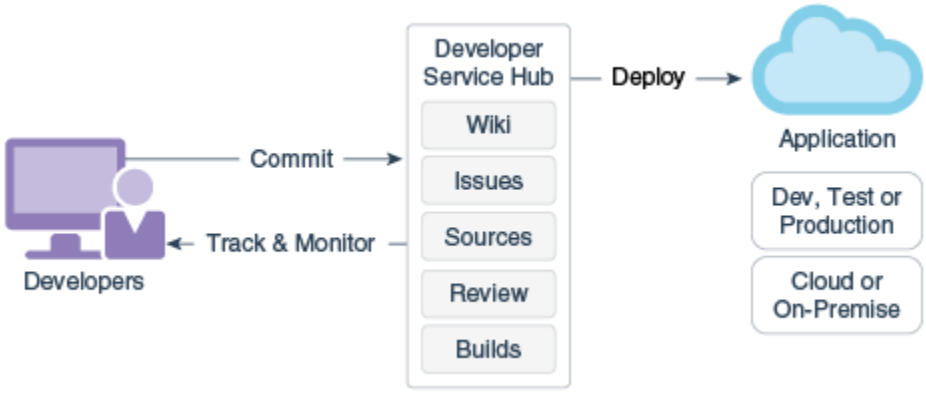
Developer Cloud Service is a collection of software and services hosted on Oracle Cloud to help you manage the application development life cycle effectively through integration with Git, Maven, issues, and wikis. Using Oracle Developer Cloud Service, you can commit your application source code to the Git repository on Oracle Cloud, track assigned issues and defects online, share information using wiki pages, peer review the source code, and monitor project builds. After successful testing, you can deploy the project to Oracle Content Management.

Sign in to the Developer Cloud Service Console for Oracle Content Management

Start developing your custom components for Oracle Content Management on the Developer Cloud Service console.

As an administrator for Oracle Cloud services, you can use My Service Administration to create and manage your Cloud services. If you're a service instance administrator for Oracle Content Management and a service administrator for Standard Developer Service, you can set them up and start using them:

1. Sign in to Oracle Cloud, using the information that was provided for your account.
2. Sign in to My Service Administration to create and manage your Oracle Content Management instance and your Standard Developer Service.



3. Verify your Oracle Developer Cloud Service email, as requested.
4. Set up you Oracle Content Management instance, using the subscription details for your service, and go to the Oracle Content Management URL for your instance.
5. Go to your URL for the Standard Developer Service.
6. Sign in to your Oracle Developer Cloud Service account.

Access the Developer Cloud Service URL and sign in to the console.

Create a Project in Developer Cloud Service

You can create a project in Developer Cloud Service using the "Content Experience Cloud" project template, or you can create a project with an empty Git repository and import the Content Toolkit from your Oracle Content Management instance.

- Create a Developer Cloud Service Project with an Oracle Content Management Template
- Create a Project in Developer Cloud Service with a Content Toolkit Download from Oracle Content Management
- Add Oracle Content Management Toolkit to the Project Code in the New Git Repository

Create a Developer Cloud Service Project with an Oracle Content Management Template

Create a project for developing custom components, templates, themes, and content layouts in Developer Cloud Service.

To create a project:

1. After you sign in to the Developer Cloud Service console, click **New Project**.
2. In the list of templates, choose **Content Management**, and then click **Next**.
3. In the properties under Project Properties, choose **CONFLUENCE** in the **Wiki Markup** field.

Create a Project in Developer Cloud Service with a Content Toolkit Download from Oracle Content Management

Create a project for developing custom components, templates, themes, and content layouts in Developer Cloud Service.

To create a project:

1. After you sign in to the Developer Cloud Service console, click **New Project**.
2. Name your project, enter or select other project details you want, and then click **Next**.
3. In the list of templates, choose **Initial Repository**, and then click **Next**.
4. In the properties under Project Properties, choose `Empty Repository` for the Initial Repository. Click **Finish**.

Add Oracle Content Management Toolkit to the Project Code in the New Git Repository

You can add Oracle Content Management Toolkit to the new, empty Git repository for your project.

1. Under **REPOSITORIES** in your new project, copy the HTTP URL of the project Git repository.

2. Open a terminal window and enter this command: `git clone <your-project >.git`
   a. When asked, enter your password for Developer Cloud Service.
   b. If you see the error "`git is not a command`", install Git from `https://git-scm.com/downloads`, and then reenter the `git clone` command.
3. `git clone git@github.com:oracle/content-and-experience-toolkit.git`

   Or you can download from here: `https://github.com/oracle/content-and-experience-toolkit/archive/master.zip`
4. `cp -R content-and-experience-toolkit/sites/cec-components <your-project>`
5. cd <your-project>
6. `git add cec-components`
7. `git commit -a -m "<your comments>"`
8. `git push`

Test Custom Components, Templates, and Content Layouts in a Local Test Harness

Run your custom components, templates, and content layouts in a local test harness before importing them to Oracle Content Management.

To start the local test harness:

1. Enter `cd cec-components` in a terminal window.

   Enter `npm start &`
2. Open a browser at `http://localhost:8085` to see your components, templates, and content layouts running in the local test harness.

When you test components on your local server, you can choose to use content from a local template or from the Oracle Content Management server.

Merge Changes

After you create a component, site template, or content layout or edit source code on your local machine, you need to merge new and changed components and site templates into your project's Git repository.

To merge changes into your Git repository, enter the following commands, in order, in a terminal window.

```
cd cec-components git pull
git add .
git status
git commit -a -m "Your comments" git pull
git push
```

Propagate Changes from Test to Production with Oracle Content Management Toolkit

After you develop a site template, you can use the command-line interface (CLI) of Content Toolkit to propagate the template from development to test to production on your Oracle Content Management servers.

To propagate changes, you can use Toolkit commands to create sites and manage their life cycles on development, test, and production servers. You can make changes to sites in a development environment and propagate those changes to test and production environments. You can also incorporate this set of command-line utilities into your scripting environments to manage your deployments. With the CLI utilities, you can roll out new items, such as assets and components, as well as updates of existing content.

The following steps show how to use the Content Toolkit CLI to propagate your changes from development to test to production:

1. Set up development, test, and production servers with the same repository and localization policy.

   To propagate changes from a development server to a test server and then to a production server, you need to set up a repository with the same name and localization policy on each of the three servers. The default localization policy for the asset repository is en-US, but you can use a different one if it is the same in all three servers.

   See Set Up Asset Repositories.

2. Register your development, test, and production servers with Oracle Content Management.

   Before you propagate changes to a site, you need to register each of the servers. You can register a server with the `cec register-server` command provided by Content Toolkit:

   ```
   cec register-server <name>
   ```

   Specify the following command options:

   - -e <endpoint> for the server URL
   - -u <user> and -p <password> for connecting to the server
   - -t <type>, which is optional, to set the server type. The default value is `pod_ec`.

     When connecting to an Oracle Content Management tenant on Oracle Public Cloud, use `pod_ec` only.

   For example, the following command registers a server that is a tenant on Oracle Public cloud:

   ```
   cec register-server DEV -e https://DEV.example.com -u user1 -p
   <password>
   ```

The next command registers a standalone development instance of Oracle Content Management:

```
cec register-server DEV -e https://DEV.git.oraclecorp.example.com -u
user1 -p <password>
```

After you register an Oracle Content Management server, you can list its contents with the `cec list` Toolkit command.

The following command lists the contents of a development server:

```
cec-compontents> cec list -s DEV
 - Logged in to remote server: <host:port>
Channels:
  Name                          Token
  StarterSite                   <site-id>

Components:
  Name                          Type                      Published
  FooterBar                     Component group
  StarterComponent              Local component
  StarterFooter                 Component group
  StarterNavMenu                Local component

Localization policies:
  Name                          Required languages        Optional
Languages
  en-US                         en-US

Repositories:
  Name
  r

Sites:                          Theme                     Type
Published   Online
  Name                          StarterSiteTheme          Enterprise

Templates:
  Name                          Theme                     Type
  StarterTemplate               StarterTheme              Standard
```

3. Upload a site template to your development server and create a site from the template.

   You can create a site template with the `cec create-template` command and then upload the template to the development server. Then you can create a site from the template with the `cec create-site` command. The following commands create a template and upload the template:

```
- cec create-template blog -f BlogTemplate
- cec upload-template blog -s DEV
```

The next command creates a site named `blog` from the uploaded template:

```
cec-components> cec create-site blog -t blog -r r -l "en-US" -d "en-
US" --server DEV
 - Logged in to remote server: <https:<host:<port>
 - establish user session
 - get template
 - get repository
 - get localization policy
 - creating enterprise site . . .
   name                 blog
   template             blog
   site prefix          blog
   repository           r
   localization policy  en-US
   default language     en-US
 - submit create site site
 - create site in process: percentage 95
 - create site in process: percentage 95
 - create site in process: percentage 95
 - create site in process: percentage 95
 - create site in process: percentage 95
 - site created
```

4. Publish the site and bring it online on your development server.

   After you create a site, you can use the `cec control-site` command to publish the site and bring it online:

```
cec-components> cec control-site
Usage: cec contrl-site <action>
Perform <action> on site in CEC server. Specify the site with -s
<site> Specify the server with -r <server>.

  publish
  unpublish
  bring-online
  take-offline

Options:
  --site, -s    <site> Site
  --server, -r  <server> The registered CEC server
  --help, -h    Show help

Examples:
  cec control-site publish -s Site1              Publish site Site1
on the server
  cec control-site publish -s Site1 -r UAT       Publish site Site1
on the registered server UAT
  cec control-site unpublish -s Site1 -r UAT     Inpublish site
Site1 on the registered server UAT
  cec control-site bring-online -s Site1 -r UAT  Bring site Site1
online on the registered server UAT
  cec control-site take-offline -s Site1 -r UAT  Take site Site1
offline on the registered server UAT
```

```
Not enough non-option arguments: got 0, need at least 1
cec-components> cec control-site publish --site blog --server DEV
 - Logged in to the remote server: https://<host>:<port>
 - establish user session
 - get site: runtimeStatus: offline publishStatus: unpublished
 - submit publish site
 - publish in process: percentage 20
 - publish in process: percentage 40
 - publish in process: percentage 49
 - publish in process: percentage 49
 - publish in process: percentage 50
 - publish in process: percentage 50
```

5. To move a site from DEV to UAT, you need to package up the site. The packaging model for moving sites between servers is the template. Create a new template from the site you created on your development server and download the template.

   The `cec create-template-from-site` command in the following example creates a site template named `blog2` from `blog`.

   ```
   cec create-template-from-site blog2 -s blog
   ```

   Download the template you created from your development site with the `cec download-template` command:

   ```
   cec-components: cec download-template blog2 --server DEV
    - Logged in to remote server: https://<host>:<port>
    - establish user session
    - export template
    - template download to /Users/<user-name>/devenv/git/webclient/developer/
   sites-toolkit/cec-components/dist/blog2.zip
    - the template will be at /Users/<user-name>/devenv/git/webclient/
   developer/sites-toolkit/cec-components/src/main/templates/blog2
    - the theme for the template will be at /Users/<user-name>/devenv/git/
   webclient/developer/sites-toolkit/cec-components/src/main/themes/blogTheme
    - create link _scs_theme_root_
    - create link _scs_design_name_
    - override component /Users/<user-name>devenv/git/webclient/developer/
   sites-toolkit/cec-components/src/main/components/Starter-Blog-Author-
   Summary
    - override component /Users/<user-name>devenv/git/webclient/developer/
   sites-toolkit/cec-components/src/main/components/Starter-Blog-Post-Content
    - override component /Users/<user-name>devenv/git/webclient/developer/
   sites-toolkit/cec-components/src/main/components/Starter-Blog-Post-Header
    - override component /Users/<user-name>devenv/git/webclient/developer/
   sites-toolkit/cec-components/src/main/components/Starter-Blog-Post-Search-
   Result
    - override component /Users/<user-name>devenv/git/webclient/developer/
   sites-toolkit/cec-components/src/main/components/Starter-Blog-Post-Post-
   Sidebar
    - override component /Users/<user-name>devenv/git/webclient/developer/
   sites-toolkit/cec-components/src/main/components/Starter-Blog-Post-Summary
    - set themeName to blogTheme in siteinfo.json
   ```

```
 - unzip tmplate content file
 *** template is ready to test: https://localhost:8085/templates/
blog2
cec upload-template blog2 --server UAT
```

6. Upload the template to create the content types and content layout maps.

```
cec upload-template blog2 --server UAT
```

7. Upload the template but exclude the content items (content template) from the template.

```
cec upload-template blog2 --server UAT -x
```

   You want to do this to create a site with content that has the same GUIDs as the original site. When you create a site from a template that contains content, all the content in the new site will have new GUIDs. Because we want to allow update of content rather than creating new content, you need to exclude the content from the template.

8. Create the site from the template.

```
cec create-site blog -t blog2 -r r -l "en-US" -d "end-US" --server
UAT
```

9. Upload the content template to the site's channel and collection. You need to do this because you excluded it from the template in step 7.

```
cec upload-content blog2 -t -r r -c blog -l "blog site" --server UAT
```

10. Publish the site and bring it online on your test server.

    Use the `cec control-site` command to publish the site and bring it online:

```
cec-components> cec control-site publish --site blog --server UAT
 - Logged in to the remote server: https://<host>:<port>
 - establish user session
 - get site: runtimeStatus: offline publishStatus: unpublished
 - submit publish site
 - publish in process: percentage 20
 - publish in process: percentage 40
 - publish in process: percentage 49
 - publish in process: percentage 49
 - publish in process: percentage 50
 - publish in process: percentage 50
 - publish blob finished
```

11. If you then make changes to your site `blog` on the DEV server, you can propagate the changes to the site you have already created on the UAT server.
12. Create another template from your site so the template will have your changes.

```
cec create-template-from-site blog3 -s blog --server DEV
```

13. Download the template.

```
cec download-template blog3 -s DEV
```

14. Upload the template and create a site from it to propagate the changes to your test environment.

```
cec upload-template blog3 -s UAT
```

This command creates or updates any components and themes that have changed, but excludes the content.

15. Now use the `update-site` command to pick up the content and update the pages.

```
cec update-site blog -t blog3 - UAT
```

For example:

```
cec-components> cec update-site blog -t blog3 --server UAT
Updating site: blog3
 - Logged in to remote server: https://<host>:<port>
 - pages               : updating file# 6    of 6    files
 - content             : updating file# 3    of 3    files
 - System Files        : updating file# 5    of 5    files
 - controller          : no files in update, removing files on server
 - favicons            : no files in update, removing files on server
 - misc                : no files in update, removing files on server
 - seo                 : no files in update, removing files on server
 - system              : no files in update, removing files on server
 - created content file /Users/<user-name>/devenv/git/webclient/developer/
sites-toolkit/cec-compnents/dist/blog3_export.zip
 - upload content file
 - get CSRF token
 - submit import job, updating content
 - import job in progress. . .
 - import job in progress. . .
 - import job in progress. . .
 - content imported:
Update Site Results:
 - Site Pages          : completed with 0 errors.
 - Embedded Content    : completed with 0 errors.
 - System Files        : completed with 0 errors.
 - Settings Files      : completed with 0 errors.
 - Content Update      : completed with 0 errors.
```

16. Check the site to verify that the changes were propagated.
17. Do the same steps to move from the UAT server to the PROD server as you did to move the site from DEV to UAT.
18. Create the site on your production server, bring it online, and verify the changes.

    You can use the `cec list` command to view the contents of your production site and make sure it includes the changes you made in the development environment. Also, you can check the site to verify that the changes have been propagated to production.

Encrypt a Password

When you register a server with Content Toolkit, you need to encrypt a password to make the server available for local use.

1. Register an Oracle Content Management server with a `cec register-server` command that includes the password in plain text.
2. Encrypt the password with the `cec create-encryption-key` command.

```
  cec create-encryption-key <file>           [alias:  cek]
      Create an encryption key to encrypt/decrypt password for
servers.
```

3. Register the server again with the encryption key, which makes the server available for local development and testing.

Encrypted passwords are stored in the server connection file. A password is decrypted when you connect to a registered server.

Register a Server

You can register a server in Content Toolkit.

Use the `cec register-server` command with an encryption key to register a Oracle Content Management server for local development and testing.

When you register the server, encrypting a password makes the server available for use with Content Toolkit. See Encrypt a Password.

Create a Usage and Permission Report for a Site

You can create a report to validate and fix permissions of target server members for test-to-production propagation for a site.

Use the cec `create-asset-report` and check it as follows:

1. Check the membership and channel assignment of all site artifacts:
   - Theme
   - Template
   - Components
   - Content type
2. Flag issues that you can address.

For example:

```
cec create-asset-report blog1 -s <registered-server> -o

cec create-asset-report trbcent -s <registered-server> -o
```

The report is generated in a JSON file, which you can check for problems with usage and permissions. The following commands are available for permission fixes:

- `cec share-type`: Share types with users in an Oracle Content Management server.
- `cec unshare-type`: Remove access to types for given users in an Oracle Content Management server.
- `cec share-repository`: Share a repository (and the types used by the repository) in an Oracle Content Management server.
- `cec unshare-repository`: Remove user access to a repository in an Oracle Content Management server.
- For example:

```
cec share-repository Repo1 -u <user-name1>,<user-name2> -r manager -t -s
<registered-server>
```

Download and Upload Documents and Folders

You can download and upload documents and folders to and from an Oracle Content Management server.

The following commands are available for document and folder downloads and uploads:

```
  cec download-folder <path>          Downloads folder from CEC
server.              [alias: dlfd]
  cec upload-folder <path>            Uploads folder to CEC
server.                   [alias: ulfd]
  cec download-file <file>            Downloads file <file> from CEC
server.         [alias:  dlf]
  cec upload-file <file>              Uploads file <file> to CEC
server.             [alias: ulf]
  cec-share-folder <name>
  cec-unshare-folder <name>
```

For `cec-share-folder <name>`, you can share a folder with users on an Oracle Content Management server and assign users a role. Specify the server with -s <server>, or use the server specified in the `cec.properties` file. The valid roles follow:

- `manager`
- `contributor`
- `downloader`
- `viewer`

For downloads, you can specify a folder hierarchy.

Create a Site from a Template and Keep the Same GUIDs for Content

As a developer, you can use an Oracle Content Management Toolkit command to create an Oracle Content Management site from a template, keeping the same GUIDs for content.

Use the following Content Toolkit command:

```
update create-site-from-template --reuse-content
```

This command creates a site in an Oracle Content Management server and preserves content IDs when creating the site. Preserving content IDs is necessary for multiple test to production runs to not end up with duplicate content items on a target server.

Create an Enterprise Template from a Standard Site

As a developer, you can use an Oracle Content Management Toolkit CLI command to create an enterprise template from a standard site.

By default, the `create-template` command creates a standard template if the site is a standard site and an enterprise template if the site is an enterprise site. You can create an enterprise template from a standard site as well.

Run `cec create-template` with the new `-enterprise` option:

```
cec create-template EnterpriseTemp1 -s StandardSite1 -e
```

Import and Export Taxonomies

Use Oracle Content Management Toolkit commands to import taxonomies from your local machine to an Oracle Content Management server or to export taxonomies from the server to your local machine.

The `cec download-taxonomy <name>` command exports a taxonomy from Oracle Content Management. It downloads a taxonomy from an Oracle Content Management server.

You can use the following options in this command:

- `--status, -t [promoted | published] [required]`: Specify the taxonomy status.
- `--id, -i`: If another taxonomy has the same name, specify the taxonomy ID.
- `--server, -s`: Specify a registered Oracle Content Management server, or use the one specified in the `cec.properties` file.
- --help, -h: Show help for the command.

Some examples of the `download-taxonomy` command follow:

```
cec download-taxonomy Taxonomy1 -t promoted

  cec download-taxonomy Taxonomy1 -i 6A6DC736572C468B90F2A1C17B7CE5E4 -
t promoted

  cec download-taxonomy Taxonomy1 -t published -s UAT
```

The `cec upload-taxonomy <taxonomy>` command imports a taxonomy to Oracle Content Management. It uploads a taxonomy to an Oracle Content Management server.

You can use the following options in this command:

- `--createnew, -c`: Create a new taxonomy.
- `--name, -n`: Specify the name of the new taxonomy.
- `--abbreviation, -a`: Specify an abbreviation for the new taxonomy.
- `--description, -d`: Specify a description of the new taxonomy.
- `--file, -f`: Indicate whether the taxonomy is from a file.
- `--server, -s`: Specify a registered Oracle Content Management server, or use the one specified in the `cec.properties` file.
- `--T2P`: Checks if an asset or taxonomy with the same ID exists on the target Oracle Content Management instance. If true, the asset is added as a new version and the taxonomy as a draft (overriding the existing draft). Otherwise, a new asset or taxonomy is created with the same IDs.
- `--New`: Always creates a new asset or taxonomy on the target Oracle Content Management instance.
- `--help, -h`: Show help for the command.

Some examples of the **upload-taxonomy** command follow:

```
cec upload-taxonomy Taxonomy1
Create a new taxonomy or a draft of an existing taxonomy on upload

cec upload-taxonomy Taxonomy1 -s UAT
Create a new taxonomy or a draft of an existing taxonomy on upload on the
registered server UAT

cec upload-taxonomy Taxonomy1 -c
Create a new taxonomy on upload

cec upload-taxonomy Taxonomy1 -c -n Taxonomy1_2 -a t12 -d
Create a new taxonomy on upload with the given name, abbreviation

"Taxonomy1 copy" and description cec upload-taxonomy
Create a new taxonomy or a draft of an existing taxonomy in <file-name>.json
-f and upload the JSON file
```

You can use Content Toolkit test to production CLI utilities to automate the import or export of assets together with a content model and its dependencies from a source Oracle Content Management server to a target Oracle Content Management server.

A manager or content administrator can import or export a taxonomy with Content Toolkit commands in a test to production environment. With manager permission, you can add a draft of a taxonomy. With content administrator permission, you can create a new taxonomy.

Taxonomy lifecycle operations, such as promote, assign to a repository, and publish taxonomy are available for test to production. You can import assets together with categorization information and taxonomies into an import file, and you can export assets together with categorization information and taxonomies from an export file.

Import and Export Recommendations

Use Oracle Content Management Toolkit commands to import recommendations from an Oracle Content Management server to your local machine or to export recommendations from your local machine to the server.

The `cec download-recommendation <name>` command exports a recommendation from Oracle Content Management. It downloads the recommendation from an Oracle Content Management server.

You can use the following options in this command:

- `--status, -t [promoted | published] [required]`: Specify the recommendation status.
- `--id, -i`: If another recommendation has the same name, specify the recommendation ID.
- `--server, -s`: Specify a registered Oracle Content Management server, or use the one specified in the `cec.properties` file.
- --help, -h: Show help for the command.

Some examples of the `download-recommendation` command follow:

```
cec download-recommendation Recommendation1 -t promoted

  cec download-recommendation Recommendation1 -i
6A6DE836572C468B90F2A1C17B7CE5E4 -t promoted

  cec download-recommendation recommendation -t published -s UAT
```

The `cec upload-recommendation <name>` command imports a recommendation to Oracle Content Management. It uploads a recommendation to an Oracle Content Management server.

You can use the following options in this command:

- `--createnew, -c`: Create new a recommendation.
- `--name, -n`: Specify the name of the new recommendation.
- `--abbreviation, -a`: Specify an abbreviation for the new recommendation.
- `--description, -d`: Specify a description of the new recommendation.
- `--file, -f`: Indicate whether the recommendation is from a file.
- `--server, -s`: Specify a registered Oracle Content Management server, or use the one specified in the `cec.properties` file.
- `--help, -h`: Show help for the command.

Some examples of the **upload-recommendation** command follow:

```
cec upload-recommendation Recommendation1
Create a new recommendation or a draft of an existing recommendation
on upload
```

```
cec upload-recommendation Recommendation1 -s UAT
Create a new recommendation or a draft of an existing recommendation on
upload on the registered server UAT

cec upload-recommendation Recommendation1 -
c
Create a new recommendation on upload

cec upload-recommendation Recommendation1 -c -n Recommendation 1_2 -a t12 -
d
Create a new recommendation on upload with the given name, abbreviation

"Recommendation1 copy" and description cec upload-recommendation
Create a new recommendation or a draft of an existing recommendation in
<file-name>.json -f and upload the JSON file
```

Add or Remove Collection Content

As a Developer, you can add content to collections and remove content from collections with Oracle Content Management Toolkit commands.

The following `cec-control-content` command adds all items in the **Repo1** repository to the **Collection1** collection on the **UAT** registered server:

```
cec control-content add -l Collection1 -r Repo1 -s UAT
```

The following `cec-control-content` command removes all items from the **Collection1** collection on the **UAT** registered server:

```
cec control-content remove -l Collection -s UAT
```

Develop Custom Field Editors Using the Oracle Content Management Toolkit

Oracle Content Management Toolkit provides support for developing components of the Field Appearance type. Developers can create and manage custom field editors.

For a component of the Field Appearance type, you can do the following tasks:

- Open, copy, or delete the component
- Publish or unpublish the component
- Export or import the component
- Add or remove members on the component
- View properties
- Choose the component logo

You can filter a list of components by the Field Appearance type.

The following Content Toolkit commands are available for developing custom field editors:

```
cec add-field-editor <name>                          Adds a field editor
to a field in a content type.                         [alias:  afe]
  cec remove-field-editor <name>                      Removes a field
editor from a field in a content type.                   [alias:  rfe]

------------------------
cec add-field-editor
------------------------
Usage: cec add-field-editor <name>

Adds a field editor to a field in a content type.

Options:
  --template, -t         The template the content type is from
[required]
  --contenttype, -c      The content type  [required]
  --field, -f            The field the field editor is for  [required]
  --contenttemplate, -n  Flag to indicate the template is a content
template
  --help, -h             Show help  [boolean]

Examples:
  cec add-field-editor editor1 -t BlogTemplate -c BlogPost -f
summary            Use editor1 as the appearance for field summary in
content type BlogPost from local template at src/templates/BlogTemplate
  cec add-field-editor editor1 -t BlogTemplateContent -n -c BlogPost -
f summary  Use editor1 as the appearance for field summary in content
type BlogPost from local template at src/content/BlogTemplateContent
------------------------
cec remove-field-editor
------------------------
Usage: cec remove-field-editor <name>

Removes a field editor from a field in a content type.

Options:
  --template, -t         The template the content type is from
[required]
  --contenttype, -c      The content type  [required]
  --field, -f            The field the field editor is for  [required]
  --contenttemplate, -n  Flag to indicate the template is a content
template
  --help, -h             Show help  [boolean]

Examples:
  cec remove-field-editor editor1 -t BlogTemplate -c BlogPost -f
summary            Remove editor1 as the appearance for field summary
in content type BlogPost from local template at src/templates/
BlogTemplate
  cec remove-field-editor editor1 -t BlogTemplateContent -n -c
BlogPost -f summary  Remove editor1 as the appearance for field
```

```
summary in content type BlogPost from local template at src/content/
BlogTemplateContent
```

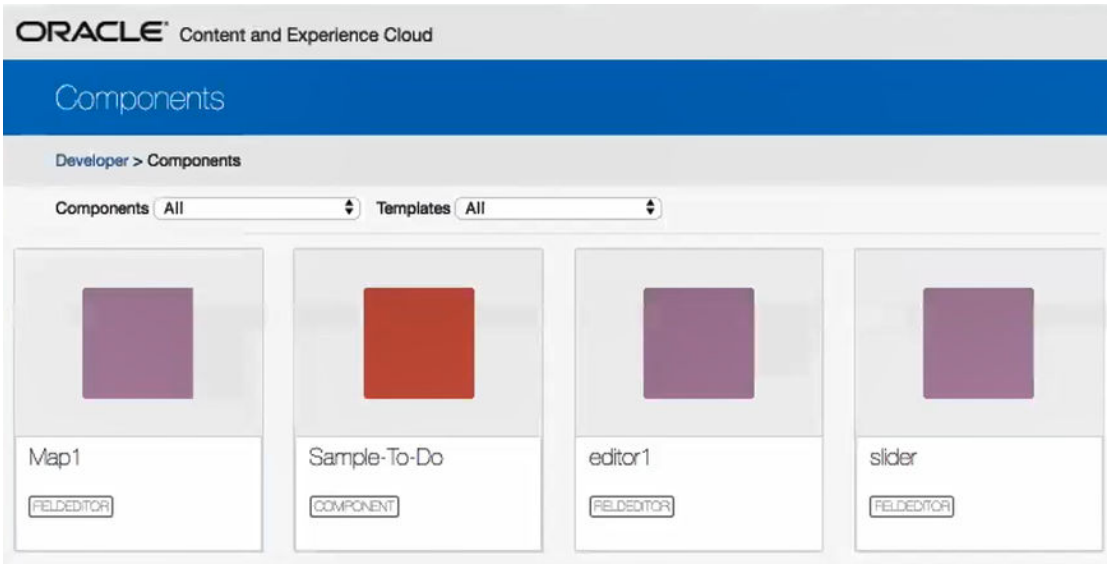
These samples of Field Appearance components are included with Content Toolkit:

- `TextFieldEditor`
- `SliderFieldEditor`
- `MapFieldEditor`

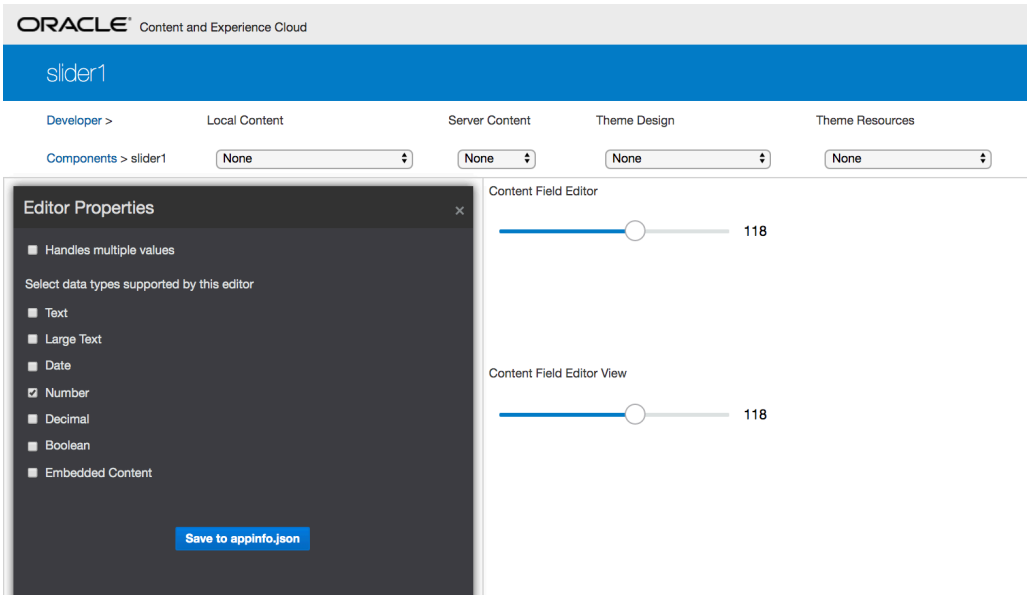
The following image shows Content Toolkit commands you can use to develop the sample Field Appearance components.

```
2
3   # create
4   cec cc editor1 -f TextFieldEditor
5
6   cec cc slider -f SliderFieldEditor
7
8   cec cc Map1 -f MapFieldEditor
9
10  #local testing
11
12  # add to content type field
13  cec add-field-editor editor1 -t SimpleContent -n -c SimpleType -f title
14
15  cec add-field-editor slider -t SimpleContent -n  -c SimpleType -f value
16
17
18  # upload editors
19  cec ulcp editor1,slider -p -s
20
21  # upload content
22
23  cec cr Repo5 -s
24
25  cec upload-content SimpleContent -r Repo5 -s
26
```

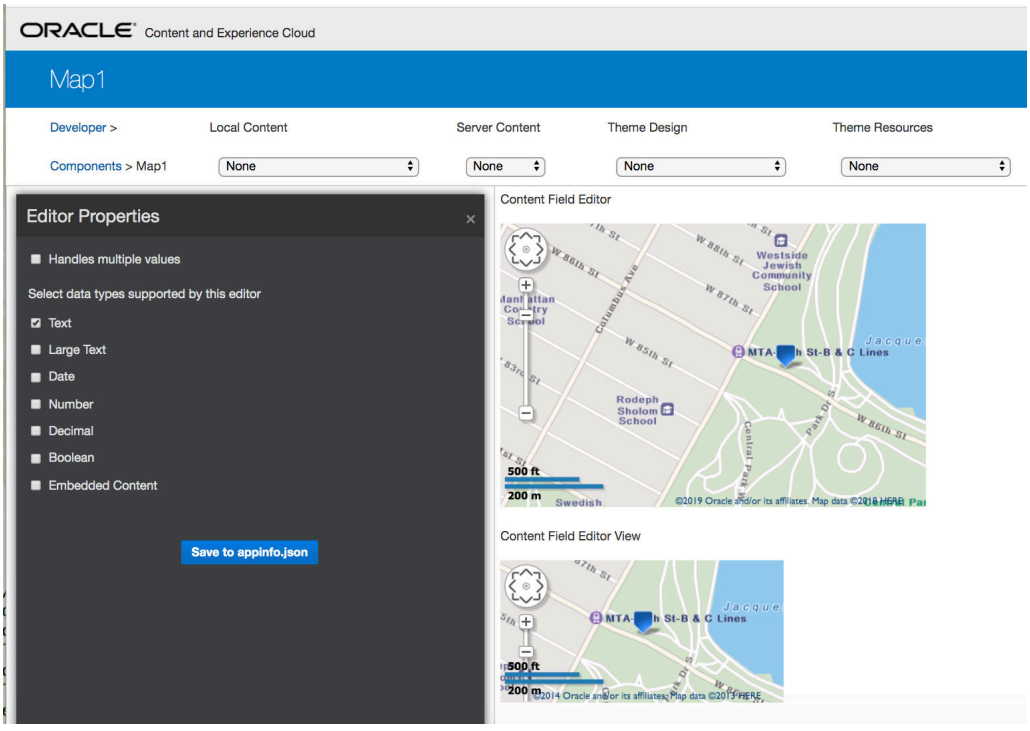
You can create the out-of-the-box Field Appearance components on your local server, test them, and then upload them to your Oracle Content Management instance. The following image shows these components on `localhost:8085`.



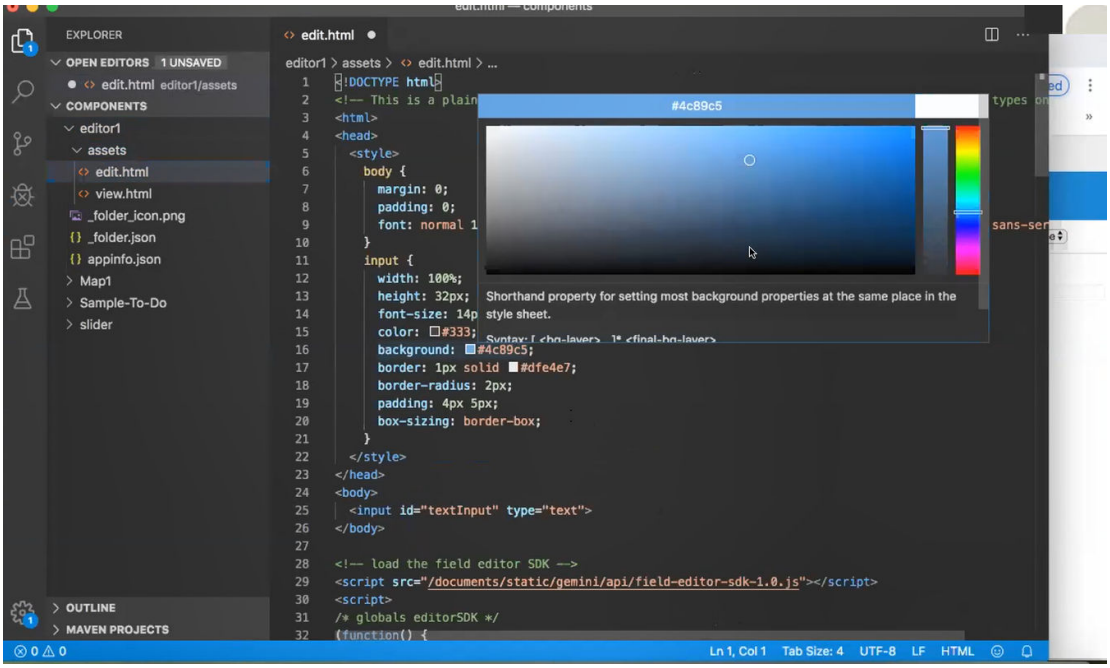
You can test each component, such as slider, on the local server. There you can select properties for the component and then save it.



For the map component, you can click around on the map to provide a location as the editor value.



You can edit the HTML file for a component to change its settings, such as background color.



You can use a Content Toolkit command to associate a field editor with a field of a content type locally:

```
cec add-field-editor editor1 -t SimpleContent -n -c SimpleType -f title

cec add-field-editor slider -t SimpleContent -n -c SimpleType -f value
```

After you finish configuring and testing the custom field editors, you can upload them to your Oracle Content Management instance, using Content Toolkit commands:

```
# upload editors
cec ulcp editor1,slider -p -s Latest

# upload content

cec cr Repo5 -s

cec upload-content SimpleContent -r Rpo5 -s
```

When you upload the custom field editors, your components are imported into your Oracle Content Management instance and can be found on the **Components** page of the **Developer** section.

Transfer or Update a Site from One Server to Another

As a developer, you can use an Oracle Content Management Toolkit command to create or update a site and its content from server A to server B.

By default all assets are transferred, optionally specify -p to transfer only published assets. Specify the source server with `-s <server>` and the destination server with `-d`

<destination>. If the site contains assets from other repositories, optionally provide the repository mapping otherwise those assets will not be transferred.

You can use the following command to update or transfer a site from test to production:

```
cec transfer-site site --from server --to server --repository r
      --localization-policy l
```

For additional options, see Use the cec Command-Line Utility.

Transfer a Site Without Content Items

As a Developer, you can use Oracle Content Management Toolkit to transfer a site without content items from one Oracle Content Management server to another.

To transfer a site with a large number of content assets, you need to separate the site and its content. You can add the `--excludecontent` option (shortcut `-x`) to the `transfer-site` command.

When this option is set, only the site will be transferred.

For example:

```
cec transfer-site Site1 -s DEV -d UAT -r Repository1 -l
LocalizationPolicy1 -x
```

Download or Upload Content Items for a Site in Groups

As a Developer, when you transfer a site from one Oracle Content Management server to another, you can download or upload the site's content items in groups.

To transfer a site with a large number of content assets, you need to separate the site and its content. After you transfer a site without content (`cec transfer-site --excludecontent`), use `cec transfer-site-content` to transfer the content of the site.

For example:

```
cec transfer-site-content <name>
```

This command creates scripts to transfer Enterprise Site content from one Oracle Content Management server to another. The command is used to transfer a large number of content items, and the items are transferred in batches. By default, this command will not execute the scripts, and all assets are transferred. You can specify `-p` to transfer only published assets.

Specify the source server with `-s <server>` and the destination server with `-d <destination>`.

Options

```
--destination, -d The registered CEC server to transfer the content
```

```
[required]
--repository, -r Repository [required]
--publishedassets, -p The flag to indicate published assets only
--number, -n The number of items in each batch, defaults to 500
--execute, -e Execute the scripts
--help, -h Show help [boolean]
```

If the option `--execute` is not set, after `cec transfer-site-content` finishes, execute the generated script `<site name>_downloadcontent` to download content from the source server, and execute `<site name>_uploadcontent` to upload the downloaded content to the destination server.

Examples

```
cec transfer-site-content Site1 -s DEV -d UAT -r Repository1 Generate script
Site1_downloadcontent and Site1_uploadcontent
cec transfer-site-content Site1 -s DEV -d UAT -r Repository1 -e Generate
script Site1_downloadcontent and Site1_uploadcontent and execute them
cec transfer-site-content Site1 -s DEV -d UAT -r Repository1 -n 200
cec transfer-site-content Site1 -s DEV -d UAT -r Repository1 -p
```

Add Content Items to a Channel

You can use the Content Toolkit `control-content` command to add content items to a channel in an Oracle Content Management server.

The `control-content <action>` command has an action, `add`, for adding content items to an Oracle Content Management channel:

```
cec control-content add -c Channel1 -r Repo1 -s UAT
```

This command add all items in the repository `Repo1` to the channel `Channel1` on the registered server UAT.

You can specify the server with `-s <server>`or use the server specified in the `cec.properties` file.

The valid actions for the `content-usage` command follow:

- `publish`
- `unpublish`
- `add`
- `remove`

Options for the `content-usage` command follow:

- `--channel, -c Channel` [required]
- `--repository, -r Repository` [required when `<action>` is `add`]
- `--server, -s` The registered Oracle Content Management server
- `--help, -h` Show help [boolean]

Examples of the `control-content` command follow:

```
cec control-content publish -c Channel1

Publish all items in channel Channel1 on the server specified in the
cec.properties file

cec control-content publish -c Channel1 -s UAT

Publish all items in channel Channel1 on the registered server UAT

cec control-content unpublish -c Channel1 -s UAT

Unpublish all items in channel Channel1 on the registered server UAT

cec control-content add -c Channel1 -r Repo1 -s UAT

Add all items in repository Repo1 to channel Channel1 on the
registered server UAT.

cec control-content remove -c Channel1 -s UAT

 Remove all items in channel Channel1 on the registered server UAT
```

Index Site Pages with Oracle Content Management Toolkit

You can use Oracle Content Management Toolkit to create content items for text on site pages and to enable page search for a site.

The following sections describe how to index site pages with Content Toolkit:

1. Create the Content Type for Site Page Text
2. Create Page Index Content Items with Content Toolkit
3. Add Content Search to a Site in Oracle Content Management

Create the Content Type for Site Page Text

For a content type, you specify a name, required field values, a default content layout for the type.

- Type name

  Specify any valid content type name.

- Fields

  The following fields are required.

Field Name	Field Type	Number of data field values	Description
site	Text	Single	Site name
pageid	Text	Single	Page ID
pagename	Text	Single	Page name

Field Name	Field Type	Number of data field values	Description
pageurl	Text	Single	Page URL
pagedescription	Text	Single	Page description
keywords	Text	Multiple (no max)	All text on the page and the values from all text fields of content items on the page, obtained by the Content Toolkit `index-site` command

```
{{#fields}}
<div class="indextype"></div>
<div>
    <a href="{{pageFullURL}}" title="{{pagename}}">{{pagename}}</a>
</div>
{{/fields}}


content.fields.pageFullURL = SCSRenderAPI.getSitePrefix() +
content.fields.pageurl;
```

- Create a content layout for the type.

  The content layout should display the site name and the URL to navigate to the page. For example, in `layout.html`:

  ```
  {{#fields}}
  <div class="indextype"></div>
  <div>
     <a href="{{pageFullURL}}"title="
  {{pagename}}">{{pagename}}</a>
  </div>
  {{/fields}}
  ```

- In `render.js`, generate the page full URL:

  ```
  content.fields.pageFullURL =
  SCSRenderAPI.getSitePrefix() +
  content.fields.pageurl;
  ```

- Set the content layout as the default content layout for the type.

  ```
  content.fields.pageFullURL =
  SCSRenderAPI.getSitePrefix() + content.fields.pageurl;
  ```

Create Page Index Content Items with Content Toolkit

You can use an Content Toolkit command to create page index content items.

Prerequisites:

- Content Toolkit has been installed and set up on your local machine.

- The site on Oracle Content Management has been published.
- The content items on the site page have been published to the site channel.

In a command-line interface, type the following Content Toolkit command:

```
cec index-site site name -c content type name -p
```

In the command, *site name* is the name of the site, *content type name* is the content type created for the page text; and the option `-p` indicates to publish the page index content items after creation.

```
Usage: cec index-site <site>

Create content item for each page with all text on the page. If the
page index content item already exists for a page, updated it with
latest text on the page. Specify -c
<contenttype> to set the page index content type. Optionally specify -
p to publish the page index items after creation or update.

Options:
  --contenttype, -c  <contenttype> page index content type
  --publish, -p      publish page index items
  --help, -h         Show
help

  [boolean]

Examples:
  cec index-site Site1 -c PageIndex
  cec index-site Site1 -c PageIndex -p
```

To see the usage, you can type `cec index-site -h`

Add Content Search to a Site in Oracle Content Management

You can add content search to an Oracle Content Management site with a search page and search field.

To add content search to a site:

1. Add a Search Page to the Site
2. Add a Search Field to the Theme

Add a Search Page to the Site

You can add a search page to a site and a Content List component to the search page.

Add the search page:

1. Add a page to the site and set it as a search page.
2. Add a Content List component to the search page.
3. Set **Content Type** to the page index content type created previously.

Add a Search Field to the Theme

To make a search field show on every page of a site, you can add the search field to the theme's layout HTML page.

For example:

```
<div align="center">
<input  id="searchonpage" type="text" size="30" placeholder="Search on
page. . ."/>
</div>
```

1. Add the input field :

```
<script>
    // Get the search field element
    const node = document.getElementById('searchonpage');
    // Get the search string from the url if it exists
    var params = (new URL(document.location)).searchParams;
    var defaultStr = params && params.get('default');
    if (defaultStr) {
        if (defaultStr.lastIndexOf('*') === defaultStr.length - 1) {
            defaultStr = defaultStr.substring(0, defaultStr.length - 1);
        }
        // Display the search string in the search field
        node.value = defaultStr;
    }
    // When enter from the search field, go to the site search page with
the search string
    node.addEventListener('keydown', function onEvent(event) {
        if (event.key === "Enter") {
            var inputElem = event.srcElement || event.target;
            var siteSearchPageUrl = 'search.html';
            var searchUrl = SCSRenderAPI.getSitePrefix() +
                siteSearchPageUrl +
                '?contentType=indextype&default=' + inputElem.value + '*';
            window.location = searchUrl;
        }
    });
</script>
```

2. Add the JavaScript at the end of the HTML body.

Index a Multilingual Site with Oracle Content Management Toolkit

You can use Oracle Content Management Toolkit to index multilingual (MLS) sites for translations and for searching pages and content items.

You can build a multilingual site index and test it before publishing the site. Use the Content Toolkit `cec index-site` command to index a multilingual site. Go to the `cec-components`

directory and issue this command without any options to view the help information for the command:

```
cec-components> cec index-site
Usage: cec index-site <site>

Create content item for each page with all text on the page. If the
page index content item already exists for a pate, updated it with
latest
text on the page. Specify -c <contenttype> to set the page index
content type. Optionally specify -p to publish the page index items
after
creation or update.

Options:
  --contenttype, -c <contenttype> page index content type
  --publish, -p     publish page index items
  --help, -h        Show
help
                                        [boolean]

Examples:
  cec index-site Site1 -c PageIndex
  cec index-site Site1 -c PageIndex -p
```

Page index items exist per page and per language. The page index content items created for each language are created as translations of the default language page index items. When you do a query in the running site, the search and the content list pick up the language from the site URL. This filters the search automatically.

Before you can publish a multilingual site, you need to index and translate it, for which you will need a translation job. See Create a New Site or Asset Translation Job in the Oracle Content Management Server.

The default language, English, is required. For each language supported (required and optional), execute the index creation and create translations of index items. If you run the index twice, it just does an update.

To index, translate, and publish a multilingual site using Content Toolkit commands:

1. Create a content type for the site and make it available in the repository. See Create the Content Type for Site Page Text.
2. Select a validation policy.
    a. Click **Assets** in the left navigation menu.
    b. Choose **Localization Policies** in the **Assets** menu.
    c. Select a localization policy.
    d. Modify the localization policy, if necessary, to include the languages you want to use for indexing and translating the site. For example, if the policy has only English, you can add French and Spanish.

       All translations are done from English.
3. Download a translation job. You can translate only the assets that are used in the site.

4. Translate the site. Site translations can be done manually or through an integrated connector.
5. Upload the translation job once translation is finished.
6. Use the `cec index-site` command to index the site. Specify -c <contenttype> to set the page index content type.

   You can also specify the `-p` option to publish the site. Then you can validate the indexing and translation before publishing the changes to the live site.

   For example, the following `cec index-site` command builds a site index for a site that uses English, French, and Spanish. The languages supported by the site are from the assigned L10n policy, including the default language.

```
cec index-site Demo2 -c search_content_type -p
 - Logged in to remote server: server-URL
 - establish user session
 - get CSRF token
 - site: Demo2, default language: en-US, channel token: channel-token
 - site localization policy: search_localization_policy
 - query site repository
 - query content type search_content_type
 - query site structure
 - content types used in the site: search_blog
 - query page data
 - query content on the pages
 - will create 11 page index items
 - will update 0 page index items
 - will remove 1 page index items
 - create page index item for Blog
 - create page index item for Privacy Policy
 - create page index item for Search
 - create page index item for Components
 - create page index item for Navigtion
 - create page index item for Detail Page
 - create page index item for Pages
 - create page index item for Page Content
 - create page index item for Developing Templates
 - create page index item for Themes
 - add page index items to site channel
 - remove page index items for page Search from site channel
 - will create/update translate for fx-FR,es-ES
 - query site stucture with locale fr-FR
 - query page data (fr-FR)
 - query content on the pages (fr-FR)
 - will create 11 page index items (fr-FR)
 - will update 0 page index items (fr-FR)
 - will remove 1 page index items (fr-FR)
 - create page index item for Themes (fr-FR)
 - create page index item for Navigation (fr-FR)
 - create page index item for Pages (fr-FR)
 - create page index item for Detail Page (fr-FR)
 - create page index item for Search (fr-FR)
 - create page index item for Page Content (fr-FR)
 - create page index item for Components (fr-FR)
 - create page index item for Developing Templates (fr-FR)
```

```
- create page index item for Blog (fr-FR)
- create page index item for Home (fr-FR)
- create page index item for Privacy Policy (fr-FR)
- add page index items to site channel
- set page index items in fr-FR as translated
- remove page index items for page Search from site channel
- query site stucture with locale es-ES
- query page data (es-ES)
- query content on the pages (es-ES)
- will create 11 page index items (es-ES)
- will create 0 page index items (es-ES)
- create page index item for Pages (en-ES)
- create page index item for Home (en-ES)
- create page index item for Themes (en-ES)
- create page index item for Components (en-ES)
- create page index item for Privacy Policy (en-ES)
- create page index item for Detail Page (en-ES)
- create page index item for Page Content (en-ES)
- create page index item for Navigation (en-ES)
- create page index item for Developing Templates (en-ES)
- create page index item for Search (en-ES)
- create page index item for Blog (en-ES)
- add page index items to site channel
- set page index items in es-ED as translated
- publish job submitted
- publish in proogress
- publish in progress
- publish page index items finished
```

7. Publish the site to include translations.

Create a Simplified Component for Easy Component Development

Use Oracle Content Management Toolkit to create a simplified component for easier development.

`SimpleHTML`, a simplified component, is available in Content Toolkit to give you an easier starting point for custom component development:

```
cec create-component -f SimpleHTML
```

A sample for JET component is also available for you to start with:

```
cec create-component MyComp -f JET-CCA-Demo-Card
```

Set Up a Site Compilation Service

You can create a Docker image for the compilation server to set up a Site Compilation Service in Oracle Content Management. This service gives you the option to define a compilation server to use before you publish.

The compilation server has an endpoint in the form of a URL on the **Administration>System>Sites and Assets** page, in the **Compilation Endpoint URL** field. There you can enter the fully-qualified URL you want to be registered with the server, and then click **Test** to validate the endpoint. See Set a Compilation Endpoint URL.

The Site Compilation Service compiles pages of a site so that HTML pages are returned when the published site is accessed. When the Oracle Content Management server publishes a site, it will call the Site Compilation Service to compile the pages, if the service is configured.

Once you specify the endpoint for a site, you can enable automatic compilation on the **Static Site Delivery** tab of the **Site Properties** panel. When you publish the site, it gets compiled through the compilation server.

> **Note:**
>
> Oracle Content Management has a compilation service built-in to easily allow you to compile a site without any additional configuration, or you may manually set up a compilation service for testing purposes to validate site compilation, or to use custom libraries.
>
> If a compilation server endpoint field is not defined, the site will automatically use the built-in site compilation service provided that you've enabled automatic compilation on the **Static Site Delivery** tab of the **Site Properties** panel.

The Site Compilation Service is an extension to the Toolkit commands. You can run `cec compilation-server` yourself, but the Docker image lets you create a compilation server using the standard mode and then adjust the configuration of the service.

To create the Docker image and publish the Docker file:

1. Download the Docker image information from GitHub.
   When you download the information from GitHub, you have an additional three files, which are underneath the compilation server. One of the Docker images is the compilation server, which includes a Docker file and readme files.
2. The `run.sh` command is executed after the Docker image is created.
   This command downloads the Oracle Content Management Toolkit, installs it, and creates your source directory.
3. Then you can customize your environment and run the `run.sh` command. You can specify which port to use, timeouts, and anything else you want to change and then update and use the .sh command.
4. After you download the Docker image information and change the compilation server directory, you can build the Docker file.
   You can remove the no-cache option if you have already downloaded the Content Toolkit so that you don't re-download everything.
5. List your Docker image or images.
6. Once the Docker image is available to you, register it.
7. Verify that it works.
   Go to the exception file REST API, which lets you see which versions are supported. This test lets you validate that it runs.

8. Now you need to register the server. It requires the first part of the server name and handles everything else inside of there.
9. Go back to your system directory and your sites assets, where you can register your compilation endpoint and click **Save**.
10. Go to your site's properties and specify what to publish and when to publish it.

Compile a Site to Improve Runtime Performance for Site Pages

Compilation of a site in Oracle Content Management can improve the runtime performance and behavior of site pages. Compilation achieves this by creating a static HTML file for each page in the site that will behave exactly as the original page.

> **Note:**
>
> You can compile content layouts only if your Oracle Content Management instance is running natively on Gen 2 Oracle Cloud Infrastructure (OCI). See Compile and Publish Sites with Oracle Content Management.

Compile a Site for Mobile Devices

You can use the Content Toolkit to compile a mobile layout for a site web page. The mobile layout can be different from the desktop page layout for the same content. Or the mobile and desktop layouts can be the same.

In the site editor, you can choose the same page layout for mobile devices as the desktop layout, or you can specify a different page layout. With Content Toolkit, you can compile the static layout for mobile devices separately.

You can view the site page differently on a mobile device. A page rendered on a mobile device might not have a banner like the page does in a desktop layout.

In Content Toolkit, the help page for `cec compile-template` shows the `targetDevice` option for targeting a particular device when you compile a site template:

```
C:\git\webclient\developer\test\sites-compiler\cec-install>cec compile-template --help
Usage: cec compile-template <source>

Compiles all the pages within the site of the template and places the compiled pages under the sites assets folder.
Optionally specify -s <server> to make content queries against this server (requires channelToken).
Optionally specify -c <channelToken> to use this channelToken when generating any content URLs.
Optionally specify -t <contentType> [draft | published] content to retrieve from the server type, defaults to published.
Optionally specify -p <pages> the set of pages to compile.
Optionally specify -d <debug> to start the compilation with --inspect-brk flag.
Optionally specify -r <recurse> recurse through all child pages of specified pages.
Optionally specify -l <includeLocale> include default locale when creating pages.
Optionally specify -a <targetDevice> [desktop | mobile] target device type when using adaptive layouts.
Optionally specify -v <verbose> to display all warning messages during compilation.


Options:
  --server, -s                  The registered CEC server
  --channelToken, -c            The channel access token to use for content URLs
  --type, -t                    The type of content to retrieve from the serve [published | draft]
  --pages, -p                   The list of pages to compile
  --recurse, -r                 Compile all child pages of those specifed in the page list
  --debug, -d                   Start the compiler with "--inspect-brk" option to debug compilation
  --noDetailPages, -e           Do not generate compiled detail pages
  --noDefaultDetailPageLink, -o Do not generate compiled detail page for items/content lists that use the default detail page
  --targetDevice, -a            The target device type when using adaptive layouts [desktop | mobile]
  --includeLocale, -l           Include default locale when creating pages
  --verbose, -v                 Run in verbose mode to display all warning messages during compilation.
  --help, -h                    Show help

Examples:
  cec compile-template Temp1                                Compiles the site in template Temp1 using content stored in the template.
  cec compile-template Temp1 -c channelToken                Compiles the site in template Temp1 using the given channelToken for any content URLs.
  cec compile-template Temp1 -c channelToken -s UAT -t draft  Compiles the site in template Temp1 retrieving draft content from the specified server.
  cec compile-template Temp1 -p 104,112,183 -r              Compiles the specified pages in the site in template Temp1 including all child pages.
  cec compile-template Temp1 -d                             Waits for the debugger to be attached.  Once attached, compiles the site in template Temp1.


C:\git\webclient\developer\test\sites-compiler\cec-install>
```

When you compile your site, you can specify whether you want to compile for `desktop` or for `mobile`. Desktop files are placed `under static/_files`. Mobile files are placed under `static/_mobilefiles`.

```
C:\git\webclient\developer\test\sites-compiler\cec-install>cec compile-template Corporate-Site-Template --targetDevice desktop
Compile Template: compiling template Corporate-Site-Template
Oracle Content and Experience Site Compiler

Compiling: desktop pages
------------------------
createPage: Processing pageId 10. Preview URL: http://localhost:8085/templates/Corporate-Site-Template/index.html
createPage: Processing pageId 100. Preview URL: http://localhost:8085/templates/Corporate-Site-Template/developing-templates.html
createPage: Processing pageId 110. Preview URL: http://localhost:8085/templates/Corporate-Site-Template/developing-templates/themes.html
createPage: Processing pageId 120. Preview URL: http://localhost:8085/templates/Corporate-Site-Template/developing-templates/pages.html
createPage: Processing pageId 130. Preview URL: http://localhost:8085/templates/Corporate-Site-Template/developing-templates/navigation.html
createPage: Processing pageId 140. Preview URL: http://localhost:8085/templates/Corporate-Site-Template/developing-templates/page-content.html
createPage: Processing pageId 150. Preview URL: http://localhost:8085/templates/Corporate-Site-Template/developing-templates/components.html
createPage: Processing pageId 160. Preview URL: http://localhost:8085/templates/Corporate-Site-Template/developing-templates/sign-in.html
createPage: Processing pageId 200. Preview URL: http://localhost:8085/templates/Corporate-Site-Template/privacy-policy.html
All page creation calls complete.


Compilation completed with 0 errors and 5 warnings.
 to display warnings, run with --verbose (-v) option.

 *** compiled template is ready to test
 *** to render non-compiled pages, remove compiled files from under: C:\git\webclient\developer\test\sites-compiler\cec-install\src\templates\Corporate-Site-Template\static

C:\git\webclient\developer\test\sites-compiler\cec-install>cec compile-template Corporate-Site-Template --targetDevice mobile
Compile Template: compiling template Corporate-Site-Template
Oracle Content and Experience Site Compiler


Compiling: mobile pages
-----------------------
createPage: Processing pageId 10. Preview URL: http://localhost:8085/templates/Corporate-Site-Template/index.html
createPage: Processing pageId 100. Preview URL: http://localhost:8085/templates/Corporate-Site-Template/developing-templates.html
createPage: Processing pageId 110. Preview URL: http://localhost:8085/templates/Corporate-Site-Template/developing-templates/themes.html
createPage: Processing pageId 120. Preview URL: http://localhost:8085/templates/Corporate-Site-Template/developing-templates/pages.html
createPage: Processing pageId 130. Preview URL: http://localhost:8085/templates/Corporate-Site-Template/developing-templates/navigation.html
createPage: Processing pageId 140. Preview URL: http://localhost:8085/templates/Corporate-Site-Template/developing-templates/page-content.html
createPage: Processing pageId 150. Preview URL: http://localhost:8085/templates/Corporate-Site-Template/developing-templates/components.html
createPage: Processing pageId 160. Preview URL: http://localhost:8085/templates/Corporate-Site-Template/developing-templates/sign-in.html
createPage: Processing pageId 200. Preview URL: http://localhost:8085/templates/Corporate-Site-Template/privacy-policy.html
All page creation calls complete.

Compilation completed with 0 errors and 6 warnings.
 to display warnings, run with --verbose (-v) option.

 *** compiled template is ready to test
 *** to render non-compiled pages, remove compiled files from under: C:\git\webclient\developer\test\sites-compiler\cec-install\src\templates\Corporate-Site-Template\static
```

After you compile a template for mobile devices, the Content Toolkit command `upload-static-site-files` will support the mobile files.

Create a New Site or Asset Translation Job in the Oracle Content Management Server

Use Oracle Content Management Toolkit to create a translation job for a site or an asset in Oracle Content Management.

Before you can index a multilingual site, you need a translation job. To create a translation job:

1. Click **Translate** on the top menu of the Sites page.
2. Enter a name for the job in the **Create Translation Job** dialog, and choose the default source language, the target languages, and the translation job contents.

   You can choose to have your translation package include all site content and targeted assets, only site content, or only assets targeted to the site's publishing channel.

   Exclude from the translation any content items that are configured with the **Do not translate** text setting. For example, product names typically are not translated.
3. Click **Create** to create the translation job.
4. Use a Content Toolkit command to list the available jobs:

```
cec components> cec list-translation-jobs
Asset translation jobs:
Name                                                Status
Source Language Target Languages                        Pending
Languages

Site translation jbs:
Name                                                Status
Source Language Target Languages                        Pending
Languages

demo1                                               READY        en-
US           fr-FR,es-ES                             fr-FR,es-ES

searchdemo1                                         TRANSLATED   en-
US           fr-FR,es-ES
```

5. Download your translation job:

```
cec components> cec download-translation-job demo1
 - translation job downloaded to /Users/<user-name>/Dev/webclient/
developers/sites-toolkit/cec-components/demo.zip
 - update the translation job status to INPROGRESS.
cec components> cec translate dmo1.zip -l all -t demo1-xlate.zip
 - target languages: fr-FR,ex-ES
 -  translation finished: /Users/<user-name>/Dev/webclient/
developers/sites-toolkit/cec-components/demo1-xlate.zip
```

6. Open the translation bundle and build the folders of resources for languages that you're translating to:

```
Unzip demo1-xlate.zip
ARchive: emo1-xlate.zip
replace assets/job.json? [n]o, [A]ll, [N]one, [r]ename: A
  inflating assets/job.json
  inflating site/job.json
  inflating assets/es-ES/CORE47653001483240C1AAF180C435F189AB-
search_siteSearch202.json
  inflating assets/es-ES/COREA570227E12194356BAA16A80A78A2670-entry1.json
  inflating assets/es-ES/CORED977BC199A3B494596F0D467CAADF7FA-entry2-json
  inflating assets/fr-FR/CORE47653001483240C1AAF18DC435F1B9A8-
search_siteSearch202.json
  inflating assets/fr-FR/COREA570227E12194356BAA16A80A78A2670-entry1.json
  inflating assets/fr-FR/CORED977BC199A3B494596F0D467CA4DF7FA-entry2.json
  inflating assets/root/CORE476530014B3240C1AAF18DC435F1B948-
search_siteSearch202.json
  inflating assets/root/COREA570227E12194356BAA16A80A7842870-entry1.json
  inflating assets/root/CORED977BC199A38494596F0D467CA4DF7FA-entry2.json
  inflating site/es-ES/10.json
  inflating site/es-ES/100.json
  inflating site/es-ES/110.json
  inflating site/es-ES/120.json
  inflating site/es-ES/130.json
  inflating site/es-ES/140.json
  inflating site/es-ES/150.json
  inflating site/es-ES/200.json
  inflating site/es-ES/201.json
  inflating site/es-ES/202.json
  inflating site/es-ES/203.json
  inflating site/es-ES/siteinfo.json
  inflating site/es-ES/structure.json
  inflating site/fr-FR/10.json
  inflating site/fr-FR/100.json
  inflating site/fr-FR/110.json
  inflating site/fr-FR/120.json
  inflating site/fr-FR/130.json
  inflating site/fr-FR/140.json
  inflating site/fr-FR/150.json
  inflating site/fr-FR/200.json
  inflating site/fr-FR/201.json
  inflating site/fr-FR/202.json
  inflating site/fr-FR/203.json
  inflating site/fr-FR/siteinfo.json
  inflating site/fr-FR/structure.json
  inflating site/root/10.json
  inflating site/root/100.json
  inflating site/root/110.json
  inflating site/root/120.json
  inflating site/root/130.json
  inflating site/root/140.json
  inflating site/root/150.json
  inflating site/root/200.json
  inflating site/root/201.json
```

```
  inflating site/root/202.json
  inflating site/root/203.json
  inflating site/root/siteinfo.json
  inflating site/root/structure.json
  inflating
  inflating
  inflating
  inflating
  inflating
  inflating
  inflating
  inflating
```

7. Import the translation job:

```
cec-components> cec import-translation-job demo1-xlate.zip
 - Logged in to remote server: <server url>
 - file demo1-xlate.zip uploaded to home folder, version 1
 - importing: percentage 5
 - importing: percentage 60
 - import demo1 finished
```

Translate a Site with a Language Service Provider

You can manage translations of a site for multiple languages with the Oracle Content Management Toolkit command-line interface and a Language Service Provider (LSP).

The localization policy for a site specifies a default language, such as English (United States) (en-US), and one or more alternate languages for the site, such as German and French. The text strings for a site can be translated into the specified alternate languages. If you change the language for the site before translation, the text strings will still appear in the default language.

Content Toolkit provides the following translation options in the command-line interface:

```
Translation
  cec list-translation-jobs                          Lists translation
jobs.                                                        [alias:
ltj]
  cec create-translation-job <name>                  Creates a
translation job <name> for a site on CEC server.
[alias:  ctj]
  cec download-translation-job <name>                Downloads
translation job <name> from CEC server.
[alias:  dtj]
  cec submit-translation-job <name>                  Submits
translation job <name> to translation connection <connection>.
[alias:  stj]
  cec ingest-translation-job <name>                  Gets translated
job <name> from translation connection and ingest.           [alias:
itj]
  cec upload-translation-job <name>                  Uploads
translation job <name> to CEC server.
```

```
[alias:  utj]
  cec create-translation-connector <name>          Creates translation
connector <name>.                                        [alias:  ctc]
  cec start-translation-connector <name>           Starts translation
connector <name>.                                         [alias:  stc]
  cec register-translation-connector <name>        Registers a translation
connector.                                          [alias:  rtc]
```

You can use the `cec list-translation-jobs` command to list the translation jobs that are already on the server. For example:

```
cec ltj -s
Server: <server-name>
Asset translation jobs:
Name                                    Status        Source Language Target
Languages                          Pending Languages
testHash                                INPROGRESS    en-US           fr-
FR,de-DE                              fr-FR,de-DE
Site translation jobs:
Name                                    Status        Source Language Target
Languages                          Pending Languages
demoTest                                TRANSLATED    en-US           de-
DE,fr-FR
```

Typing any `cec` command without parameters or with `-h` provides help for the command. See Use the cec Command-Line Utility.

The following sections provide information about translating a site with an LSP:

1. Create a Translation Job with Oracle Content Management Toolkit
2. List Translation Jobs
3. Create a Translation Connector
4. Generate a Site Map for a Multilingual Site
5. Submit a Translation Job to a Language Service Provider
6. Upload a Translation Job to the Server

Create a Translation Job with Oracle Content Management Toolkit

You can use an Oracle Content Management Toolkit command to create a site translation job on your local system.

To create a new translation job for a site, use the `cec create-translation-job` command. This command finds all the assets for the site and creates a zip file for everything that needs to be translated from that site.

```
cec create-translation-job FridayDemo -s Take2 -l all
 - Logged in to remote server: <server-name>
 - establish user session
 - site: Take2, default language: en-US
 - query channel
 - site localization policy: MyLP
 - target languages: de-DE, fr-FR
```

```
 - create translation job submitted
 - creating: percentage 50
 - translation job FridyDemo created
```

For translation options, see Create a New Site or Asset Translation Job in the Oracle Content Management Server.

List Translation Jobs

You can list translation jobs on the server to verify that your job was created and is ready to work with.

```
cec list-translation-jobs -s
Server: <server-name>
Asset translation jobs:
Name                                    Status        Source Language
Target Languages                        Pending Languages
testHash                                INPROGRESS    en-US
fr-FR,de-DE                             fr-FR,de-DE
Site translation jobs:
Name                                    Status        Source Language
Target Languages                        Pending Languages
demoTest                                TRANSLATED    en-US
de-DE,fr-FR
FridayDemo                              READY         en-US
de-DE,fr-FR                             de-DE,fr-FR
```

Notice that the `FridayDemo` job is in a `READY` state.

Create a Translation Connector

A Language Service Provider (LSP) can help you translate a site. With a translation connector to the LSP, you can submit and ingest translation jobs.

Before you submit a translation job, you need to create a translation connector. To translate a site without an LSP, you can create a mock translation connector to run against. Use the `cec create-translation-connector` command to create a translation connector and the `cec start-translation-connector` command to start it:

```
cec create-translation-connector connector1
 - translation connector connector1 created at <sites-toolkit folder>/
cec-components/src/main/connectors/connector1
 - install connector
. . .
Start the connector: cec start-translation-connector connector1 [-p
<port>]
cec start-translation-connector connector1 -p 7777
NodeJS running. . .:
Site page: http://localhost:7777
```

Use the Content Toolkit to test the translation connector by running it through the expected APIs:

1. Register the connector with Content Toolkit.

   ```
   >cec register-translation-connector
   ```

2. Open up the toolkit and go to the "Translation Connections" page.

   ```
   >http://localhost:8085/public/translationconnections.html
   ```

3. Run through the steps on the translation connector validation page. These steps use the `translationBundle.zip` file in the `/data` folder in your connector environment to validate your connector.

You can use the Translation Connector SDK to develop a translation connector for Oracle Content Management. This SDK is a sample NodeJS implementation of the Translation connector API. The sample accepts an Oracle Content Management translation job zip file, translates all the resource in the file, and returns a new zip file containing all the translations.

The SDK requires the user to have access to an LSP to do the actual string translations. A mock LSP server is included in the SDK to mimic the responses from an LSP by simply prepending the targeted locales to the strings.

The Translation Connector SDK consist of three main modules.

- **Connector:** The translation connector that implements the required Oracle Content Management Translation Connector API.
- **Job Manager:** A file system-based sample job manager that maintains the state of the connector jobs while they are translated by the Language Service Provider.
- **Provider:** The implementation of the specific set of APIs required by your LSP to submit documents for translation and retrieve the translated documents.

You can copy the mock translation Provider JS and implement all the methods inside it.

Generate a Site Map for a Multilingual Site

Use Content Toolkit to generate a site map for a multilingual site and publish the map to the site.

You can use the `cec create-site map <site>` command to create a site map for a multilingual site on an Oracle Content Management server. For example:

```
cec create-site-map Site1 -u http://www.example.com/site1
```

This command traverses the a site structure, produces a site map hierarchy that matches the site page hierarchy, and creates a site map at the specified site URL on the Oracle Content Management server.

Command options follow:

```
  --url, -u         <url> Site
URL                                                  [required]
  --changefreq, -c  How frequently the page is likely to change
  --file, -f        Name of the generated site map file
```

```
  --publish, -p     Upload the site map to CEC server after creation
  --help, -h        Show
help
[boolean]
```

Valid values for the `<changefreq>` option follow:

- `always`
- `hourly`
- `daily`
- `weekly`
- `monthly`
- `yearly`
- `never`
- `auto`

Examples of the `cec create-site-map` command follow:

```
cec create-site-map Site1 -u http://www.example.com/site1

cec create-site-map Site1 -u http://www.example.com/site1 -f
sitemap.xml

cec create-site-map Site1 -u http://www.example.com/site1 -p

cec create-site-map Site1 -u http://www.example.com/site1 -c weekly -p
```

To publish a site map, a site update is created, the site map is updated, and then the update is committed.

Submit a Translation Job to a Language Service Provider

Content Toolkit provides a zip file that you can send to a Language Service Provider to start work on a translation job.

You can submit the translation job to the LSP through your translation connector. The submission takes awhile because the connector needs to unzip the file and submit all of the individual files to the LSP. Then the LSP can create a project for your translation job. Once the files have been imported into the project, you can start selecting files for translations. Then the LSP starts monitoring the status of the translations.

To check the status, list your translation jobs locally, using the `cec list-translation-jobs` command with no options. When the status of your job is `READY TO INGEST`, you can download a zip file from the LSP to ingest the translation job. The translation connector has submitted your zip file to the LSP, the LSP has translated the list of files, and the connector has retrieved the files back from the LSP, in a zip file that you can download and ingest.

```
cec list-translation-jobs
Local translation jobs:
```

```
Name                                    Status             Source Language
Target Languages
FridayDemo                              READY TO INGEST    en-US
de-DE,fr-FR
demoTest                                READY TO INGEST    en-US
de-DE,fr-FR
```

Ingesting the zip file pulls the translation job back from the connector, into your Content Toolkit.

```
cec ingest-translation-job FridayDemo
 - use connection <lsp name>
 - query translation connection to get job status
 - get translation
 - translation saved to <sites-toolkit folder>/cec-components/dist/
FridayDemo-translated.zip
 - validate translation file
 - translation job ingested to <sites-toolkit folder>/cec-components/src/
main/translationJobs/FridayDemo
```

After you ingest the zip file, when you list translation jobs locally, the status of your translation job is `TRANSLATED`.

```
cec list-translation-jobs
Local translation jobs:
Name                                    Status             Source Language
Target Languages
FridayDemo                              TRANSLATED         en-US
de-DE,fr-FR
demoTest                                READY TO INGEST    en-US
de-DE,fr-FR
```

You can upload the translated job to the Oracle Content Management server. Normally the job will go through an initial quick translation, which is sent back to you for review. Translation of a site can take a few weeks to finish, with ingestion of a translation job returned by the LSP, corrections to the translations, and resubmissions of the translation job.

Upload a Translation Job to the Server

After you ingest a translation job, you can upload it to the Oracle Content Management server and then check the translation on your site.

Use the `cec upload-translation-job` command to upload your translation zip file to the server.

```
cec upload-translation-job FridayDemo
 - created translation job zip file <sites-toolkit folder>cec-components/
dist/FridayDemo.zip
 - Logged in to remote server: <server-name>
 - file FridayDemo.zip uploaded to home folder, version 1
 - importing: percentage 5
 - importing: percentage 60
```

```
 - importing: percentage 60
 - import FridayDemo finished
```

After you upload your translation job, the status of the job on the server is `INPROGRESS`:

```
cec list-translation-jobs -s
Server: <server-name>
Asset translation jobs:
Name                                    Status        Source Language
Target Languages                        Pending Languages
testHash                                INPROGRESS    en-US
fr-FR,de-DE                             fr-FR,de-DE
Site translation jobs:
Name                                    Status        Source Language
Target Languages                        Pending Languages
demoTest                                TRANSLATED    en-US
de-DE,fr-FR
FridayDemo                              INPROGRESS    en-US
de-DE,fr-FR
```

To verify the translation, you can check the text strings in the assets on the site being translated.

About Toolkit PowerShell

Toolkit PowerShell in Oracle Content Management (OCM) allows power users to run CLI-like commands for listing or querying information about OCM objects in the web interface. The commands available in the web interface are a subset of the CLI utilities that are available in the OCM Content Toolkit.

Access to this feature requires users to have the OCM Developer role (so regular contributors will not be able to access the Toolkit PowerShell user interface).

To access the Toolkit PowerShell:

1. Click **Developer**.
2. Click **Open Toolkit PowerShell**.

On the Toolkit PowerShell page, you have a link in the upper right corner of the page to the Oracle Content Toolkit GitHub page, where you can download and install the toolkit CLI.

A drop-down menu lists the available commands. You can select a command from the drop-down menu, optionally specify additional arguments, and then click **Run** to execute the command. After you run a command, the output is displayed on the page. When you run more than one command, the output for each of the commands is displayed in new tabs.

For example, selecting the `list` command and running it will display all the resources in the system. Running the `describe-asset` command with the `--help` argument will display the help information describing this command.

- You can add and close tabs.
- You can click the links available in the output text to open up new tabs that describe the highlighted resource.

- The help option displays the help documentation for the selected command in a new browser tab.

Part VI
Appendixes

The following appendices are available:

- Tutorial: Develop Components with Knockout
- Troubleshoot

28

Tutorial: Develop Components with Knockout

This tutorial takes you through working with the set of JavaScript objects, which leverage the standard Knockout ViewModel and Template functionality, to create a component which is stored in the Oracle Content Management Component Catalog.

> **Note:**
>
> Oracle Content Management is moving away from Knockout JavaScript technology in favor of Mustache as the recommended default template system in JavaScript. If you are just starting out with Oracle Content Management, it is suggested you use Mustache, Preact, React, or other JavaScript UI technology in your component development.

Introduction and Prerequisites for Component Development with Knockout

This tutorial presents steps and verification procedures to create a sample component using JavaScript objects, which leverage the standard Knockout JS ViewModel and Template functionality.

You should be able to take the code referenced in these steps (provided in files that are seeded when you create a component) and update only the `.html` template and JavaScript `viewModel` with your own code.

> **Note:**
>
> While Oracle Content Management does not dictate the JavaScript technology you use to create components, typically the factory JavaScript function is the same for each implementation of a component in whichever JavaScript framework is chosen.

Prerequisites

This tutorial only focuses on the implementation of a component. For a more general information about components, see Develop Components.

To complete the steps in this tutorial, you must meet the following requirements:

- You must have access to an Oracle Content Management instance with permissions to create sites and components.
- The Oracle Content Management instance server has been synced to your local computer using the Oracle Content Management desktop or using a custom component. See Develop Custom Components with Developer Cloud Service.

In addition, you should be familiar with these JavaScript concepts and frameworks:

- JavaScript browser debugging
- JavaScript Closure
- JavaScript Asynchronous Module Definition (AMD) development
- RequireJS and KnockoutJS frameworks

Continue to Step 1: Create a Component.

Step 1: Create a Component

This step explains how to create your custom component in Oracle Content Management.

When you create a custom component, it must be registered to be usable by Oracle Content Management. To inform Oracle Content Management about your component, you use the Components page in Site Builder to register the component.

There are two types of components to register.

- **Local component**:
  - This is a component whose files are stored on the Oracle Content Management instance server.
  - The main advantage is that you don't have to worry about cross-domain or cross-protocol issues because the files are located with your site.
  - The disadvantage is that you can't execute any middle-tier logic in the Oracle Content Management server, so you must use REST APIs to remote servers that support CORS.
  - This type of component may be embedded into the page directly, or you can choose to use an inline frame to render the component on the page.
- **Remote component**:
  - A component where the files are stored on a remote server and you only register the URLs to the Render and Settings panel for the component.
  - A remote component provides an advantage if you have server-side logic that must execute when creating the content for your component.
  - The disadvantage is that you must ensure that any cross-domain and security issues are resolved for accessing those URLs.
  - Remote components always use an inline frame to render on the page.

To create and register a local component:

1. On the Oracle Content Management home page, click **Developer**.

   The **Developer** page is displayed.
2. Click **View all Components**.
3. From the menu, choose **Create Local Component**.
4. Enter a name for the component; for example, **A_Local_Component**.
5. Enter an optional description.
6. Click **Create**.

   After you have done this you'll see a component named `A_Local_Component` in your list of components.

Check the Results for Step 1

Now that you've successfully created a component, you should see the component in the Component palette for any site you create. Use these steps to validate your component creation:

1. Create a site named `localComponentTest`.
2. Select the site and click **Open**.
3. Click **Edit**.
4. Create an update for the site and give it a name and, optionally, a description.
5. Select a page on the site.
6. Click  in the side palette and select **Custom** to display the list of custom components.
7. Select `A_Local_Component` from the Custom component list and drag and drop it onto the page.

You should now see a default rendering for the local component you created.

8. Select  in the banner for the component you just dropped onto the page.
9. Select **Settings**.
10. Change the alignment and set the style for the component.
11. Close the Settings panel.

The following steps explain how the custom component is built and how to modify it for your own purposes. Continue to Step 2: Review the Structure of Your Local Component.

Step 2: Review the Structure of Your Local Component Rendering

In this step we review the structure of the default files created for a local component.

For a simple `Hello World` example, four JavaScript objects and the number of lines of code may seem like too much, but this is to provide you with the basis for building more complex components, as well as dealing with interaction with the Oracle Cloud Sites Service page lifecycle.

To review the structure of your local component:

1. On the Oracle Content Management home page, click **Developer**.

   The **Developer** page is displayed.
2. Click **View all Components**.
3. From the menu, choose **Create Local Component**.
4. Enter a name for the component; for example, **A_Local_Component**.
5. Enter an optional description.
6. Click **Create**.

   After you have done this you'll see a component named `A_Local_Component` in your list of components.

1. Using the Oracle Content Management desktop sync client, locate your component and sync it with the file system.

   If you don't have the desktop client, you can view all components and select the component in the Components page of the Oracle Content Management interface and drill down to see the files.
2. If you list the files under the component, you will see these files:

```
assets
    render.js
    settings.html
appinfo.json
_folder_icon.jpg
```

3. Open the `render.js` file under the `/assets` directory.

   The main points of the `render.js` file are:

- It is structured as a JavaScript AMD module so that it can be “required” into the page.
- It also includes references to KnockoutJS and JQuery that are already loaded as part of the Oracle Content Management page.

Consider the structure of the `render.js` file.

In the contents of the `render.js` file there are two JavaScript objects that implement the required Oracle Content Management component APIs: `sampleComponentFactory` and `SampleComponentImpl`. These objects are an example of an implementation for creating any KnockoutJS based components. The implementation of these objects will change based on the technology you use.

- `sampleComponentFactory`
  - This object is returned by the `render.js` AMD module.
  - This is a very simple Factory object and implements the single `createComponent()` interface.
  - More complex implementations may use the `args` value passed to return different implementations of the component based on the `viewMode` parameter. This enables you to have a significantly lighter-weight implementation of the component for runtime versus Site Builder.
- `SampleComponentImpl`
  - The main function within this object is the `render` function, which is used to render the component onto the page.

    To render the `Knockout` component into the page, the `render` function dynamically adds the template to the page, then applies the `viewModel` bindings to the template.
  - The rest of the implementation deals with initialization of the `viewModel` parameter and template, and with handling the messaging between the page and the component.

The last two objects in the `render.js` file, `sampleComponentTemplate` and `SampleComponentViewModel`, provide a custom implementation for the component. The implementation of these will differ based on your requirements.

- `sampleComponentTemplate`
  - This object provides the KnockoutJS template creation. It waits until the component has all the data initialized before attempting to display anything.
- `SampleComponentViewModel`
  - The `viewModel` retrieves the information stored by Oracle Content Management on behalf of the component, then selects how to appropriately lay out the component based on that data
  - General Knockout observables used by the template to handle access to the metadata stored on the component’s behalf:

    ```
    self.imageWidth = ko.observable('200px');
    self.alignImage = ko.observable();
    self.layout = ko.observable();
    self.showTopLayout = ko.observable();
    self.showStoryLayout = ko.observable();
    ```

  - Triggers and actions integration:

Trigger: A function to raise an Oracle Content Management trigger from the component that can be bound to actions of other components on the page.

```
self.imageClicked = function (data, event) {
  self.raiseTrigger("imageClicked"); // matches appinfo.json
};
```

Action: A function to handle the callback for when the component is told to execute an action with a given payload.

```
self.executeActionsListener = function (args) {
  // get action and payload
  var payload = args.payload,
  action = args.action;

  // handle 'setImageWidth' actions
  if (action && action.actionName === 'setImageWidth') {
    $.each(payload, function(index, data) {
      if (data.name === 'imageWidth') {
        self.imageWidth(data.value);
      }
    });
  }
};
```

Callback to execute any registered actions on demand.

```
SitesSDK.subscribe(SitesSDK.MESSAGE_TYPES.EXECUTE_ACTION,
$.proxy(self.executeActionsListener, self));
```

- Subscriptions to component life cycle:
  * Component initialization: Make sure the component doesn't render until all data has been fetched. This is handled through Knockout observables.

    ```
    self.componentLayoutInitialized = ko.observable(false);
    self.customSettingsDataInitialized = ko.observable(false);
    ```

    Get the initial values for any required properties. This is handled by callbacks to retrieve the data.

    ```
    SitesSDK.getProperty('componentLayout',
    self.updateComponentLayout);
    SitesSDK.getProperty('customSettingsData',
    self.updateCustomSettingsData);
    ```

  * Metadata updates: Callback whenever component metadata stored on the component's behalf is changed; for example, when the user invokes the Settings panel and updates the data.

    ```
    SitesSDK.subscribe(SitesSDK.MESSAGE_TYPES.SETTINGS_UPDATED,
    $.proxy(self.updateSettings, self));
    ```

> **Note:**
>
> Because the Oracle Content Management server always sets the mime-type for `.html` files, you cannot upload a `.html` file and use the required `"text!"` plug-in to load it. Therefore, for templates, you either need to use a different extension to load it using the `"text!"` plug-in, or load it in-line in the JavaScript directly as shown in the seeded data.

Check the Results for Step 2

You should now have an overview of how the structure of a custom component renderer is created. To validate that it works:

1. Update the `sampleComponentTemplate` object in the `render.js` file to change the following line. Change this code:

   ```
   '<!-- ko if: initialized -->'+
   ```

   Use this code instead:

   ```
   '<!-- ko if: initialized -->'+
   '<div data-bind="text:\'image width is: \' + imageWidth()"></div>' +
   ```

2. Sync or upload the component to the Oracle Content Management instance server.
3. Edit a page within the site and drop in the `A_Local_Component` custom component onto the page.

   At this point you should see `image width is: 260px` in the component.
4. Bring up the Settings panel and click the **Custom Settings** button.
5. Change the **image Width** field to `300px`.
6. At this point two things will happen in the component:
   a. The default image will expand from 260px to 300px in size.
   b. The text you added will update to `image width is 300px`.

Continue to Step 3: Review the Structure of Local Component Settings.

Step 3: Review the Structure of Local Component Settings

In this step we review the structure of the settings specified for a local component.

Similar to the `render.js` file in the `/assets` directory, there is a pre-created `settings.html` file in the same directory. The `settings.html` file renders any custom settings data for your component. In the default implementation, there is a single property `imageWidth` in the custom settings data.

To review the structure of your local component:

1. Using the Oracle Content Management desktop sync client, locate your component and sync it with the file system.

If you don't have the desktop sync client, you can select the component on the **Components** tab of the Oracle Content Management web interface and drill down to see the files.

2. If you list the files under the component, you'll see these files:

```
assets
    render.js
    settings.html
appinfo.json
_folder_icon.jpg
```

Open the `settings.html` file under the `/assets` directory and review the content. Unlike the `render.js` file, the `settings.html` file uses an inline frame in the Settings panel in Site Builder, which is why it also needs access to the supporting files to render correctly within the inline frame. Site Builder is needed to manage your site so any errors in your JavaScript code can be isolated from Site Builder, which is why the `settings.html` file uses an inline frame.

These are the main areas of the `settings.html` file:

- Knockout Template to render the Settings panel.

```
<!-- ko if: initialized() -->
<div class="scs-component-settings">
  <div>
    <!-- Width -->
    <label id="widthLabel" for="width" class="settings-heading"
data-bind="text: 'Image Width'"></label>
    <input id="width" data-bind="value: width"
placeholder="example: 200px or 33%" class="settings-text-box">
  </div>
</div>
<div data-bind="setSettingsHeight: true"></div>
<!-- /ko -->
```

- Custom Binding Handler to adjust the height of the inline frame once the Settings panel has been rendered.

```
ko.bindingHandlers.scsCompComponentImpl
```

- A Knockout ViewModel to apply to the Knockout Template.

```
SettingsViewModel
```

These are the main elements of `SettingsViewModel` :

- Subscriptions to component lifecycle.
- Component initialization:
  - Make sure the component doesn't render until all data has been fetched. This is handled through Knockout observables.

```
self.initialized = ko.observable(false);
```

- Make sure we don't attempt to update the data until we're ready.

```
self.saveData = false;
```

- Get the initial values for any required properties. This is handled by callbacks to retrieve the data.

```
SitesSDK.getProperty('customSettingsData', function (data) {
  //update observable
  self.width(data.width);

  // note that viewModel is initialized and can start saving data
  self.initialized(true);
  self.saveData = true;
});
```

- Save any property changes to the custom settings data.

```
self.save = ko.computed(function () {
  var saveconfig = {
    'width': isNaN(self.width()) ? self.width() : self.width() + 'px'
  };

  // save data in page
  if (self.saveData) {
    SitesSDK.setProperty('customSettingsData', saveconfig);
  }
}, self);
```

To add another property that you want to capture, several steps are required:

1. Update the user interface to display the new value.
2. Initialize the value to the current value stored against the component.
3. Save any changes to the value back to the component.

To add another property to your custom component, make these changes to the `settings.html` file:

1. Add another observable to handle the new property. Change this code:

```
self.width = ko.observable();
```

Use this code instead:

```
self.width = ko.observable();
self.imageBannerText = ko.observable();
```

2. Get any current value for the new property when the settings panel is first displayed. Change this code:

```
self.width(data.width);
```

Use this code instead:

```
self.width(data.width);
self.imageBannerText(data.imageBannerText);
```

3. Save any change to this new property. Change this code:

```
    'width': isNaN(self.width()) ? self.width() : self.width() +
'px'
```

Use this code instead:

```
'width': isNaN(self.width()) ? self.width() : self.width() + 'px',
'imageBannerText': self.imageBannerText()
```

4. Add a user interface to display the new field. Change this code:

```
<label id="widthLabel" for="width" class="settings-heading" data-
bind="text: 'Image Width'"></label>
<input id="width" data-bind="value: width" placeholder="example:
200px or 33%" class="settings-text-box">
```

Use this code instead:

```
<label id="widthLabel" for="width" class="settings-heading" data-
bind="text: 'Image Width'"></label>
<input id="width" data-bind="value: width" placeholder="example:
200px or 33%" class="settings-text-box">

<label id="imageBannerTextLabel" for="imageBannerText"
class="settings-heading" data-bind="text: 'Image Banner'"></label>
<input id="imageBannerText" data-bind="value: imageBannerText"
placeholder="Text to display above an image" class="settings-text-
box">
```

5. Sync or upload the `settings.html` file.

If you were to run this now, the field would display. However, the size of the Settings panel doesn't change automatically. Because you increased the size of the panel, you also must update the `components.json` registration entry to the new size.

1. Download the `appinfo.json` file, which is at the same level as the `assets/` directory for you component, and update the size of the settings panel. Change this code:

```
"settingsHeight": 90,
```

Use this code instead:

```
"settingsHeight": 160,
```

2. Sync or upload the `appinfo.json` file.

Check the Results for Step 3

You should now be able to see and enter the new property you added to the Settings panel.

1. Refresh your page in your site so Site Builder can pick up changes to the component.
2. Take the page into Edit mode.
3. Drag and drop your component onto the page.
4. Bring up the Settings panel against your component.
5. Click the **Custom Settings** button.

   You will see two fields displayed for each of the properties you have in your `settings.html` file.

Continue to Step 4: Display the New Property in the Component.

Step 4: Display the New Property in the Component

At the end of this section, you will be able to enter a value for a new property in the Settings panel and see the custom component change to reflect the new value. Updates to the property will also automatically be saved for you with the page.

In the `render.js` file, you must update two JavaScript objects in the component:

- `SampleComponentViewModel`
- `sampleComponentTemplate`

Edit `render.js` and update the `SampleComponentViewModel` component to include the new property. Change this property:

```
self.showStoryLayout = ko.observable();
```

Use this instead:

```
self.showStoryLayout = ko.observable();
self.imageBannerText = ko.observable();
```

Update `SampleComponentViewModel` to get any change in values. Change this property:

```
self.imageWidth(customData && customData.width);
```

Use this instead:

```
self.imageWidth(customData && customData.width);
self.imageBannerText(customData && customData.imageBannerText);
```

Change `sampleComponentTemplate` to display the new property. Change this property:

```
'<div data-bind="text: \'image width is: \' + imageWidth()"></div>' +
```

Use this instead:

```
'<div data-bind="text: imageBannerText"></div>' +
```

Sync or upload the component to the Oracle Content Management server.

You've now altered the component to display the new property. Unlike the Settings panel that is embedded in an inline frame on the page, because the component is inserted directly into the page, as it grows in size the area available to it will automatically increase.

Check the Results for Step 4

To see the new property displayed:

1. Refresh your page in your site to Site Builder can pick up changes to the component.
2. Take the page into Edit mode.
3. Drag and drop your component onto the page.
4. Bring up the Settings panel against your component.
5. Click the **Custom Settings** button.
6. Change `Image Banner` to `Workspace`.

   You will see the component update on the page to `Worksapce` appear above the image.

Continue to Step 5: Register Triggers.

Step 5: Register Triggers

In this step, you will review how an Oracle Content Management trigger can been registered, which you can select using the Trigger Actions option under the Link tab in the Settings panel for your component.

Triggers are part of Oracle Content Management intercomponent communication. Any component can raise any number of triggers. The component can provide a payload for a trigger, which is then passed to any action that is executed when the trigger is raised. Users can select what actions should be executed for each trigger. Finally, components that are built to work together can automatically raise triggers to execute actions on the other component without the user needing to define the interaction between the components.

For components you add, triggers are registered as part of the registration data for the component. To add a trigger, update the `"triggers"` property array with each trigger the component supports. You also must specify the payload the trigger supports so that the user interface can be created to allow the user to map values within the payload to properties supported by the action.

Open the `appinfo.json` file and review the `"triggers":[],` entry.

```
"triggers": [{
    "triggerName": "imageClicked",
    "triggerDescription": "Image clicked",
    "triggerPayload": [{
```

```
      "name": "payloadData",
      "displayName": "Trigger Payload Data"
   }]
}],
```

In this entry, you'll see the following:

- A triggerName, `"imageClicked"`, which should be a unique value, and will be typically namespaced by your custom component ID.
- A triggerDescription, `"Image clicked"`, which is used by the user interface dialog to display your trigger.
- A single value triggerPayload, `"payloadData"`, for your trigger. Users will be able to select entries in this payload and map them to fields in the action.

Check the Results for Step 5

You can see and select your trigger when you go to the **Link** tab in the Settings panel for your component:

1. Refresh your page in your site so Site Builder can pick up changes to the component.
2. Take the page into Edit mode.
3. Drag and drop your component onto the page.
4. Bring up the Settings panel against your component.
5. Select the **Link** tab at the top of the Settings panel.
6. Click **Trigger Actions** as the Link Type.
7. Click the **Image clicked** trigger that you registered.
8. In the dialog, drag the **Show Alert** action from within the Page Actions section. (Page Actions are built-in actions supplied by Oracle Content Management.)
9. In the **Message** field, select the **Trigger Payload Data** value, which is the name of the entry in the payload you saw when you registered the trigger. You can change this to any name you want.

Now you're able to register a trigger and map the trigger to a built-in action, passing through a value. In the next step we'll review how the trigger is raised to execute the action.

Continue to Step 6: Raise Triggers.

Step 6: Raise Triggers

In this step, we'll show you how the trigger you saw registered is raised.

Triggers can be raised at any point by a component. Typically it is raised by a user interaction, by clicking a button or selecting a row in a table. However, the component can raise the trigger based on any criteria; for example, when data changes because of a REST call.

For this sample, when you click the image it will raise a trigger passing through the current value of the `whoAreYou` property.

Review the `render.js` file and look at the `SampleComponentViewModel` object.

To raise a trigger:

1. Review the function in the `SampleComponentViewModel` object that calls the Sites SDK to raise the trigger.

```
self.raiseTrigger = function (triggerName) {
  SitesSDK.publish(SitesSDK.MESSAGE_TYPES.TRIGGER_ACTIONS, {
    'triggerName': triggerName,
    'triggerPayload': {
        'payloadData': 'some data here'
    }
  });
};
```

2. Now you need something in the user interface to call the function to raise the trigger. Review the `render.js` file and update the `sampleComponentTemplate` object to have this entry:

```
     '<div data-bind="attr: {style: imageStyle, \'data-layout\':
alignImage()}, click: imageClicked">' +
```

In the `SampleComponentViewModel` object, you see the JavaScript function that is called when the image is clicked. This function calls the Sites SDK to tell it to trigger all the actions defined for the trigger `"imageClicked"`, which is the value passed in from the `click` binding in step 2. It also passes through a `triggerPayload` that has a single field:`payloadData` and passes through a static value `'some data here'`. These values `imageClicked` and `whoAreYou` match those in the `appinfo.json` file where the trigger is registered (in the previous step).

In the sample code, the trigger is raised by a data-bind of the `click` binding and passes in the trigger name `imageClicked`. There are currently three renderings of the `<scs-image>` component based on the layout the user chooses. To ensure that the trigger is raised for each of the layouts, edit the `render.js` file to make the following changes.

- Raise triggers from different layouts. Find the two entries of this code:

```
'<div data-bind="attr: {style: imageStyle, \'data-layout\':
alignImage()}">' +
```

  Change the code to this:

```
'<div data-bind="attr: {style: imageStyle, \'data-layout\':
alignImage()}, click: imageClicked">' +
```

- Specify the payload to pass to the triggers. Change this code:

```
self.raiseTrigger = function (triggerName) {
  SitesSDK.publish(SitesSDK.MESSAGE_TYPES.TRIGGER_ACTIONS, {
    'triggerName': triggerName,
    'triggerPayload': {
      'payloadData': 'some data here'
    }
  });
};
```

Use this code instead:

```
    self.raiseTrigger = function (triggerName) {
      SitesSDK.publish(SitesSDK.MESSAGE_TYPES.TRIGGER_ACTIONS, {
        'triggerName': triggerName,
        'triggerPayload': {
          'payloadData': self.imageBannerText() // pass banner text as
payload
        }
      });
    };
```

- Sync or upload the `render.js` file to the Oracle Content Management instance server.

Now that you've reviewed the required code, you can hook up the trigger so that your custom component will raise it when the button is clicked.

Check the Results for Step 6

You should now be able to register an action to execute against your trigger and also have the action execute when the trigger is raised:

1. Refresh your page in your site so Site Builder can pick up changes to the component.
2. Take the page into Edit mode.
3. Drag and drop your component onto the page.
4. Bring up the Settings panel against your component.
5. Select the **Link** tab at the top of the Settings panel.
6. Select **Trigger Actions** as the Link Type.
7. Click the **imageClicked** trigger that you saw registered.
8. In the dialog, drag the **Show Alert** action from the **Page Actions** section.
9. In the **Message** field, select the **payloadData** value, which is the payload you entered when you registered the trigger.
10. Close the Settings panel and switch Site Builder to Preview mode.
11. Click the image in the component.

    An alert will appear showing `no message defined` because you haven't specified the `imageBannerText` value.
12. Take the page into Edit mode and bring up the Settings panel again for the component.
13. Click **Custom Settings** and enter `Workplace`.
14. Close the Settings panel and switch the page to Preview mode.
15. Click the image in the component.

    Now it should show the updated payload `Workplace`, which is invoked from the change you made to the `click` binding.

You can execute any number of actions when a trigger is raised.

> **Note:**
>
> There is no pre-defined order to when an action is executed. Although each action will be called in the order it is listed, there is no wait for it to complete before the next action is called. If an action makes an asynchronous call, it may not complete before the next action is executed.

Continue to Step 7: Register Actions.

Step 7: Register Actions

Oracle Content Management actions are called on components when triggers are raised.

A component can register any number of actions and also define the payload the action supports. When a user selects an action, they can populate the payload to be passed to the action.

As with registering triggers, you can register actions that your component supports in the `appinfo.json` file registration data. To review the registration of the sample action in your component, open the `appinfo.json` file and find the `"actions"` code.

```
"actions": [{
  "actionName": "setImageWidth",
  "actionDescription": "Update the image width",
  "actionPayload": [{
    "name": "imageWidth",
    "description": "Image Width in pixels",
    "type": {
      "ojComponent": {
        "component": "ojInputText"
      }
    },
    "value": ""
  }]
}]
```

This registered action will be visible in the Action dialog that's invoked when you click a trigger in the **Link** tab in the Settings panel for the component.

Check the Results for Step 7

1. Refresh your page in your site so Site Builder can pick up the changes to the component.
2. Take the page into Edit mode.
3. Drag and drop your component onto the page.
4. Drop a Button component onto the page.
5. Bring up the Settings panel against the Button component.
6. On the General tab, change the label of the button to **Click me!**.
7. Select the **Link** tab on the Settings panel.

8. Select **Trigger Actions** as the **Link Type**.
9. Click the **Click on Button** trigger against the Button component.
10. In the dialog, expand the `A_Local_Component` component in the left side palette.
11. Drag and drop the **Update the image width** action from the `A_Local_Component` component onto the page.
12. Enter **300px** in the **Image width in pixels** field.

You've now seen how you can register an action and how that action will show up in the user interface. In the next step you will learn how to handle an action within your component when it is called.

Continue to Step 8: Execute Actions.

Step 8: Execute Actions

At the end of this topic, you'll be able to drop components on the page that execute actions within your component. This leverages the action registration you created in the previous step.

For a component to execute an action, it must listen for the `EXECUTE_ACTION` message. This message also includes the payload passed to the action from which you will extract the expected values.

To listen for the `EXECUTE_ACTION` message, edit the `render.js` file and update the `SampleComponentViewModel` object with the following entry:

```
SitesSDK.subscribe('EXECUTE_ACTION', $.proxy(self.executeActionsListener,
self));
```

When the `EXECUTE_ACTION` message is received, the associated callback function is executed:

```
self.executeActionsListener = function (args) {
  // get action and payload
  var payload = args.payload,
  action = args.action;

  // handle 'setImageWidth' actions
  if (action && action.actionName === 'setImageWidth') {
    $.each(payload, function(index, data) {
      if (data.name === 'imageWidth') {
        self.imageWidth(data.value);
      }
    });
  }
}
```

This creates a JavaScript function to execute the action, then uses the Sites SDK to call the function whenever the `EXECUTE_ACTION` message is raised.

The component will be called whenever an `EXECUTE_ACTION` message is raised, and it is up to the component to only handle actions it is designed to handle. To do this, you must check the name of the action to ensure it is one the component can handle.

The payload for the action is an array of values. Typically, you will need to find the payload values you care about from the array.

> **Note:**
>
> Because the action listener is a callback, you should use JavaScript Closure or appropriately bind the function to ensure you have access to your viewModel when the function is executed.

Check the Results for Step 8

1. Refresh your page in your site so Site Builder can pick up changes to the component.
2. Take the page into Edit mode.
3. Drag and drop your component onto the page.
4. Drag and drop a Button component onto the page.
5. Bring up the Settings panel against the Button component.
6. On the General tab, change the Label of the button to `Click me!`
7. Select the **Link** tab at the top of the Settings panel.
8. Select **Trigger Actions** as the Link Type.
9. Click the **Click on Button** trigger against the Button component.
10. In the dialog, expand the `A_Local_Component` component in the left side.
11. Drag and drop the **Update the image width** action from the `A_Local_Component` component onto the right side.
12. Enter `300px` in the **Image Width in pixels** field.
13. Switch the page to Preview mode.
14. Click the **Click me!** button.

At this point the size of your image will increase to `300px`.

> **Note:**
>
> Triggers and actions are designed to support inter-component communication. They are not designed to create or manage state. If you refresh the page, the page will revert to its original state as if no triggers were raised or actions executed.

Continue to Step 9: Create a Distinct Title for Each Instance of the Component.

Step 9: Create a Distinct Title for Each Instance of the Component

This step explains how to create distinct titles for different instances of your component.

When you drop your component onto the page, you will have noticed the banner for your component reads: `A_Local_Component`. While this is fine if the user only drops one of your components onto the page, you may want to create distinct titles so the user can distinguish between different instances of your component.

You can use the Sites SDK to update the title for the component. In this step, you will update it based on the `"imageBannerText"` property.

To update the title, edit the `render.js` file and add this code to your `SampleComponentViewModel` object:

```
self.updateDescription = ko.computed(function () {
  SitesSDK.setProperty('description', self.imageBannerText());
});
```

This Knockout computation will update the description for your component whenever the `imageBannerText` observable changes.

Check the Results for Step 9

1. Refresh your page in your site so Site Builder can pick up changes to the component.
2. Take the page into Edit mode.
3. Drop your component onto the page.
4. Bring up the Settings panel against your component.
5. Click the **Custom Settings** button.
6. Change the **Image Banner** to **Workplace**.
7. Close the Settings panel and hover your cursor over your component to show the banner.

   You should now see `A_Local_Component Workplace` displayed.

Continue to Step 10: Use Nested Components with In-Line Editing.

Step 10: Use Nested Components with In-Line Editing

Oracle Content Management components are implemented using KnockoutJS component architecture. This means that if you are using KnockoutJS to implement your components, you can include Oracle Content Management built-in components directly into your template.

> **Note:**
>
> Because Oracle Content Management built-in components can only run in the Oracle Content Management page, you can't use nested components if your component is rendered in an inline frame.

To leverage nested components:

1. Implement your component using KnockoutJS.
2. Use RequireJS to include your component and use the same Knockout "ko" instance variable that is created by Oracle Content Management.

   This is required because Oracle Content Management extends Knockout with components and these components won't be available if you use your own instance of KnockoutJS.

In this step you will review how the Oracle Content Management Image, Paragraph, and Title components are rendered in your custom component. A user will be able to edit it directly within the page and access the Settings panel for the nested component.

To see how these components are included in your template, edit the `render.js` file and look at the `sampleComponentTemplate` object. The default section that is rendered is shown here:

```
'<!-- ko if: alignImage() !== \'right\' -->' +
'<div style="display:flex;">' +
'<div data-bind="attr: {style: imageStyle, \'data-layout\':
alignImage()}, click: imageClicked">' +
'<scs-image params="{ scsComponent: { \'renderMode\': mode,
\'parentId\': id, \'id\': \'imageId\', \'data\': imageData } }"></scs-
image>' +
'</div>' +
'<div data-bind="attr: {style: paragraphStyle}">' +
'<scs-title params="{ scsComponent: { \'renderMode\': mode,
\'parentId\': id, \'id\': \'titleId\', \'data\': titleData } }"></scs-
title>' +
'<scs-paragraph params="{ scsComponent: { \'renderMode\': mode,
\'parentId\': id, \'id\': \'paragraphId\', \'data\':
paragraphData } }"></scs-paragraph>' +
'</div>' +
'</div>' +
'<!-- /ko -->' +
```

Looking at the `<scs-image>` nested component, you will see the following entry:

```
'<scs-image params="{ scsComponent: { \'renderMode\': mode,
\'parentId\': id, \'id\': \'imageId\', \'data\': imageData }}"></scs-
image>' +
```

The `scsComponent` data passed to the `params` template binding includes the following:

- `renderMode`: This refers to the mode Site Builder is in. You can use this to enable and disable features. For example, when used by the `<scs-title>` component, it adds the rich text editor when running in `edit` mode.
- `parentId`: This is required so the Oracle Content Management component knows that it is rendering as a nested component. All changes to the nested component will be saved in the data for the custom component.
- `id`: A unique ID for the nested component. This will be further namespaced by the ID for the custom component.

- `data`: Initial data for the nested component. If the component isn't further modified, it will render with this initial data.

The referenced `id` and `mode` values are passed in to your custom component in the `SampleComponentViewModel` object, so you don't need to modify the object to get these values:

```
// Store the args
self.mode = args.viewMode;
self.id = args.id;
```

The syntax for all the other supported nested components follows the same pattern as for `<scs-paragraph>`; for example: `<scs-image>, <scs-title>, <scs-button>`.

Check the Results for Step 10

1. Refresh your page in your site so Site Builder can pick up changes to the component.
2. Take the page into Edit mode.
3. Drag and drop your component onto the page.
4. Click the `As a page author, you can edit. . .` text in your component and update the description using the rich text editor.
5. Switch to Preview mode to see your update.
6. Switch back to Edit mode.
7. Bring up the Settings panel against your component.
8. Click the **Components** link that now appears, because it found your nested component.
9. Click **Paragraph**, which is the nested component it found.

You can now update the properties against the Paragraph component within your component.

> **Note:**
>
> Until the component has been instantiated, Oracle Content Management does not know about any nested components that may exist in the template. To tell Oracle Content Management about hidden nested components, you can use the `SitesSDK.setProperty('visibleNestedComponents', []);` API. To have hidden nested components show-up by default, you must update the `"nestedComponents": []` array in the component registration.

Continue to Step 11: Support Different Layouts.

Step 11: Support Different Layouts

In this step we will review layouts that allow the user to alter how the component displays.

A custom component can support any number of layouts that you want to allow the user to choose. Each of these layouts will alter how the custom component displays. Layouts are another extension to the registration data.

To review the three layouts supported in the sample code, review the `"componentLayouts"` entry in the `appinfo.json` file

```
"componentLayouts": [
  {
    "name": "default",
    "displayName": "IMAGE_LEFT_LAYOUT"
  },
  {
    "name": "right",
    "displayName": "IMAGE_RIGHT_LAYOUT"
  },
  {
    "name": "top",
    "displayName": "IMAGE_TOP_LAYOUT"
  }
],
```

If you bring up the Settings panel against the custom component, you will see an option to switch between layouts. To enable your component to react to the change in selection, the render.js file has code to get the currently selected value and listen for changes to this value.

Edit the `render.js` file and look at the `SampleComponentViewModel` object.

- There is a `layout` observable, which is referenced in the template:

```
self.layout = ko.observable();
```

- There is an update function to handle whenever this value changes:

```
    self.updateComponentLayout = $.proxy(function (componentLayout)
{
      var layout = componentLayout ? componentLayout : 'default';
      self.layout(layout);
      self.alignImage(layout === 'right' ? 'right' : 'left');
      self.showTopLayout(layout === 'top');
      self.showStoryLayout(layout === 'default' || layout ===
'right');

      self.componentLayoutInitialized(true);
    }, self);
```

- The initialization code gets the original value for the layout and calls the update function:

```
SitesSDK.getProperty('componentLayout', self.updateComponentLayout);
```

  The property change listener checks for any changes to this property and calls the update function:

```
self.updateSettings = function (settings) {
  if (settings.property === 'componentLayout') {
    self.updateComponentLayout(settings.value);
```

```
    } else if (settings.property === 'customSettingsData') {
      self.updateCustomSettingsData(settings.value);
    }
  };

  SitesSDK.subscribe(SitesSDK.MESSAGE_TYPES.SETTINGS_UPDATED, $.proxy(self.u
  pdateSettings, self));
```

Finally the `sampleComponentTemplate` template object has code to reflect changes in this value:

```
'<!-- ko if: alignImage() === \'right\' -->' +
```

Together, these changes allow you to select your layout in the Settings panel and have the component update.

Check the Results for Step 11

1. Refresh your page in your site so Site Builder can pick up changes to the component.
2. Take the page into Edit mode.
3. Drag and drop your component onto the page.
4. Bring up the Settings panel against your component.
5. Select **Image Right** from the Layout property.

   At this point the component will update to show the `"<scs-image>"` component.

Continue to Step 12: Define Custom Styles.

Step 12: Define Custom Styles

Components you create are treated like any other component in the `design.json` and `design.css` files in the theme used for your site.

To add your own style for your custom component, confirm the `id` value you used when you registered your component. In the `appinfo.json` file; this was `"id": "hello-world"`.

Using that value, edit the theme's `design.json` file and add in the new styles you want to support against that `id`. For example, edit the `/designs/default/design.json` file in your theme and add this code:

```
"hello-world": {
  "styles": [{
    "name": "Plain",
    "class": "hello-world-default-style"
  },
  {
    "name": "Gothic",
    "class": "hello-world-gothic-style"
  }]
},
```

If you bring up the Settings panel against your component, you should now see **Plain (default)** and **Gothic** as the two options listed in the Style tab. However, switching between these options will not do anything until you actually define the style classes listed in the `design.css` file.

Edit the theme's `design.css` file and add in the cascading style sheet (CSS) classes of your style. For example, edit the `/designs/default/design.css` file in your theme and add this code:

```
.hello-world-default-style .scs-component-content {
  font-family: "Helvetica Neue", "Helvetica", "Arial", sans-serif;
  font-size: 24px;
  font-weight: normal; }

.hello-world-gothic-style .scs-component-content {
  font-family: "Century Gothic","CenturyGothic","AppleGothic",sans-
serif;
  font-size: 32px;
  font-weight: bold; }
```

Save and sync your files to the Oracle Content Management instance server.

Check the Results for Step 12

1. Refresh your page in your site so Site Builder can pick up the changes to the component.
2. Take the page into Edit mode.
3. Drag and drop your component onto the page.
4. Bring up the Settings panel against your component.
5. Go to the Style tab.
6. Switch between the **Gothic** and **Plain** styles that were defined in your `design.json` file.

   You'll notice that the font size within your component adjusts to reflect the changes as it switches between the applied CSS class for each selection.

Continue to Step 13: Render a Component in an Inline Frame.

Step 13: Render a Component in an Inline Frame

The sample so far has shown a local component rendered in-line in the page. You can also choose to render a component in an inline frame.

For example, you may choose to render a component in an inline frame if your component does non-omnipotent updates to the page, which requires you to re-create the page whenever properties change. In addition, remote components are always rendered in an inline frame.

The samples in this section are taken from the files created for you when you choose the **Create a component that renders in an iframe** option when creating a local component. You can, however, take these set of files and host them on your remote server so they apply equally to remote components.

Similarities Between Inline Frame and Non-Inline Frame Components

Settings Panel

Because the Settings panel is always placed into the page in an inline frame, the code for the Settings panel doesn't change regardless of whether the component uses an inline frame or not. You'll create the same Settings panel code for both use cases.

Sites SDK API

The SDK API is the same for both use cases. You'll use the same code to raise triggers, listen for actions, and get and set property values. While certain properties may not be applicable in both cases (for example, you can't set the `"height"` property for a component that doesn't use an inline frame) the API remains the same. Therefore, you can copy the code between both of these types of components and the example code discussed in this tutorial will work for both cases.

Differences Between Inline Frame and Non-Inline Frame Components

File Structure and Dependencies

When you select **Create a component that renders in an iframe** when creating a local component, you will see the following files created for you:

```
<component name>
    assets
        css
            app-styles.css
        js
            jquery.mn.js
            knockout.mn.js
            sites.min.js
        render.html
        settings.html
    appinfo.json
    _folder_icon.jpg
```

These files are created to allow you to immediately run your component in an inline frame on the page. The main differences between this structure and that of a standard local component are:

- JavaScript dependencies:
  - You are getting a complete copy of these files so your component will run. These files are required for the sample inline frame component to run. You can add and remove the content of this directory based on your requirements.
  - Because everything under the `assets` directory for your component is pushed to a public site when the component is published, everything in the `js` directory will be available both in Site Builder and at runtime.
  - Note: These files are created for ease of use. You should look at consolidating these files in the theme or in another public location rather than create separate versions of these files for each of your inline frame components.
- `render.html`:

  - This is a full HTML document as opposed to the `render.js` file for standard components, which is an AMD module.

Component "Height" Management

One of the issues in using an inline frame is the height management of the inline frame itself. If you get this wrong, you will see scroll bars appearing for the component on the page, which you may or may not want.

In order to manage the height of the inline frame, the component must tell the page how tall it wants the inline frame to be. With remote components, you may be dealing with cross-domain issues, so you must use Sites SDK messaging to ask the page to set the inline frame to the required height after the component has rendered on the page. This is done by using the `SitesSDK.setProperty('height', {value})` API. (SeeOracle Content and Expeience SDKs.)

For example, create the `setHeight` function and a custom binding handler to call it when the component has rendered on the page.

- Update height function:

```
// set the height of the iFrame for this App
self.setHeight = function () {
// use the default calculation or supply your own height value as a
second parameter
SitesSDK.setProperty('height');
};
```

- Knockout custom binding handler to call `setHeight` whenever the component is rendered on the page or a property changes:

```
ko.bindingHandlers.sampleAppSetAppHeight = {
  update: function (element, valueAccessor, allBindings, viewModel,
bindingContext) {
    // create dependencies on any observables so this handler is
called whenever it changes
    var imageWidth = viewModel.imageWidth(),
        imageUrl = viewModel.imageUrl(),
        titleText = viewModel.titleText(),
        userText = viewModel.userText();

  // re-size the iFrame in the Sites page now the template has
rendered
  // Note: If you still see scrollbars in the iframe after this, it
is likely that CSS styling in your app is the issue
  viewModel.setHeight();
 }
};
```

- Template update to call binding handler:

```
<div data-bind="sampleAppSetAppHeight: true"></div>
```

Trigger and Action Registration

While the trigger/action registration for components that are not in inline frames is located in the `appinfo.json` file, for inline frame components, the component itself is responsible for providing this information. This is done using these two APIs:

```
SitesSDK.subscribe('GET_ACTIONS', self.getAppActions);
SitesSDK.subscribe('GET_TRIGGERS', self.getAppTriggers);
```

Here's an example of using these APIs.

```
// Register TRIGGERS meta-data
SampleAppViewModel.prototype.getAppTriggers = function (args) {
  var triggers = [{
    "triggerName": "imageClicked",
    "triggerDescription": "Image clicked",
    "triggerPayload": [{
      "name": "payloadData",
      "displayName": "Trigger Payload Data"
    }]
  }];

  return triggers;
};

// Register ACTIONS meta-data
SampleAppViewModel.prototype.getAppActions = function (args) {
  var actions = [{
    "actionName": "setImageWidth",
    "actionDescription": "Update the image width",
    "actionPayload": [{
      "name": "imageWidth",
      "description": "Image Width in pixels",
      "type": {
        "ojComponent": {
        "component": "ojInputText"
        }
      },
      "value": ""
    }]
  }];

  return actions;
};
```

Access to Theme Styles

Because the component is rendered in an inline frame, it doesn't have access to the styles available in the theme. The Sites SDK provides an API to retrieve these styles so they can be applied to elements within the inline frame.

This topic is explored more in Step 14: Use Custom Styles When the Component is Rendered in an Inline Frame.

Mixed HTTPS Versus HTTP Protocol

Because Oracle Content Management uses the HTTPS protocol, all resources referenced within the page also must use `HTTPS`. Resources include the base `.html` file that will be rendered in the inline frame along with all the files that it references.

This resource requirement applies mostly to remote components, however, you must be aware of this constraint. Resources for local components using inline frames are provided by the Oracle Content Management server, so these components already use a matching protocol.

Continue to Step 14: Use Custom Styles When the Component is Rendered in an Inline Frame.

Step 14: Use Custom Styles When the Component Is Rendered in an Inline Frame

Components rendered in an inline frame don't have direct access to the `design.css` file. Instead there is an additional step to get the URL for the `design.css` in your component and add it to the page. You then must update your component to reflect the user-selected style.

To include and use the `design.css` file in your component requires changes in the `render.html` file:

1. Locate and include the URL to the `design.css` file
2. Get the value of the select style class whenever it changes
3. Update the template to reflect the `styleClass` selected
4. Reflect changes to the selected style class in your component
5. Make sure the inline frame resizes when the style changes

Here are the detailed instructions for editing the `render.html` file:

1. Locate and include the URL to the `design.css` file.

   Dynamically add the `design.css` file to the `<head>` section of the page. After it has been loaded, set the height of the inline frame because it may have been altered by applying the styles.

   Add the following code into the `viewModel` object:

```
// Dynamically add any theme design URL to the <head> of the page
self.loadStyleSheet = function (url) {
    var $style,
        styleSheetDeferred = new $.Deferred(),
        attempts = 100,
        numAttempts = 0,
        interval = 50,
        pollFunction = function () {
            // try to locate the style sheet
            for (var i = 0; i < document.styleSheets.length; i++) {
                try {
                    // locate the @import sheet that has an href
based on our expected URL
                    var sheet = document.styleSheets[i],
```

```
                        rules = sheet && sheet.cssRules,
                        rule = rules && rules[0];
                    // check whether style sheet has been loaded
                    if (rule && (rule.href === url)) {
                        styleSheetDeferred.resolve();
                        return;
                    }
                } catch (e) {}
            }
            if (numAttempts < attempts) {
                numAttempts++;
                setTimeout(pollFunction, interval);
            } else {
                // didn't find style sheet so complete anyway
                styleSheetDeferred.resolve();
            }
        };

    // add the themeDesign stylesheet to <head>
    // use @import to avoid cross domain security issues when determining
when the stylesheet is loaded
    $style = $('<style type="text/css">@import url("' + url + '")</
style>');
    $style.appendTo('head');

    // kickoff the polling
    pollFunction();

    // return the promise
    return styleSheetDeferred.promise();
};

// update with the design.css from the Sites Page
SitesSDK.getSiteProperty('themeDesign', function (data) {
    if (data && data.themeDesign && typeof data.themeDesign === 'string')
{
        // load the style sheet and then set the height
        self.loadStyleSheet(data.themeDesign).done(self.setHeight);
    }
});
```

2. Get the value of the select style class whenever it changes.

   Create an observable to track when the value of the `styleClass` property changes. :

   ```
   self.selectedStyleClass = ko.observable();
   ```

   Note that we can't render until we have the style class. Change this code:

   ```
   self.customSettingsDataInitialized = ko.observable(false);
   self.initialized = ko.computed(function () {
       return self.customSettingsDataInitialized();
   }, self);
   ```

Use this code instead:

```
self.customSettingsDataInitialized = ko.observable(false);
self.styleClassInitialized = ko.observable(false);
self.initialized = ko.computed(function () {
    return self.customSettingsDataInitialized() &&
self.styleClassInitialized();
}, self);
```

Get the initial value for the selected style class by adding:

```
self.updateStyleClass = function (styleClass) {
    self.selectedStyleClass((typeof styleClass === 'string') ?
styleClass : 'hello-world-default-style'); // note that this 'hello-
world' prefix is based on the app name
    self.styleClassInitialized(true);
};
SitesSDK.getProperty('styleClass', self.updateStyleClass);
```

3. Update the template to reflect the `styleClass` . Change this code:

```
<p data-bind="attr: {id: 'titleId'}, text: titleText"></p>
```

Use this code instead:

```
<p data-bind="attr: {id: 'titleId'}, text: titleText, css:
selectedStyleClass"></p>
```

4. Reflect changes to the selected style class in your component. Change this code:

```
if (settings.property === 'customSettingsData') {
    self.updateCustomSettingsData(settings.value);
}
```

Use this code instead:

```
if (settings.property === 'customSettingsData') {
    self.updateCustomSettingsData(settings.value);
}
if (settings.property === 'styleClass') {
    self.updateStyleClass(settings.value);
}
```

5. Make sure the inline frame re-sizes when the style changes. Change this code:

```
// create dependencies on any observables so this handler is called
whenever it changes
var imageWidth = viewModel.imageWidth(),
      imageUrl = viewModel.imageUrl(),
      titleText = viewModel.titleText(),
      userText = viewModel.userText();
```

Use this code instead:

```
// create dependencies on any observables so this handler is called
whenever it changes
var imageWidth = viewModel.imageWidth(),
      imageUrl = viewModel.imageUrl(),
      titleText = viewModel.titleText(),
      userText = viewModel.userText(),
      selectedStyleClass = viewModel.selectedStyleClass();
```

6. Save and sync your files to the Oracle Content Management instance server.

Check the Results for Step 14

1. Refresh your page in your site so Site Builder can pick up changes to the component.
2. Take the page into Edit mode.
3. Drag and drop your component onto the page.
4. Bring up the Settings panel against your component.
5. Go to the Style tab.
6. Switch between **Gothic** and **Plain** styles defined in your `design.json` file.

   You'll notice that the font size within your component adjusts to reflect the changes as it switches between the applied CSS class for each selection.

Continue to Step 15: Integration with the Page Undo and Redo Behavior.

Step 15: Integration with the Page Undo and Redo Behavior

Because Oracle Content Management stores properties on behalf of the custom component, changes to those properties are automatically part of the page's **Undo** and **Redo** behavior.

To ensure that it is clear what is happening when a user clicks **Undo** or **Redo**, these "undo events" should only happen when a user has actually done something to the page. For example, bringing up the custom component Settings panel should not update the properties within the page until the user actually makes a change to the property. Simply initializing the properties in the Settings panel should not cause an update event.

If care is not taken to ensure this behavior, then unexpected behavior may occur. The page will still run, but to the detriment of the user experience. For example, these behaviors may occur:

- The **Save** button will become active simply by bringing up the Settings panel.
- The user must click **Undo** multiple times before any effect is visible.
- The `Redo` stack is removed because the component wrote back an unexpected change and updated the `Redo` stack with the new value.

The sample code provided in this tutorial for the Settings panel gives an example of how to ensure you are only writing back when you're ready to actually call `saveData` and not on initialization. Similar care should be taken within the component itself to not update `customSettingsData` unless it involved a user interaction, though typically this is less of a concern.

Continue to Step 16: Asset Management.

Step 16: Asset Management

This step describes and explains how to manage the assets used by a component.

Assets include components and custom components that Oracle Content Management must know about to manage the life cycle of the assets.

Oracle Content Management content Folder

Each site that you create in Oracle Content Management comes with its own `content` folder. This is a hidden folder that you won't normally see. When the site is published, all the files in the `content` folder are also published to the file system.

For example, when you select an image using the Image component, Oracle Content Management makes a copy of the image you selected and places it in the content folder. Your URL always points to this copied version of the image so that if you delete the original image, your site won't break. This also applies to the other components provided by Oracle Content Management: Gallery, Gallery Grid, Document, Social Bar, File Download, as well as background images for slots and componentGroups.

For a custom component to take part in this asset life cycle, the custom component must tell Oracle Content Management about any assets it wants the service to manage on its behalf. Because this involves making a copy of the asset, the custom component must also use Oracle Content Management APIs to select the asset so that we know how to manage it.

Manage URLs

The URL to an asset changes based on a number of criteria.

- The runtime URL to a component is different than the Site Builder URL to the component
- If you copy a page, Oracle Content Management also makes a copy of all the referenced assets in the content folder so you never have two components pointing to the same asset in the content folder
- Dropping a componentGroup onto the page makes new copies for any assets referenced by a component in the componentGroup

In addition, while a relative URL may be fine for a local component, remote components require the fully qualified URL to any asset that you want Oracle Content Management to manage on your behalf so they can render their inline frame content with the full URL.

Because you can't rely on the URL remaining static, you must only hold references to the ID to the asset in your code and retrieve the asset's URL when you want to render the asset.

Manage Assets

These Sites SDK APIs are available for managing assets.

```
SitesSDK.getProperty('componentAssets', callback);
```

- This gets the array of current assets
- Each asset entry consists of:

- **id**: Unique ID for the asset.
- **title**: Oracle Content Management title metadata.
- **description**: Oracle Content Management description metadata.
- **fileName**: Original name of the selected file. Useful for display in the Settings panel for your custom component so users know what file they selected. This is not the name of the file copied to the content folder.
- **source**: Macro enabled URL to the asset. This value will change over time and should not be referenced by your component, but it must be saved as part of the asset.
- **url**: Fully qualified URL to the asset based on the context in which `getPropert()` was called.

```
SitesSDK.setProperty('componentAssets', [assets]);
```

- Call this to save all the assets you want Oracle Content Management to manage on your behalf.
- If you don't call this, then no asset will be saved.
- Any assets not in this array will be deleted when the site is published.
- The `assets` parameter is an array of assets in the same format as you get returned by `getProperty` and is also returned by `filePicker`.

> **Note:**
>
> No `url` value is stored. That value is dynamically created when you ask for the assets.

```
SitesSDK.filePicker(options, callback);
```

- An API to bring up the file picker to select the list of assets.
- It calls the callback on successful selection of assets passing in the array of selected assets.
- Nothing is saved at this point and it's up to the component to call `setProperty('componentAssets', [assets]);` to save items from this selection combined with any other assets to be saved.

Example of Select Asset

This section shows you how to select an asset, store its ID, and re-fetch the actual values from the stored assets.

1. Edit the `settings.html` file.
2. Change the template object to include an `Image selection`.

```
<div>
    <!-- Image selection -->
    <label id="imageLabel" for="imageAsset" class="settings-heading" data-
bind="text: 'Image'"></label>
    <input id="imageAsset" data-bind="value: imageName" readonly
class="settings-text-box">
```

```
    <button id="imageSelect" type="button" class="selectbutton"
data-bind="click: showFilePicker">Select Image</button>
</div>
```

3. Change the viewModel to add an observable to store the ID of the selected asset.

```
self.imageID = ko.observable();
```

4. Change the viewModel to manage selection of the asset by bringing up the file picker and displaying the name of the selected asset.

```
//
// handle component assets
//
self.assets = []

// bring up a file picker to select the assets
self.showFilePicker = function () {
    // select an image
    SitesSDK.filePicker({
        'multiSelect': false,
        'supportedFileExtensions': ['jpg', 'png']
    }, function (result) {
        if (result.length === 1) {
            // update the array of assets
            self.assets = result;

            // update the image in customSettingsData
            self.imageID(result[0].id);
        }
    });
};

// update the display name based on the assets
self.imageName = ko.computed(function () {
    var imageName = '',
        imageID = self.imageID();
    for (var i = 0; i < self.assets.length; i++) {
        if (self.assets[i].id === imageID) {
            imageName = self.assets[i].fileName;
            break;
        }
    }

    return imageName
}, self);
```

5. Update the viewModel to first retrieve the assets before getting the `customSettingsData`. This code will also cause the `self.imageName` to be invoked when the `self.ImageID()` observable changes.

```
SitesSDK.getProperty('componentAssets', function (assets) {
    self.assets = assets;
    SitesSDK.getProperty('customSettingsData', function (data) {
```

```
            //update observable
            self.imageWidth(data.imageWidth);
            self.imageID(data.imageID);
            self.titleText(data.titleText);
            self.userText(data.userText);

            // note that viewModel is initialized and can start saving data
            self.initialized(true);
            self.saveData = true;
        });
    });
```

6. Finally, update the `save` function to save the `imageID` and make sure to update the `componentAssets` with your list of referenced assets.

```
self.save = ko.computed(function () {
    var saveconfig = {
        'imageWidth': isNaN(self.imageWidth()) ? self.imageWidth() :
self.imageWidth() + 'px',
        'imageID': self.imageID(),
        'titleText': self.titleText(),
        'userText': self.userText()
    };

    // store the selected asset and update custom settings
    if (self.saveData) {
        SitesSDK.setProperty('componentAssets', self.assets);
        SitesSDK.setProperty('customSettingsData', saveconfig);
    }
}, self);
```

Check the Results for Select Asset

1. Refresh your page in your site so Site Builder can pick up changes to the component.
2. Take the page into Edit mode.
3. Drag and drop your component on the page.
4. Bring up the Settings panel.
5. Click the **Select image** button.
6. Browse (or upload) and select an image.

   Note that the name of the image is stored showing the selected image.
7. Close out of the Settings panel.
8. Bring up the Settings panel again.

   Note that the name of the image is again reflected.

Example of Render Asset

This section shows you how to retrieve the assets and render them in your component, and also have the component dynamically update whenever values are changed in your settings panel.

> **Note:**
>
> Although this is showing an example for a local component that is in an inline frame on the page, similar code will work for components rendered inline in the page.

1. Edit the `render.html` file.
2. Update the template to include your asset:.

```
<!-- ko if: imageURL -->
<div style="flex-shrink:0;">
    <img data-bind="attr: {style: imageStyle, id: 'imageId', src:
imageURL, alt: '', title: ''}, click: imageClicked" />
</div>
<!-- /ko -->
```

3. In the viewModel, create two observables to get the `imageID` from the `customSetttingsData` and store the `imageURL` retrieved from the stored list of assets.

```
self.imageID = ko.observable();
self.imageURL = ko.observable();
```

4. Update the viewModel so that whenever the `imageID` changes, it gets the corresponding image asset URL.

```
self.imageID.subscribe(function (imageID) {
    // whenever the image changes get the updated referenced asset
    SitesSDK.getProperty('componentAssets', function (assets) {
        for (var i = 0; i < assets.length; i++) {
            if (assets[i].id === imageID) {
                self.imageURL(assets[i].url);
                break;
            }
        }
    });
});
```

5. Update the viewModel to retrieve the ID from the `customSettingsData`.

Check the Results for Render Asset

1. Refresh your page in your site so Site Builder can pick up changes to the component.
2. Take the page into Edit mode.
3. Drag and drop your component on the page.
4. Bring up the Settings panel.
5. Click the **Select image** button.
6. Browse (or upload) and select an image.

Note that the name of the image is stored showing the selected image.

7. Close out of the Settings panel.

   At this point you should see your selected image rendered in the component.

Continue to Tutorial Review.

Tutorial Review

This tutorial gives you an overview of how to create a customized component using a Knockout Component Factory.

The main purpose of this tutorial is that using this pattern, you can create any custom component by just updating the `SampleComponentViewModel` and `sampleComponentTemplate` JavaScript objects. The `sampleComponentFactory` and `SampleComponentImpl` objects haven't changed as you went through the tutorial. You were able to implement these changes without having to deal with communicating with the page, and were able to perform these tasks:

- Communicate changes from your Settings panel to your component, and have those changes persisted.
- Execute triggers and actions, and interact with other components on the page.
- Create layouts and leverage nested components.
- Define component-specific styles.

While this example split out the custom component into a number of files, this was for clarity of the tutorial. For optimization, you should consider appropriately packaging your files to avoid multiple downloads.

Finally, while this tutorial is suitable for Knockout based components, if you want to create custom components using another JavaScript technology stack such as AngularJS, you must re-implement the `SampleComponentImpl` object to create the corresponding communication with that framework along with a technology specific implementation of the actual component. This work is beyond the scope of this tutorial.

29
Troubleshoot

You may have some questions about creating sites. Here's some answers for you.

- I am trying to create a site, but there are no templates
- I can't delete a site
- I can't open the site tree or edit a page
- I added a component, but it doesn't show up on the page
- My folder, file, and conversation components don't work
- I uploaded a new version of an image, but it doesn't show up on the page
- I changed the page layout and some of my content disappeared
- I added a component based on another service but it isn't working
- My enterprise site shows a warning

I am trying to create a site, but there are no templates

Templates must be installed and shared before you can create a site.

The templates provided with Oracle Content Management must be installed and shared by an administrator before you can use them. Contact your administrator.

Similarly, when you create a template, whether by importing, copying, or creating from a site, the template can't be used by anyone else until you explicitly share it.

See Configuring Site Settings in *Administering Oracle Content Management*.

I can't delete a site

Here are some reasons why you may not be able to delete a site:

- The site is online.

  First, take the site offline and then delete it. You must have the manager role for a site to take the site offline.

- You don't have the required privileges.

  You can delete a site if you created the site (you're the site owner) or if someone has shared a site with you and has given you the contributor or manager role.

See Manage Sites and Site Settings for details.

I can't open the site tree or edit a page

Site Builder opens in preview mode by default, which allows you to view the site, but not edit it. You can only edit a site if the editor is in edit mode.

Make sure that  is set to **Edit**.

I added a component, but it doesn't show up on the page

There are several possible reasons for this.

- Verify that the URL associated with a remote component (app) is valid. See Register Remote Components.
- The editor encloses remote components in an HTML element called an inline frame (iframe tag). Not all remote components allow being enclosed in an inline frame.

  Check with the component provider to find out if it can be enclosed in an inline frame.

My folder, file, and conversation components don't work

Components that communicate with Oracle Content Management require certain resources and settings.

The following components require access to resources in Oracle Content Management:

- Folder list
- File list
- Documents manager
- Conversation

Folder and file components require access to the REST application programming interface (API).

Verify the following:

- The component must have access to the API endpoint to access folder and file REST operations.

  By default, the Oracle Content Management REST API endpoint is available if you use the standard URL provided for the site. If you use a custom URL, you may have to explicitly provide access to the /documents REST API endpoint.

  See How do I map a site URL?

I changed the page layout and some of my content disappeared

If you choose a layout with fewer or differently named slots, existing content in other slots won't display in the new layout.

The content isn't deleted, it just can't be displayed unless the layout you choose has a slot with the same name.

Make sure you choose a layout with the same number of slots. If the layout you choose has the same number of slots, but some of your content still does not display,

it's likely the slots have different names. Contact the theme designer to resolve the discrepancy.

I uploaded a new version of an image, but it doesn't show up on the page

When you select an image to use with a component, a unique copy is created and stored with the site.

When you upload files, they are stored in the site repository. If you upload a file with the same name as an existing file, it creates a new version of the file.

The image file you see on the page is a copy of the file selected from the repository and is not automatically updated. This is by design and prevents inadvertent changes in existing pages when an image file is updated in the repository.

To update the image used with the component, you must explicitly select the image from the component property page. See Images for details.

I added a component based on another service but it isn't working

You can integrate your site with several different processes and services.

If you created a component based on integration with Process Cloud Service, Oracle Intelligent Advisor (formerly Oracle Policy Automation), and other services, both services must use the same identity domain. Check with the person who set up the integration to verify that the two services use the same identity domain.

My enterprise site shows a warning

If an enterprise site includes v1.0 content layouts, you'll see a warning in Site Builder and when viewing the site because the layouts won't be able to render. They need to be updated to v1.1 content layouts.

This situation can happen in one of two ways:

- If you make an existing site translatable, the site will automatically be converted to an enterprise site by being assigned a default language and localization policy. If the site template includes v1.0 content layouts, the content layouts won't be able to render.
- If you create an enterprise site from an older template that uses v1.0 content layouts, again, the content layouts won't be able to render.

To correct this issue, update the content layouts to v1.1 content layouts. You can create a new content layout in the component catalog to see the difference. You'll need to add a line to register the version of the content layout and change `data.fieldname` to `fields.fieldname`.