

Oracle® Cloud

Developing with Oracle Content Management As a Headless CMS



F24098-35
June 2023



Oracle Cloud Developing with Oracle Content Management As a Headless CMS,

F24098-35

Copyright © 2019, 2023, Oracle and/or its affiliates.

Primary Authors: Clare Yan, Hareesh S Kadlabalu, Keith MacDonald, Bruce Silver, Ron van de Crommert, Mark Paterson, Preston So, Ankur Saxena, Sarah Maslin

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

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

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

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

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

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

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

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

Contents

Preface

Audience	vii
Documentation Accessibility	vii
Related Resources	vii
Conventions	vii

1 Get Started

Overview of Oracle Content Management	1-1
Access Oracle Content Management	1-2
Understand Roles	1-2
Manage Assets	1-3
Collaborate on Documents	1-3
Build Sites	1-4
Integrate and Extend Oracle Content Management	1-4
Oracle Content Management As a Headless CMS	1-4
Content Management	1-5
Content Delivery	1-6
Key Concepts	1-6
Content Model	1-7
Assets	1-8
Asset Management	1-8
Asset Properties	1-9
Asset Types	1-10
Create Asset Types	1-10
Content API for Assets	1-12
Digital Asset Types	1-16
Seeded Digital Asset Types	1-16
Custom Digital Asset Types	1-17
Creating Custom Digital Asset Types	1-18
Creating Digital Assets	1-20
Content API for Digital Assets	1-21
Content Item Types	1-24

Create Content Item Types	1-25
Create Content Items	1-29
Content API for Content Items	1-31
Asset Repositories	1-34
Videos	1-36
Multilingual Content and Translations	1-39
Language Attributes of Assets	1-41
Discover Available Translations	1-41
Fetch a Specific Translation of an Asset	1-42
Translation Jobs	1-42
Content Versions	1-44
Publishing and Channels	1-45
Publishing	1-45
Channels	1-46
Policies	1-46
Publishing Process	1-47
Taxonomies	1-47
Taxonomies from a Management Perspective	1-48
Taxonomy Life Cycle	1-50
Taxonomies from a Delivery Perspective	1-51
Discovering the Structure of a Taxonomy	1-51
Discovering Asset Categorization	1-51
Friendly URLs for Assets	1-53
Quick Start	1-54
Register for Oracle Cloud	1-55
Select an Account Name and Home Region	1-55
Provide Payment Information	1-56
Provision an Instance of Oracle Content Management	1-57
Choose a Storage Compartment	1-58
Create Your Oracle Content Management Instance	1-59
Add a Content Model, Some Content, and a Channel	1-61
Create a Content Type	1-61
Create a Publishing Channel	1-63
Create an Asset Repository	1-64
Create a Content Item (Asset)	1-65
Publish Content Assets to a Channel	1-66
Configure Oracle Content Management As a Headless CMS	1-67
Configure Cross-Origin Resource Sharing (CORS)	1-68
Acquire and Refresh API Access Tokens	1-68
Issue Your First Request to Oracle Content Management	1-69
Retrieve Content Through Postman	1-70

Retrieve Content Through cURL	1-72
Retrieve Content Through an XMLHttpRequest	1-73
Next Steps	1-73

2 Oracle Content Management REST APIs for Headless Development

REST API for Content Delivery	2-1
REST API for Content Management	2-2

3 Oracle Content Management SDKs

Content SDK for JavaScript	3-1
Mobile SDKs	3-1
Content SDK for Java	3-1
Content SDK for Swift	3-2
Sites SDK	3-2
Translation Connector SDK	3-3

4 Starter Site CLI for React Development

Install the Starter Site CLI	4-1
Run CLI Commands	4-2
Get Content from Oracle Content Management	4-3
Set Up the Oracle Content Management Server Connection	4-3
Create a Site	4-3
Build a Site	4-5
Run a Site in Development Mode	4-5
Run a Site with Oracle Content Management Server Content	4-5
Build a Site for Production	4-6
Run a Site in Production Mode	4-6
Structure of the React JS Site Template	4-6
Generated Components	4-9
Starter Site Runtime	4-10

5 Connecting to Headless Experiences

Create an Experience Object	5-1
Configure Experience Object Properties	5-2
Add Outgoing Targets to Experience Objects	5-2
Add a TARGET_IDENTIFIER Token	5-3
Enable and Disable Incoming Webhook	5-3
Analyze and Extract Payload Information	5-4

View Event Information for an Experience	5-6
View Connected Headless Experiences	5-6
Launch Properties After a Successful Experience Creation	5-7
Set Security Admin Settings Through APIs	5-7
Sharing an Experience Object	5-7

6 Instrumenting Headless Sites with Consumption Analytics

Analytics Script	6-1
Asset Events	6-2
Page Instrumentation	6-2
Component Attributes	6-3
JavaScript API Calls	6-4
Configuration Options	6-5
Add data-asset-operation Markup for Digital Assets	6-6
Add data-asset-operation Markup for Referenced Field Types	6-7
Use a Site-Specific Oracle Infinity Account	6-7
Use Your Own Oracle Infinity Tag	6-7

7 Samples

Preface

Developing with Oracle Content Management As a Headless CMS describes how to use Oracle Content Management as a headless content management system (CMS) to develop advanced websites and web applications using modern technologies.

Audience

Developing with Oracle Content Management As a Headless CMS is intended for developers who want to use a headless content management system (CMS) and modern web technologies to develop advanced websites and web applications.

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.

Related Resources

For more information, see these Oracle resources:

- *Getting Started with Oracle Cloud*
- *Administering Oracle Content Management*
- *Building Sites with Oracle Content Management*
- *Collaborating on Documents with Oracle Content Management*
- *Integrating and Extending Oracle Content Management*

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.

Convention	Meaning
<i>italic</i>	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.

1

Get Started

The Oracle Help Center has lots of resources to help you get started with Oracle Content Management.

Resources in the [Oracle Help Center](#) include [documentation](#), [videos](#), [guided tours](#), and developer information. And if you need it, there's [support](#) and a [community](#) to help.

The following sections also provide information to help you get started:

- [Overview of Oracle Content Management](#)
- [Oracle Content Management As a Headless CMS](#)
- [Key Concepts](#)
 - [Asset Repositories](#)
 - [Content Item Types](#)
 - [Assets](#)
 - [Digital Asset Types](#)
 - [Multilingual Content and Translations](#)
 - [Content Versions](#)
 - [Publishing and Channels](#)
 - [Taxonomies](#)
 - [Friendly URLs for Assets](#)
- [Quick Start](#)

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.

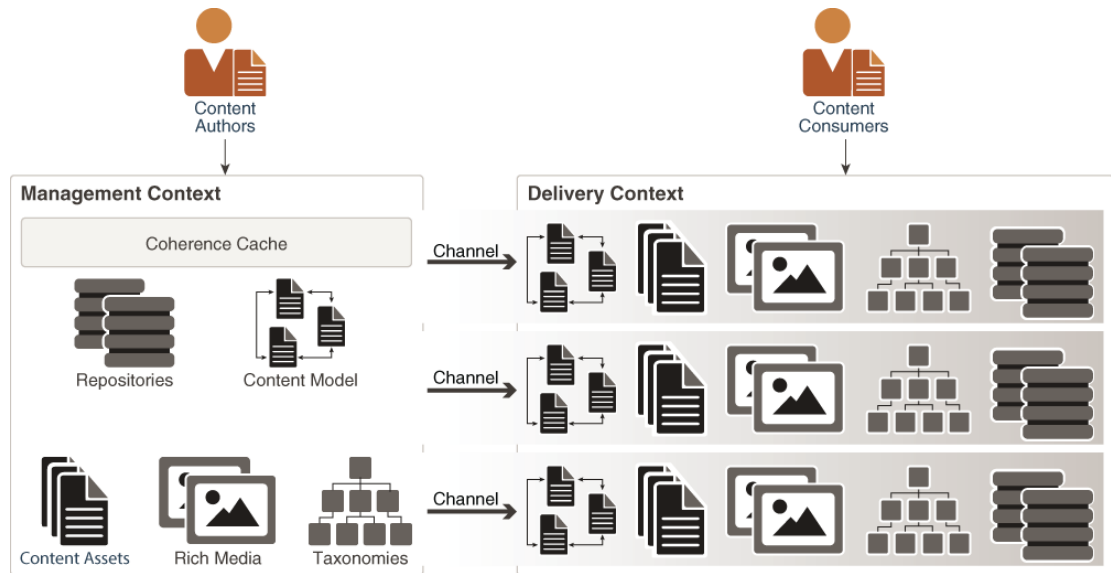
[Learn more...](#)

Oracle Content Management As a Headless CMS

Oracle Content Management can be used as a powerful and flexible back-end content management system (CMS) in the cloud. It's built from the ground up as a central

content hub that makes content accessible through REST APIs for publication in any context or display on any device.

There are two fundamental perspectives to look at Oracle Content Management: **content management** and **content delivery**. The following diagram shows a logical view of the overall architecture.



Content Management

Oracle Content Management offers many tools for effective content management, which involves content definition, creation, collaboration, approval, and administration. All this can be done from a variety of user interfaces (web browser, mobile apps, Microsoft Office, desktop app), and this is how content authors mostly experience the product.

When you sign in to Oracle Content Management as a content author, you'll typically manage your content and collaborate with other people from any of the clients. You can add assets, manage assets, share assets with other people, or have context-specific conversations on individual assets. You can also get insight into what content is being authored, how it's being published, and what workflows it's in through content analytics dashboards.

Oracle Content Management also provides management interfaces that allow content and system administrators to perform system administrative and monitoring tasks.

Different roles can be assigned to users to control what they can do in the content management environment, and workflows may be in place to guide the content creation and management processes.

Oracle Content Management can handle all kinds of content, including digital assets, structured content, rich media, and content assets. Assets can be stored in repositories, where they can be categorized using taxonomies and accessed for further processing. The structure and interdependencies of all content are captured in the content model, which basically defines the content management environment.

Even though Oracle Content Management offers front-end user interfaces to manage content, all management operations can also be accomplished programmatically, through a set of management REST APIs. The content management user interfaces use these

management APIs. These APIs are available to you as well for performing integrations, data massaging, or any other data manipulation needs.

Content Delivery

Content delivery is another important aspect of Oracle Content Management. This is all about getting content to end users, such as website visitors or app users. These are the content consumers.

Once content is authored and has gone through an approval process, it can be published, which makes it available for websites and apps to use. Published content is made available to clients in read-only format through a set of RESTful application program interfaces (APIs).

Publishing content involves a certain set of policies (checks and balances) and a logical notion of a destination or channel. Content can be published to many channels at the same time. It's also possible to withdraw content from a channel by unpublishing it. The acts of publishing and unpublishing alter the visibility of content in a particular context.

Content delivery as such doesn't have a user interface in Oracle Content Management. However, published content is visible in the management interfaces. It's useful to think of published content as read-only copies of assets in the management perspective.

All content delivery can be done programmatically, through a set of delivery REST APIs. This allows you to develop websites and applications using Oracle Content Management as a "headless" back-end content management system (CMS). Several tutorials are available to get you started with various technologies.

Key Concepts

Some key concepts can help you understand how to develop with Oracle Content Management as a headless CMS:

- [Content Model](#)
- [Assets](#)
- [Digital Asset Types](#)
- [Content Item Types](#)
- [Asset Repositories](#)
- [Videos](#)
- [Multilingual Content and Translations](#)
- [Content Versions](#)
- [Publishing and Channels](#)
- [Taxonomies](#)
- [Friendly URLs for Assets](#)
- [GraphQL](#)
- [Cross-Origin Resource Sharing \(CORS\)](#)

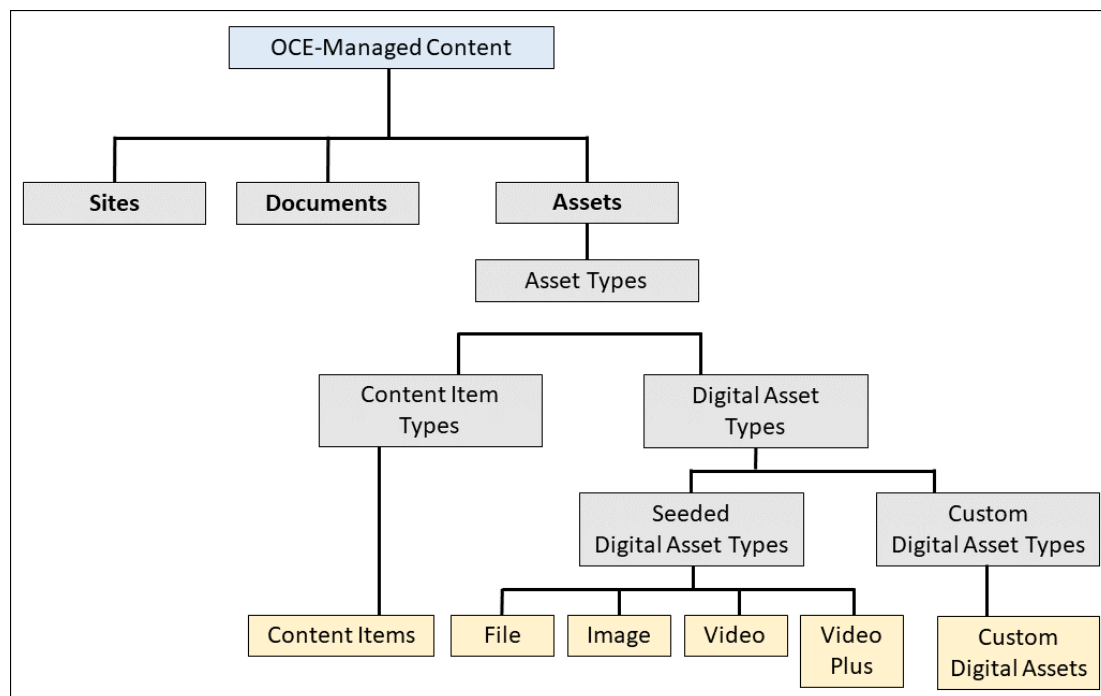
Content Model

Oracle Content Management is your universal content hub in the cloud, providing an API-first content management and delivery platform.

To learn more about Oracle Content Management, visit <https://www.oracle.com/content-experience>.

Oracle Content Management manages three types of content, as shown in the diagram below:

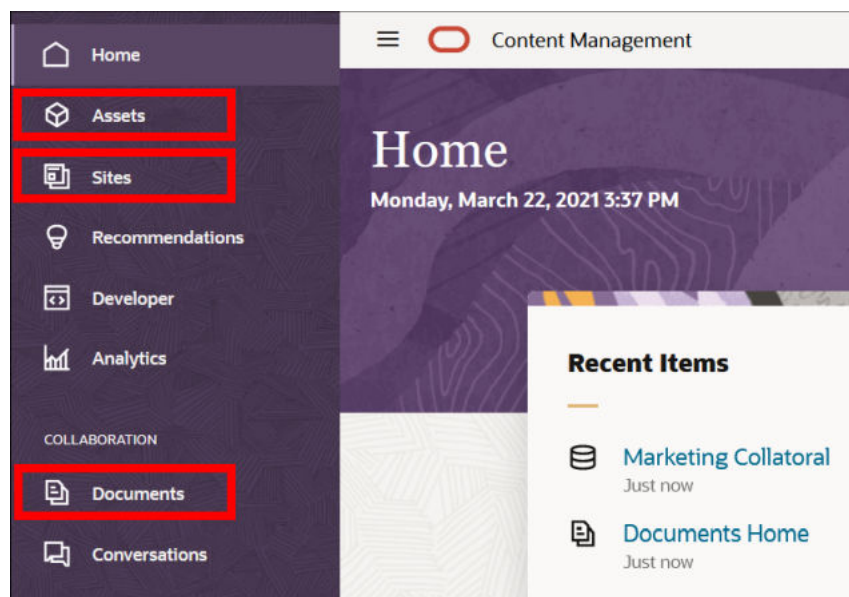
- **Sites**—You can build sites in an easy-to-use, feature-rich WYSIWYG environment using content and assets managed in Oracle Content Management. See *Building Sites with Oracle Content Management* for further information.
- **Documents**—This is content that's mostly used for collaboration, syncing, or backup purposes. It will often be business content such as documents, spreadsheets, or presentations, but it can be any file type, including images and videos. See *Collaborating on Documents with Oracle Content Management* for further information.
- **Assets**—This is content that's typically used for content modeling, publishing, and delivery. It will often be media files (images or videos), but they can also be documents or structured content that includes text. In headless development, you'll mostly be working with assets, so that's what we'll be focusing on here. See [Assets](#) for further details, and also *Managing Assets with Oracle Content Management*.



 **Note:**

"Regular" (that is, non-asset) documents can easily be turned into assets if their business use changes. New, separate copies of these documents will then be created for use as assets.

Content authors and contributors typically manage and access sites, documents, and assets in the Oracle Content Management web interface, desktop app, or mobile apps. As a headless developer, you'll primarily use the [REST APIs](#)—mostly the [REST API for Content Delivery](#) and [REST API for Content Management](#)—to access content programmatically for inclusion in applications or delivery to various [publishing channels](#).



Assets

Assets represent the smallest units of managed content in Oracle Content Management. They're typically used for content modeling, publishing, and delivery.

As such, they differ from "regular" (that is, non-asset) documents, which are mostly used for collaboration, syncing, or backup purposes. Assets will often be media files (images or videos), but they can also be documents or structured content that includes text. In headless development, you'll mostly be working with assets.

Asset Management

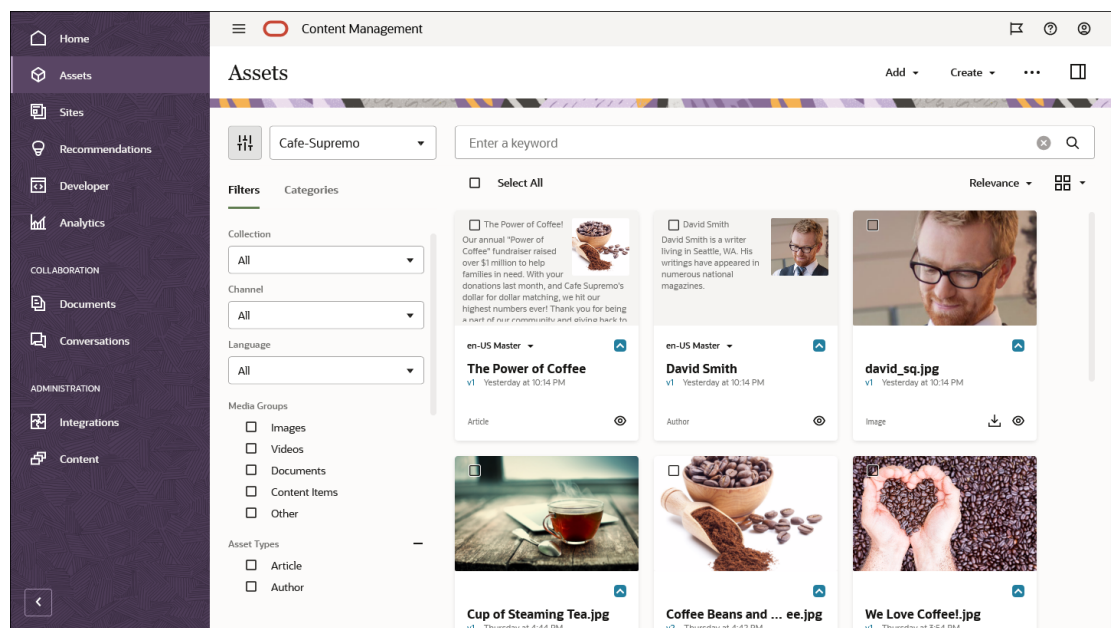
Content contributors manage assets in [repositories](#) in the Oracle Content Management web interface.

Depending on their assigned privileges, content contributors can perform a number of asset management tasks, including:

- Manage asset types

- Manage localization policies
- Manage publishing channels
- Manage taxonomies
- Manage repositories
- Manage workflows
- Manage audience attributes
- Use digital assets
- Use structured content
- Use recommendations
- Use collections

See *Managing Assets with Oracle Content Management* for more information.



Asset Properties

All assets share a number of common properties, while different types of assets also have their own additional sets of properties.

Each of these properties can be programmatically accessed using the [REST APIs](#) for content modeling and delivery.

These are some of the basic properties that every asset of any type has:

- id
- type
- typeCategory
- name

- description
- slug
- language
- translatable
- createdAt (*value, timezone*)
- updatedAt (*value, timezone*)

In addition to these standard properties, assets of different types also have other sets of properties. For example, some image and video properties include:

- mediaType
- mimeType
- width
- Height
- size
- extension
- formats
- duration (videos only)

Asset Types

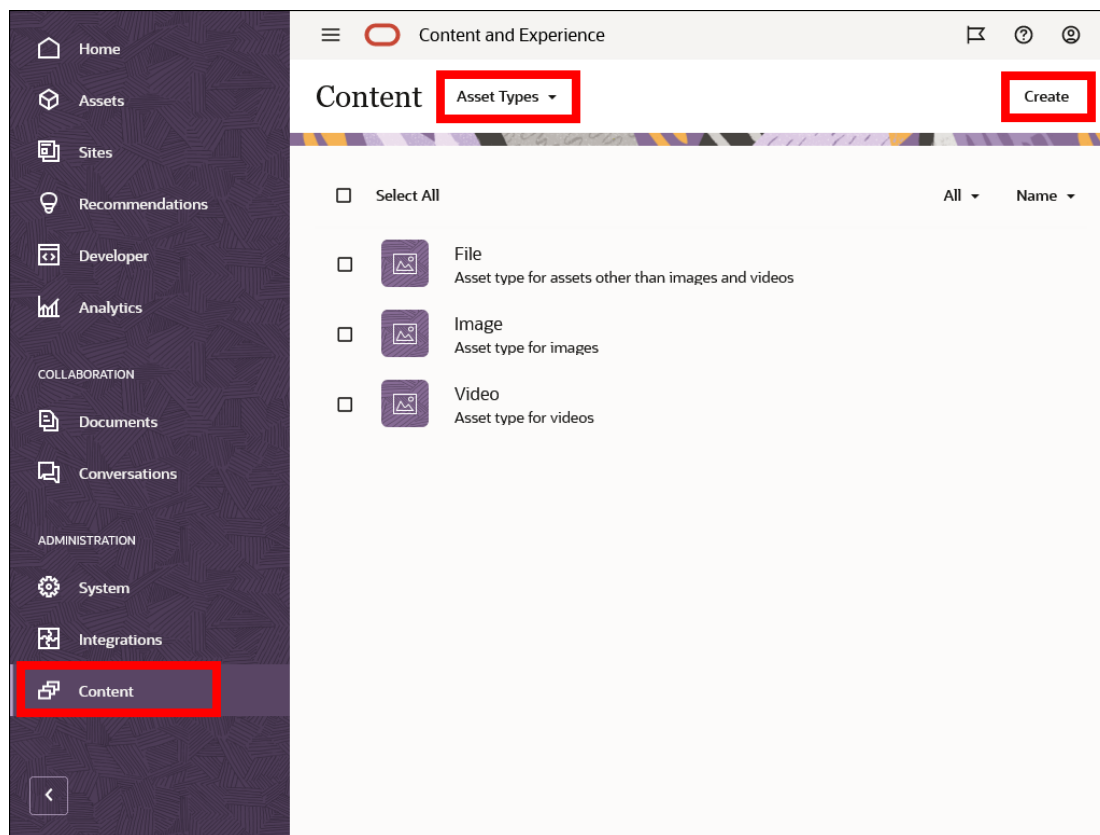
Every asset in Oracle Content Management is categorized as a specific asset type. These types determine how an asset can be managed and published.

There are two main types of assets:

- **Digital asset types**—These define file media types (MIME types) and sets of attributes for assets of that type. There are a number of **seeded (out-of-the-box) digital asset types**, but you can also define your own **custom digital asset types**. For example, you could have a "Logo" digital asset type, which consists of a PNG or JPG file along with some associated attributes such as copyright statement and caption. When you create an asset from a digital asset type, it's called a *digital asset* (so file plus associated attributes).
- **Content item types**—These are defined groups of data fields of various data types. For example, you might create a blog article content type, where each asset stores values for title, body, author, photo, date created, and a list of references to related articles. When you create an asset from a content item type, it's called a *content item*.

Create Asset Types

If you have the required administrative privileges, you can create asset types in the Oracle Content Management web interface (under **Administration > Content**).



When creating a new asset type, you choose to create either a [content item type](#) or a [digital asset type](#).

Create Assets

Content contributors can create new digital assets by uploading new files from their local computer or adding existing items in their Documents section of Oracle Content Management.

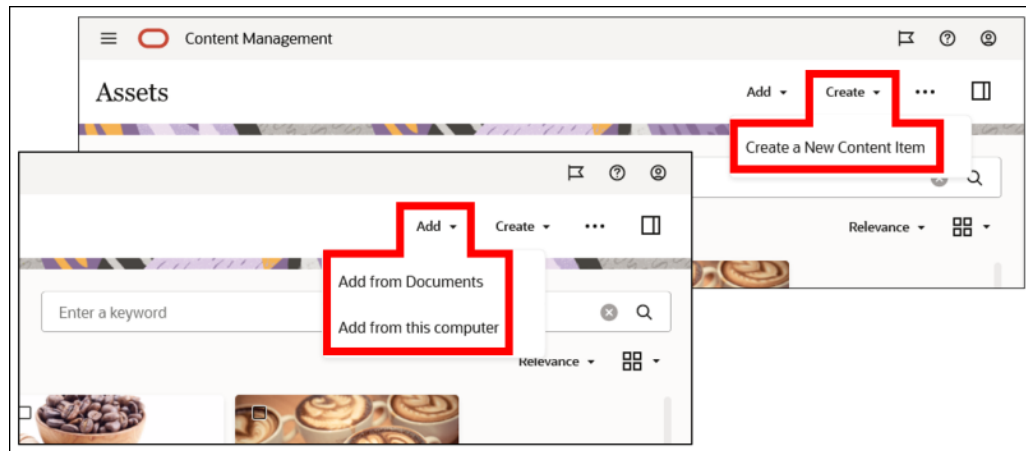
While adding assets, content contributors can assign digital asset types, categories, [channels](#), tags, and collections to support content modeling, routing, and delivery.

Similarly, they can also create new content items based on the content item types made available to them.



Note:

Assets must be published before they're available for use in publishing channels (and also through REST APIs).

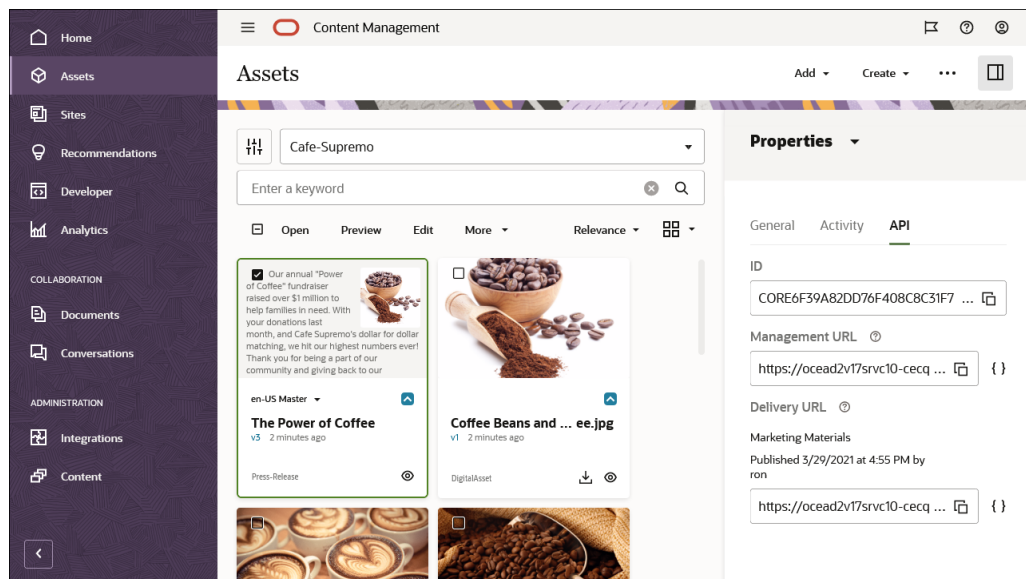


Content API for Assets

Assets exist both in a management context and in a delivery context. For an authored asset to be used in an application, it must first be published, so it becomes available in the delivery context.

We'll mostly focus on the delivery context here as that is of primary interest to developers. See *Managing Assets with Oracle Content Management* for more information on assets from a management perspective.

Once published, each asset is available as a REST resource. You can see the address to the resource as part of the asset properties, which you can find in the Oracle Content Management web interface.



Let's explore what such a REST resource looks like in the delivery context. First, the delivery URL has a specific form.

```
https://.../content/published/api/v1.1/items/
CORE6F39A82DD76F408C8C31F7D8FCB3E8C0?
channelToken=1c92cd5b68b245ax87ffb8898ff2fdbd
```

- The access context identifier `/published` indicates that this is about delivering published content.
- The object identifier `/items` indicates that we're working with asset items (and not other object types).
- This is followed by the identifier of the asset, which is a series of letters and numbers (for example, `CORE6F39A82DD76F408C8C31F7D8FCB3E8C0`). Each asset has its own unique identifier.
- At the end is a query parameter, `channelToken`, which is an identifier for the channel that this asset has been published to. It's the context in which the asset is being accessed.

You can see the full JSON response data for an asset by clicking the `{}` brackets next to the delivery URL. It looks something like this:

```
{
  "id": "CORE6F39A82DD76F408C8C31F7D8FCB3E8C0",
  "type": "Press-Release",
  "typeCategory": "ContentType",
  "name": "The Power of Coffee",
  "description": "Press release 'The Power of Coffee' for marketing kit",
  "slug": "1481786546522-the-power-of-coffee",
  "language": "en-US",
  "translatable": true,
  "createdDate": {
    "value": "2021-03-29T18:33:13.732Z",
    "timezone": "UTC"
  },
  "updatedDate": {
    "value": "2021-03-29T23:55:37.057Z",
    "timezone": "UTC"
  },
  "fields": {
    "body_text": "<!--mde-->\n\r\nOur annual 'Power of Coffee' fundraiser raised over $1 million to help families in need. With your donations last month, and Cafe Supremo's dollar for dollar matching, we hit our highest numbers ever! Thank you for being a part of our community and giving back to our community. Our annual 'Power of Coffee' fundraiser raised over $1 million to help families in need. With your donations last month, and Cafe Supremo's dollar for dollar matching, we hit our highest numbers ever! Thank you for being a part of our community and giving back to our community.",
    "photo": {
      "id": "CONTA37AC23CE5284C46ACF4D3C3B10A9950",
      "type": "DigitalAsset",
      "typeCategory": "DigitalAssetType",
    }
  }
}
```

Or, in text form:

```
{
  "id": "CORE6F39A82DD76F408C8C31F7D8FCB3E8C0",
  "type": "Press-Release",
  "typeCategory": "ContentType",
  "name": "The Power of Coffee",
  "description": "Press release 'The Power of Coffee' for marketing kit",
  "slug": "1481786546522-the-power-of-coffee",
  "language": "en-US",
  "translatable": true,
  "createdDate": {
    "value": "2021-03-29T18:33:13.732Z",
```

```
    "timezone": "UTC"
  },
  "updatedAt": {
    "value": "2021-03-29T23:55:37.057Z",
    "timezone": "UTC"
  },
  "fields": {
    "body_text": "\n\rOur annual \"Power of Coffee\" fundraiser raised
over $1 million to help families in need. With your donations last
month, and Cafe Supremo's dollar for dollar matching, we hit our
highest numbers ever! Thank you for being a part of our community and
giving back to our community. Our annual \"Power of Coffee\"
fundraiser raised over $1 million to help families in need. With your
donations last month, and Cafe Supremo's dollar for dollar matching,
we hit our highest numbers ever! Thank you for being a part of our
community and giving back to our community. ",
    "photo": {
      "id": "CONTA37AC23CE5284C46ACF4D3C3B10A9950",
      "type": "DigitalAsset",
      "typeCategory": "DigitalAssetType",
      "name": "Coffee Beans and Ground Coffee.jpg",
      "links": [
        {
          "href": "https://.../content/published/api/v1.1/items/
CONTA37AC23CE5284C46ACF4D3C3B10A9950?
channelToken=1c92bb5b68b245da87ffb8672ff2fddb",
          "rel": "self",
          "method": "GET",
          "mediaType": "application/json"
        }
      ]
    },
    "title": "The Power of Coffee!"
  },
  "links": [
    {
      "href": "https://.../content/published/api/v1.1/items/
CORE6F39A82DD76F408C8C31F7D8FCB3E8C0?
channelToken=1c92bb5b68b245da87ffb8672ff2fddb",
      "rel": "self",
      "method": "GET",
      "mediaType": "application/json"
    },
    {
      "href": "https://.../content/published/api/v1.1/items/
CORE6F39A82DD76F408C8C31F7D8FCB3E8C0?
channelToken=1c92bb5b68b245da87ffb8672ff2fddb",
      "rel": "canonical",
      "method": "GET",
      "mediaType": "application/json"
    },
    {
      "href": "https://.../content/published/api/v1.1/metadata-catalog/
items/CORE6F39A82DD76F408C8C31F7D8FCB3E8C0?
channelToken=1c92bb5b68b245da87ffb8672ff2fddb",
```

```

    "rel": "describedby",
    "method": "GET",
    "mediaType": "application/schema+json"
  }
]
}

```

The JSON data matches the various metadata fields for the asset in Oracle Content Management:

The screenshot shows the Oracle Content Management interface for an asset titled "The Power of Coffee". The interface includes fields for Name, Description (Optional), Content Item Data Fields (Title, Body Text, Photo), and a Photo gallery. Red arrows point from these fields to a JSON response on the right. The JSON response contains standard fields (id, type, name, description, slug, language, translatable, createdDate, updatedDate) and a "fields" node with user-defined fields (bodyText, photo). The photo field is a reference field containing href, id, type, name, and links.

```

{
  "id": "CORE6F39A82DD76F408C8C31F7D8FCB3E8C0",
  "type": "Press-Release",
  "typeCategory": "ContentType",
  "name": "The Power of Coffee",
  "description": "Press release 'The Power of Coffee' for marketing kit",
  "slug": "1481786546522-the-power-of-coffee",
  "language": "en-US",
  "translatable": true,
  "createdDate": {
    "value": "2021-03-29T18:33:13.732Z",
    "timezone": "UTC"
  },
  "updatedDate": {
    "value": "2021-03-29T23:55:37.057Z",
    "timezone": "UTC"
  },
  "fields": {
    "bodyText": "<!--mde-->\n\nOur annual 'Power of Coffee' fundraiser raised over $1 million to help families in need. With your donations last month, and Cafe Supremo's dollar for dollar matching, we hit our highest numbers ever! Thank you for being a part of our community and giving back to our community. Our annual 'Power of Coffee' fundraiser raised over $1 million to help families in need. With your donations last month, and Cafe Supremo's dollar for dollar matching, we hit our highest numbers ever! Thank you for being a part of our community and giving back to our community.",
    "photo": {
      "id": "CONTA37AC23CE5284C46ACF4D3C3B10A9950",
      "type": "DigitalAsset",
      "typeCategory": "DigitalAssetType",
      "name": "Coffee Beans and Ground Coffee.jpg",
      "links": [
        {
          "href": "https://_jcontent/published/api/v1.1/items/CONTA37AC23CE5284C46ACF4D3C3B10A9950/channelToken=1c92bb5b68b245da87fb8672f2fddb",
          "rel": "self",
          "method": "GET",
          "mediaType": "application/json"
        }
      ],
      "title": "The Power of Coffee!"
    },
    "links": [
      {
        "href": "https://_jcontent/published/api/v1.1/items/CORE6F39A82DD76F408C8C31F7D8FCB3E8C0/channelToken=1c92bb5b68b245da87fb8672f2fddb",
        "rel": "self",
        "method": "GET",
        "mediaType": "application/json"
      },
      {
        "href": "https://_jcontent/published/api/v1.1/items/CORE6F39A82DD76F408C8C31F7D8FCB3E8C0/channelToken=1c92bb5b68b245da87fb8672f2fddb",
        "rel": "canonical",
        "method": "GET",
        "mediaType": "application/json"
      },
      {
        "href": "https://_jcontent/published/api/v1.1/metadata-catalog/items/CORE6F39A82DD76F408C8C31F7D8FCB3E8C0/channelToken=1c92bb5b68b245da87fb8672f2fddb",
        "rel": "describedby",
        "method": "GET",
        "mediaType": "application/schema+json"
      }
    ]
  }
}

```

Fields in green are all standard fields; they are present in every asset. One such standard field is the unique identifier (id) of the asset. Other standard fields are asset properties such as type, name, description, slug, language, translatability status (translatable), and the creation and update dates (*createdDate* and *updatedDate*).

A child "fields" node contains all user-defined field values, such as title, body, photo, or whatever else was specified in the asset type definition.

Looking at the "photo" field in the example above, you may notice that this looks a bit different. That's because it's a reference field. These don't contain the complete data set for the asset, but merely an identifier, a name, a type, and a link to the asset. In fact, any media field (the data type for the "photo" field) is actually a reference field behind the scenes. It's a reference to an out-of-the-box type called DigitalAsset (which is why "type" under the photo field says "DigitalAsset").

Now, suppose you have an "Article" content item with an "author" reference field, and you want to access all of the author field along with the article without having to incur additional network calls. You can easily do that by simply adding `expand=fields.author` as a query parameter, and the author field expands inside the Article JSON response.

Digital Asset Types

Digital assets in Oracle Content Management are often media files such as images or videos for use in websites or other publishing channels, but they can be any file type, including documents.

There are two different digital assets types in Oracle Content Management:

- Seeded digital asset types
- Custom digital asset types

Seeded Digital Asset Types

Oracle Content Management natively classifies digital assets into one of four out-of-the-box (seeded) classes, which are essentially predefined asset types in their own right.

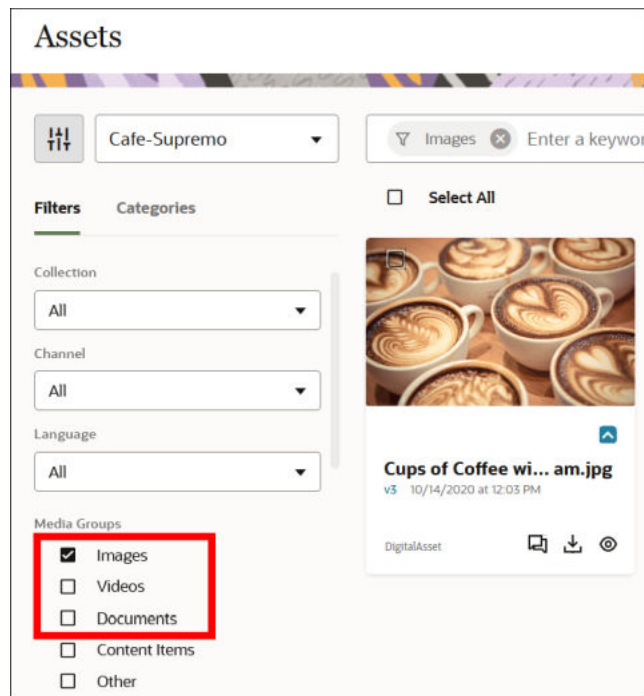
- Image
- Video
- Video Plus (if enabled in Oracle Content Management)
- File (essentially anything that's not an image or video)

 **Note:**

If you go to the Asset Types page in the Oracle Content Management web interface, you'll see these seeded asset types in your list, and you can't delete them.

This classification—along with repositories, categories, tags, collections, and so on—helps content modelers design their model to represent their business needs accurately. The out-of-the-box digital asset types are automatically assigned to any assets that are uploaded to Oracle Content Management, unless a custom digital asset type is selected instead.

Users can easily filter their repositories in the Oracle Content Management web interface to show only assets of a specific type.



The "type" field in a JSON response for an asset shows its type; for example:

```
"id": "CONT8760313315D948B68E76A7A07F840DCC",  
"type": "Image",  
"typeCategory": "DigitalAssetType",  
...
```

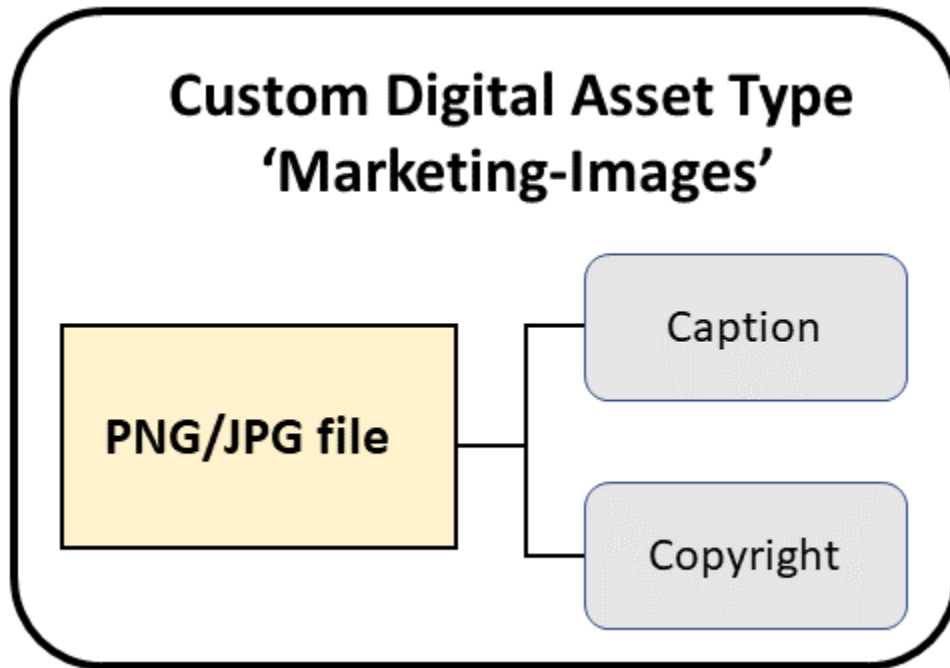
Custom Digital Asset Types

Sometimes the seeded (out-of-the-box) digital asset types don't provide enough control over content modeling. In addition to the out-of-the-box digital asset types, you can also create your own custom digital asset types. These provide a powerful mechanism for managing digital asset content models.

For example, you may want to create a set of enterprise-approved marketing images that content authors can choose from to include in their marketing content. These marketing images are just like any other image in the system, but they're a class of their own.

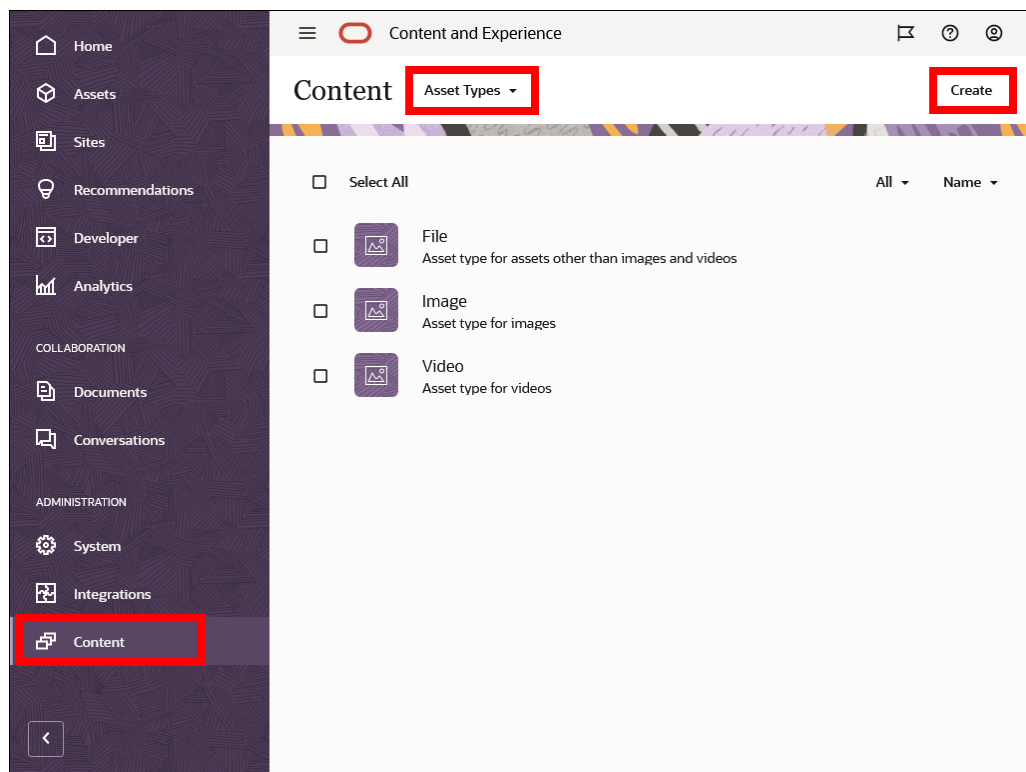
To support use cases like this, Oracle Content Management allows you to create custom digital asset types, which are essentially just like any other content type, but they're built around a file. Custom digital asset types can limit the file extensions that are allowed, and can also have custom fields (attributes), just like any other type (with some limitations).

Let's go back to the marketing images as an example. You could define a custom digital asset type called "Marketing-Images", which limits the file type to PNG and JPG, and requires a caption and copyright statement to be provided. This enables you to provide a more controlled experience for content authors as well as type safety in the content model. This is what the custom digital asset type would look like:



Creating Custom Digital Asset Types

If you have the required administrative privileges, you can create custom digital asset types in the Oracle Content Management web interface (under **Administration > Content**).

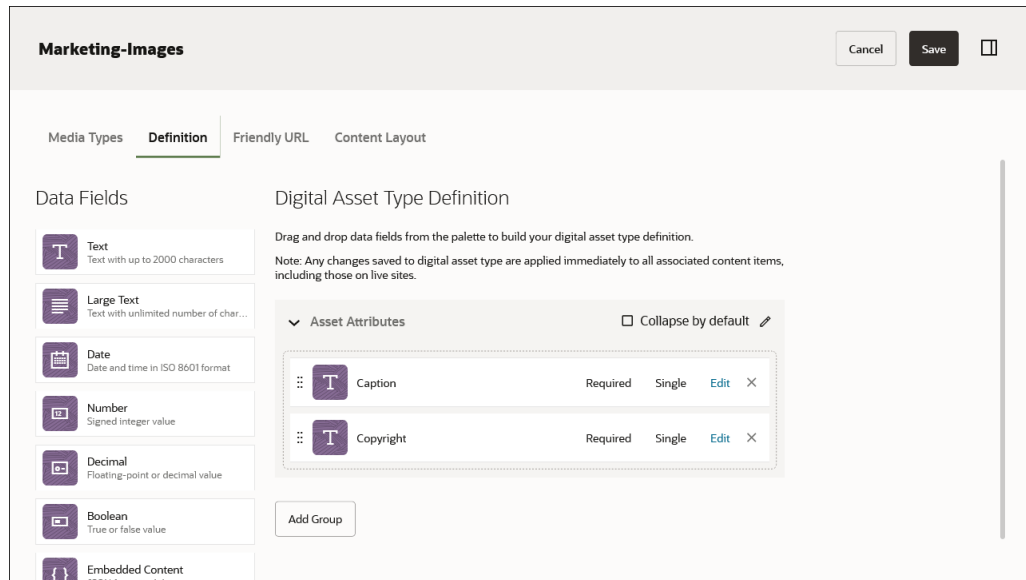


When creating a new asset type, you choose to create either a content item type or a digital asset type.

You can select media types that are allowed for a digital asset type. They can be any combination of media types.

File Extension	Media Type
jpg	image/jpeg
png	image/png

In addition to media types, you can also add custom fields, which can be any data type except Media and Reference fields. Content authors can capture additional business context in these custom fields and use them in content discovery. Developers can use these fields in their applications, just like any other type.



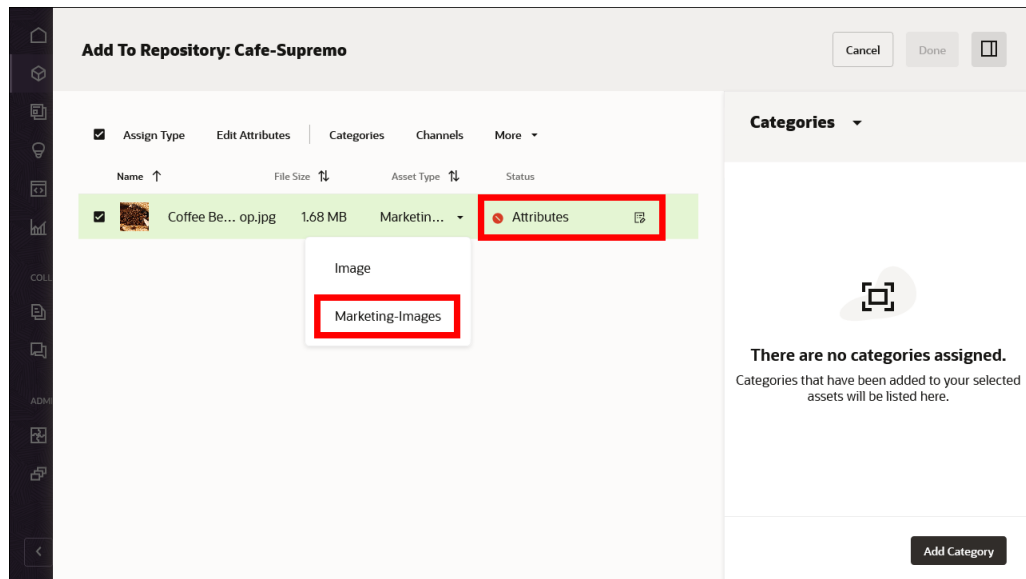
Other content types can reference custom digital asset types for media fields under Media Settings. Oracle Content Management will limit authors' choices to what's defined in the settings.

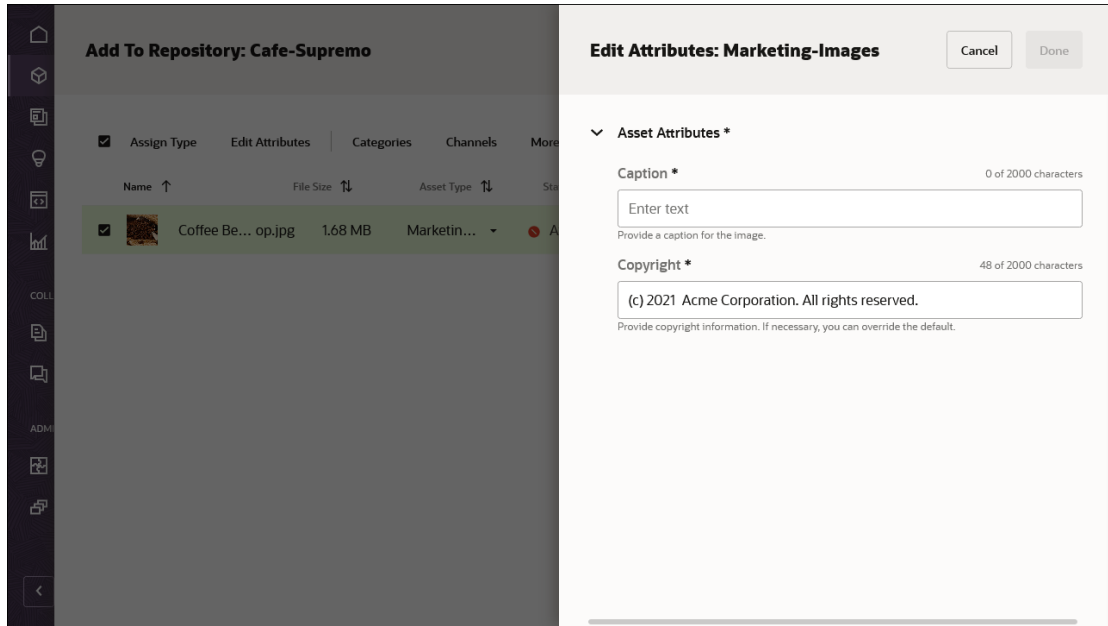
Creating Digital Assets

After a digital asset type has been defined and made available to a repository, content authors can add assets to that repository and designate them to be of that type.

They'll have to meet the file type requirements set for the type, and content authors will need to provide the attributes required (say, a caption or copyright statement).

Once the custom digital asset is in Oracle Content Management, it can be managed and accessed like any other asset.

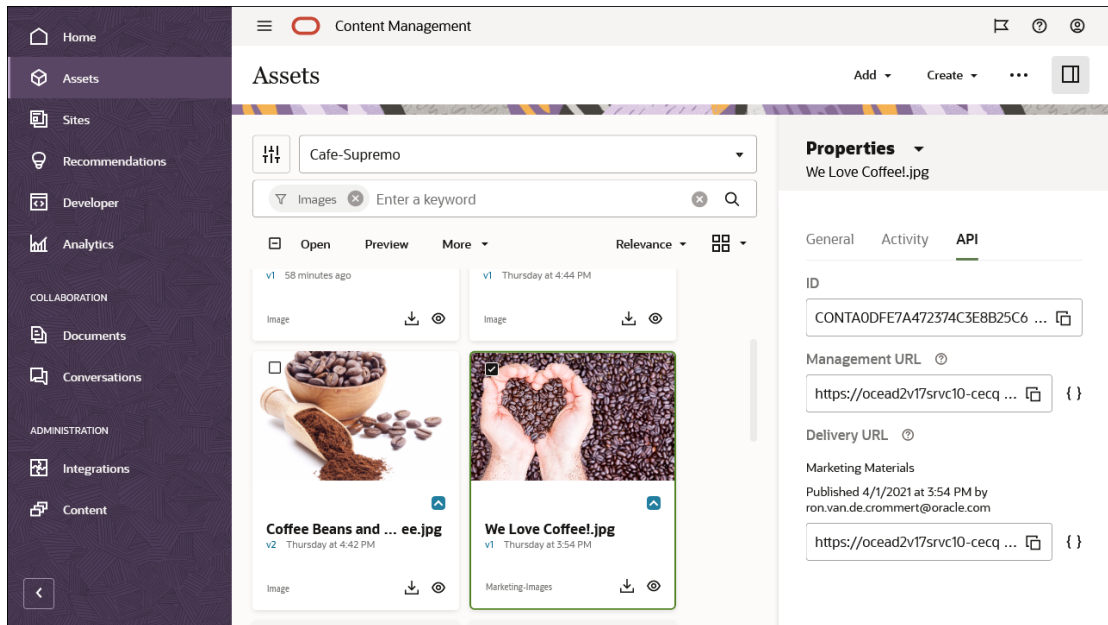




Content API for Digital Assets

Digital assets based on digital asset types behave exactly like any other asset in Oracle Content Management. They can participate in content workflows, they can be discovered and routed using API calls, and so on.

Once published, each digital asset is available as a REST resource. The address to the resource can be found as part of the content item properties.



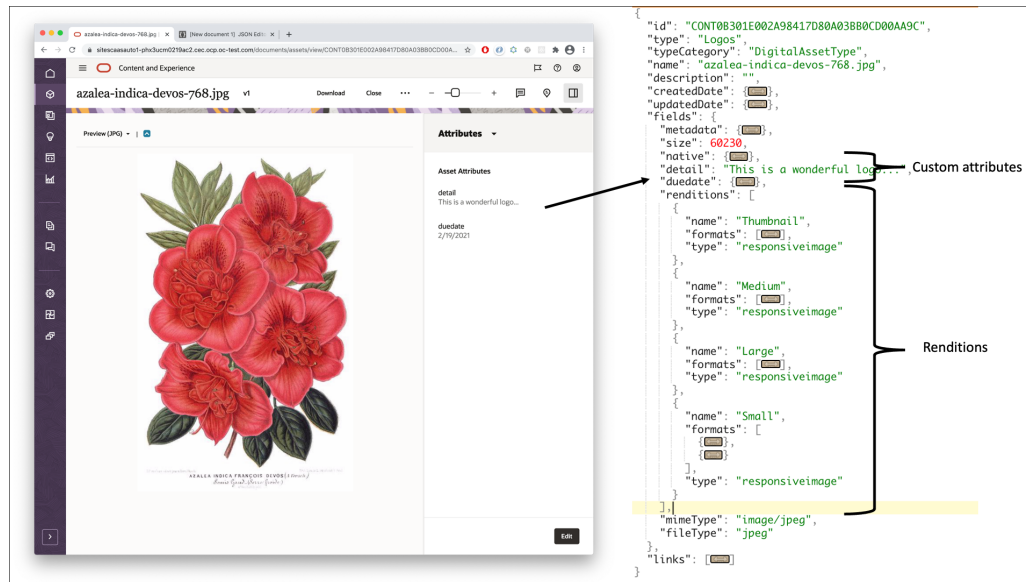
You can see the full JSON response data for an asset by clicking the { } brackets next to the delivery URL. Here's an example of what part of the JSON response looks like for a custom

digital asset (note "DigitalAssetType" in the typeCategory field and the custom digital asset type name in the type field):

```
{
  "id": "CONTA0DFE7A472374C3E8B25C6AC65F797F4",
  "type": "Marketing-Images",
  "typeCategory": "DigitalAssetType",
  "name": "We Love Coffee!.jpg",
  "description": "",
  "slug": "1481786684329-we-love-coffee!",
  "createdDate": {
    "value": "2021-04-01T22:54:46.569Z",
    "timezone": "UTC"
  },
  "updatedDate": {
    "value": "2021-04-01T22:54:46.569Z",
    "timezone": "UTC"
  },
  "fields": {
    "copyright": "(c) 2021 Acme Corporation. All rights reserved.",
    "metadata": {
      "width": "5184",
      "height": "3456"
    },
    "size": 11903181,
    "native": {
      "links": [
        {
          "href": "https://.../content/published/api/v1.1/assets/
CONTA0DFE7A472374C3E8B25C6AC65F797F4/native/We+Love+Coffee%21.jpg?
channelToken=1c92bb5b68b245da87ffb8672ff2fddb",
          "rel": "self",
          "method": "GET",
          "mediaType": "image/jpeg"
        }
      ]
    },
    "caption": "We love coffee!",
    "renditions": [
      {
        "name": "Thumbnail",
        "formats": [
          {
            "format": "jpg",
            "size": 0,
            "mimeType": "image/jpeg",
            "metadata": {
              "width": "150",
              "height": "100"
            },
            "links": [
              {
                "href": "https://.../content/published/api/v1.1/assets/
CONTA0DFE7A472374C3E8B25C6AC65F797F4/Thumbnail/We+Love+Coffee%21.jpg?
format=jpg&type=responsiveimage&channelToken=1c92bb5b68b245da87ffb8672f
```

```
f2fddb",
    "rel": "self",
    "method": "GET",
    "mediaType": "image/jpeg"
  }
],
},
{
  "format": "webp",
  "size": 0,
  "mimeType": "image/webp",
  "metadata": {
    "width": "150",
    "height": "100"
  },
  "links": [
    {
      "href": "https://.../content/published/api/v1.1/assets/
CONTA0DFE7A472374C3E8B25C6AC65F797F4/Thumbnail/We+Love+Coffee%21.jpg?
format=webp&type=responsiveimage&channelToken=1c92bb5b68b245da87ffb8672ff2fdd
b",
      "rel": "self",
      "method": "GET",
      "mediaType": "image/webp"
    }
  ]
}
],
"type": "responsiveimage"
},
{
  "name": "Medium",
  "formats": [
    {
      "format": "jpg",
      "size": 0,
      "mimeType": "image/jpeg",
      "metadata": {
        "width": "1024",
        "height": "682"
      }
    },
    [etc.]
```

Digital assets authored and classified as custom digital asset types have exactly the same API contract as the seeded asset types, including renditions and properties.



Learn More...

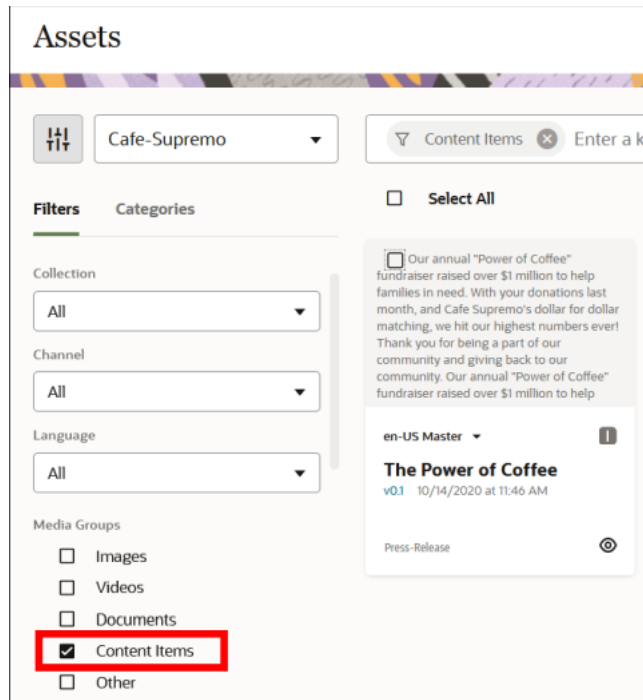
- Manage Asset Types
- Create a Digital Asset Type
- Add and Remove Assets

Content Item Types

A content type in Oracle Content Management basically represents a "content model" for a specific kind of content. The content model defines what constitutes a piece of content of that type.

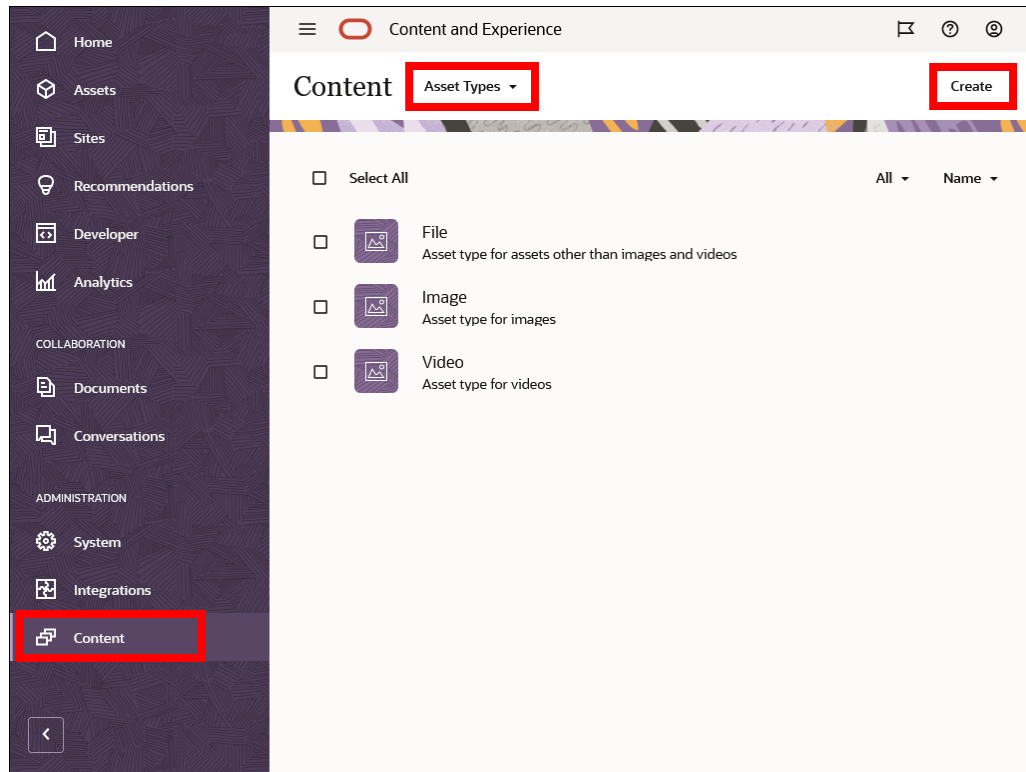
Say, you want to create a set of articles. Each article has a specific (structured) format: it has a title, a body, an author, and a picture. You can then define a content type that includes these properties. If content authors want to create an article, they must include all these elements for the article to be complete and valid.

Users can easily filter their repositories in the Oracle Content Management web interface to show only structured content items based on content types.



Create Content Item Types

If you have the required administrative privileges, you can create content item types in the Oracle Content Management web interface (under **Administration > Content**).

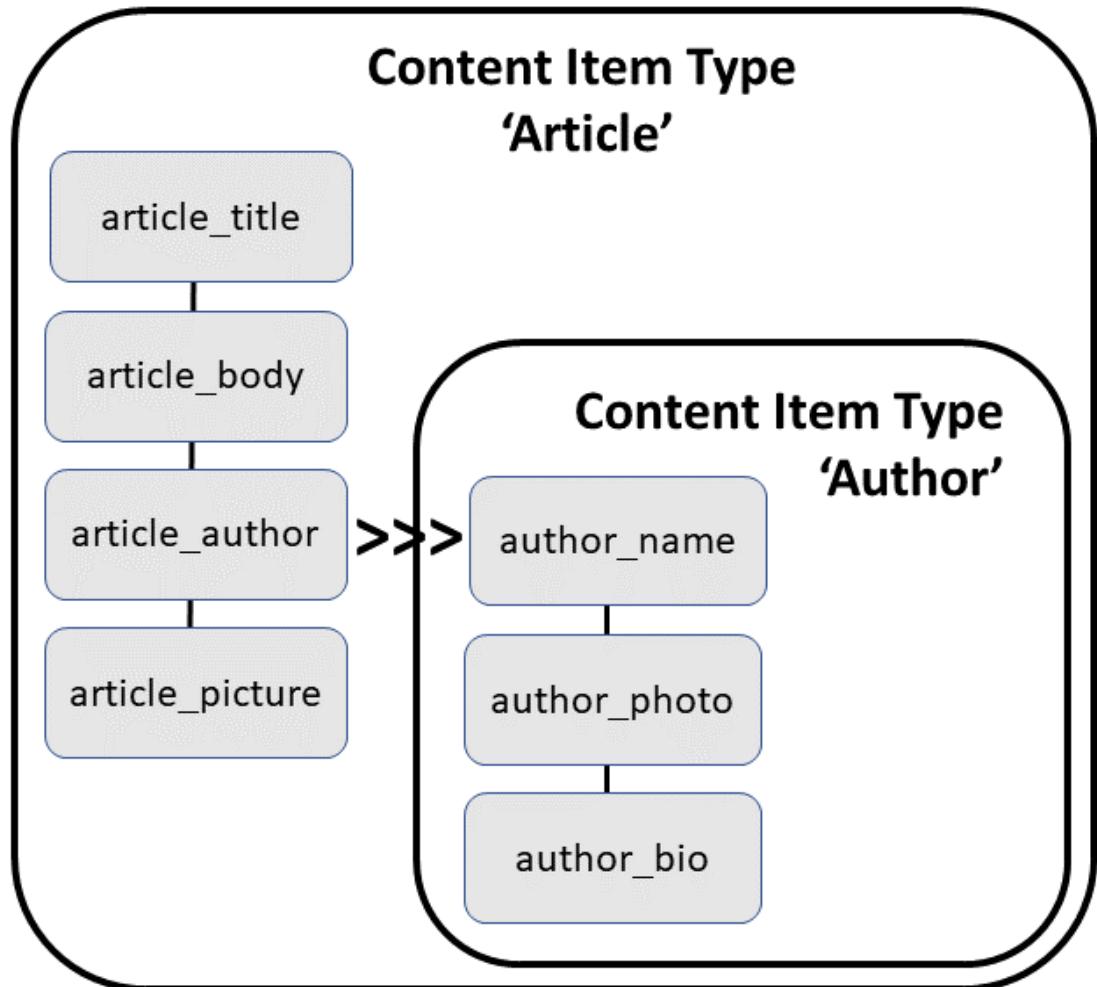


When creating a new asset type, you choose to create either a content item type or a digital asset type:

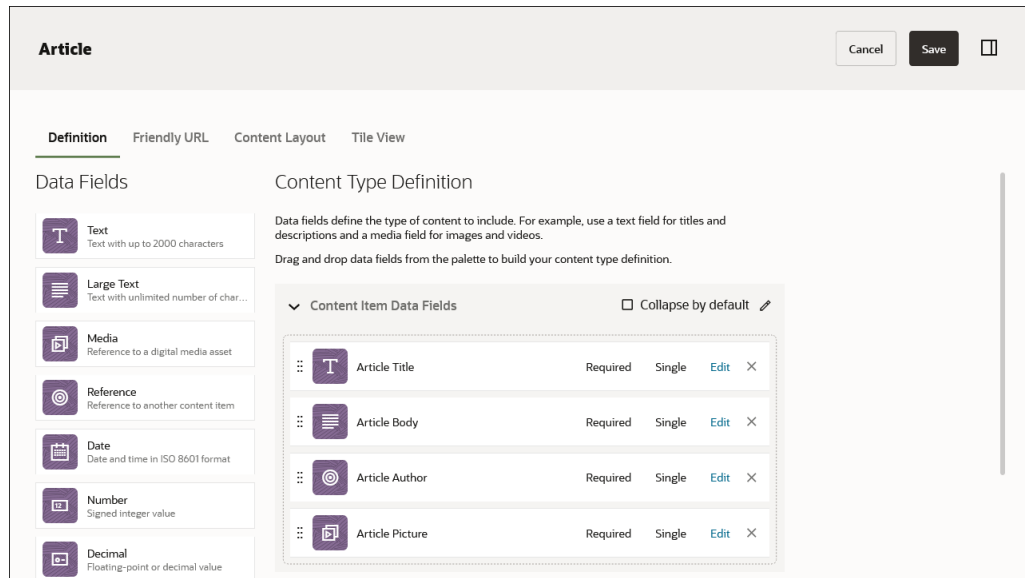
A screenshot of the 'Create Asset Type' dialog box. The title is 'Create Asset Type' with a close button (X) in the top right corner. The form contains three input fields: 'Name' with the value 'Article', 'Description (Optional)' with the value 'Content type for marketing materials', and 'Choose asset type' with two radio button options: 'Create a Content Item type' (which is selected) and 'Create a Digital Asset type'. At the bottom right are two buttons: 'Cancel' and 'Create'.

Let's have a closer look at a content item type called "Article". It could have four data fields: `article_title`, `article_body`, `article_author`, and `article_picture`. The Author data

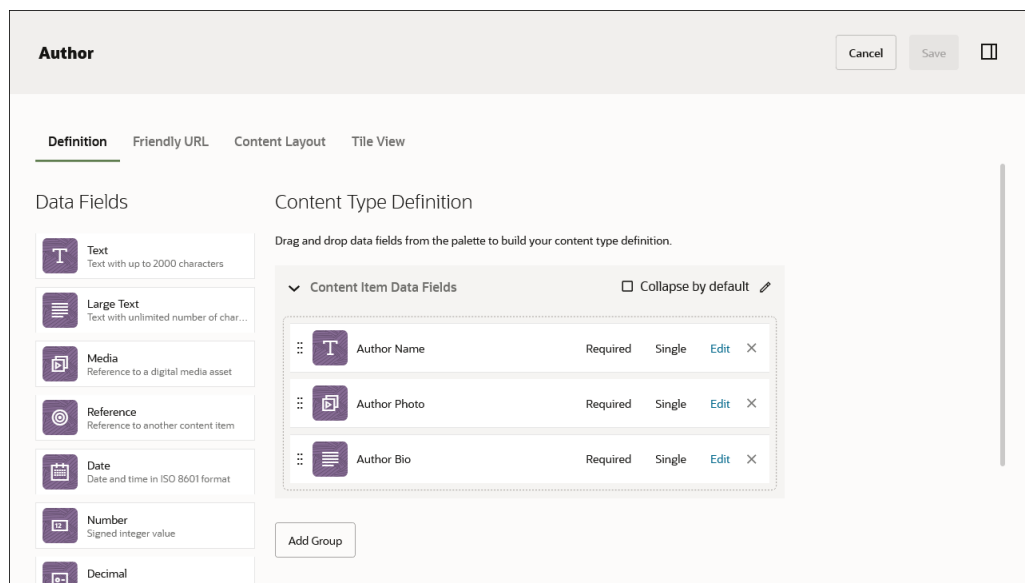
field is a reference to a content type called "Author", which itself has three data fields: author_name, author_photo, and author_bio



This is what the "Article" content item type definition looks like:



And here's the "Author" content item type:



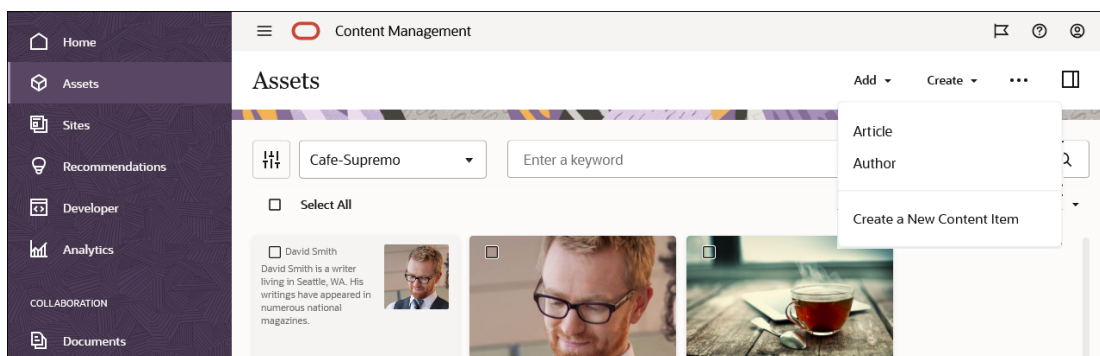
Each content item type consists of a set of field definitions. Fields support various data types and relationships along with constraints and validation rules. In the preceding example, four fields have been defined for the content item type named "Article": Article Title (`article_title`), Article Body (`article_body`), Article Author (`article_author`), and Article Picture (`article_picture`).

A content type can have any number of fields, and each field can be any of the supported data types (Text, Large Text, Media, Reference, Date, and so on). Fields can also have a single value or multiple values. Among field types, Reference is a special type. References make it possible for types to link to each other. For example, the Article content type has a field called Author of type Reference, which allows instances of the Author content type to be linked to instances of the Article content type. In relational database terms, this is similar to a foreign key. References, like any other field, can have multiple values. You can create sophisticated data models using content types and references.

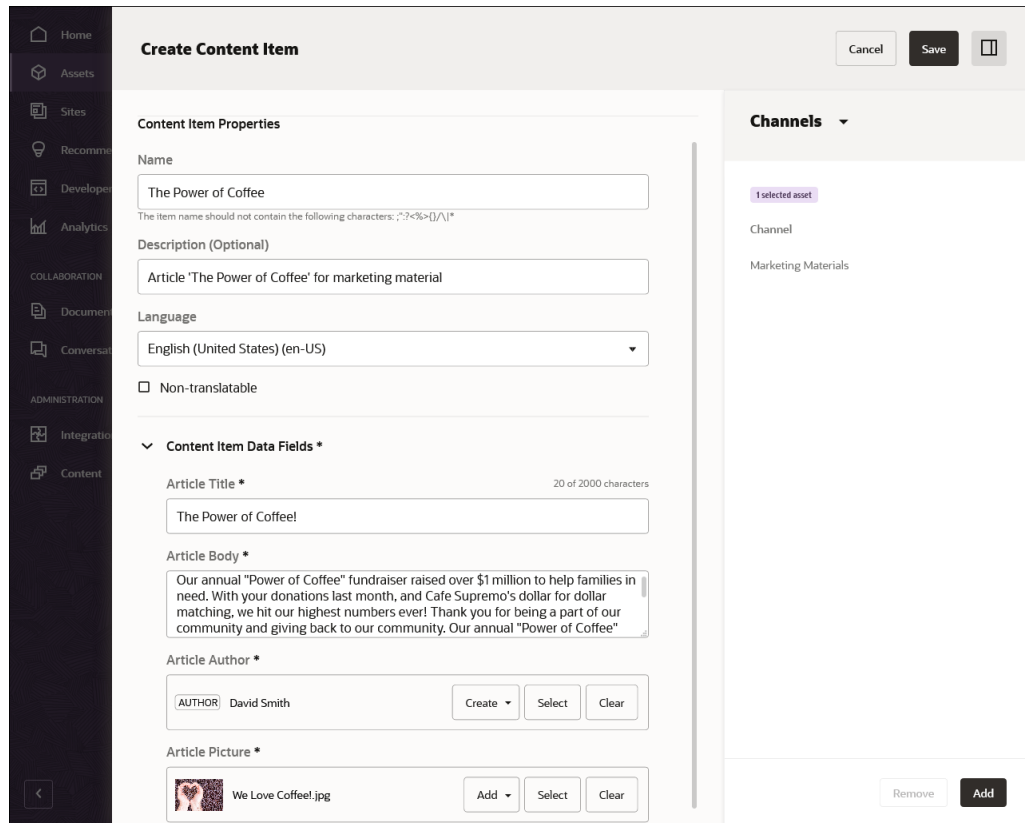
Oracle Content Management automatically creates data entry forms with the defined fields for a content type, along with content discovery and smart suggestions. Content authors use these forms to create pieces of content based on that content type. These structured content items are managed by Oracle Content Management as individual units, and they can be published and used in any channel just like any other content.

Create Content Items

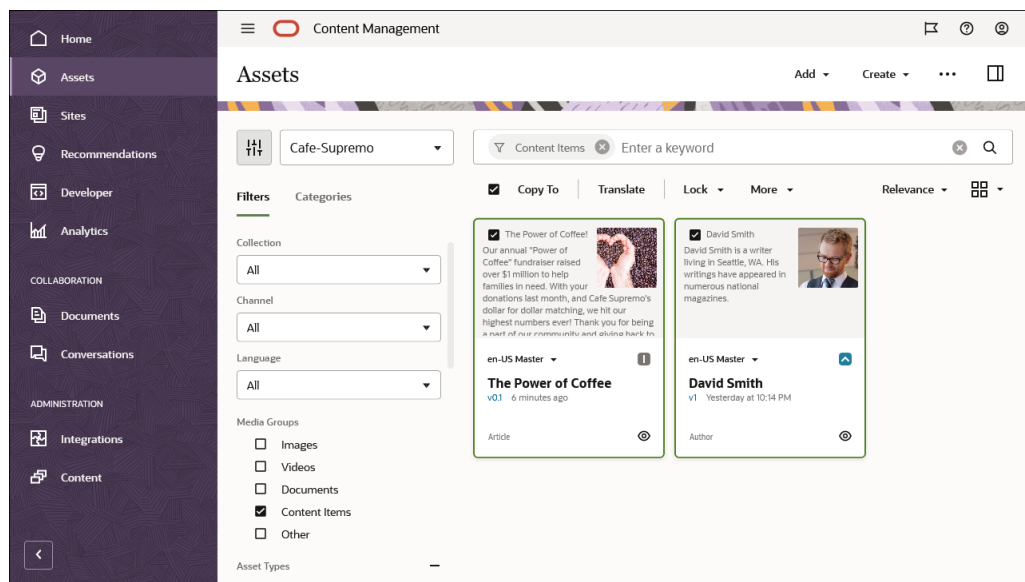
Once a content type is defined and made available to a repository, content authors can go into the repository that the content type is associated with and create a new content item based on the Author or Article content type, as well as any media content associated with them (for example, photos).



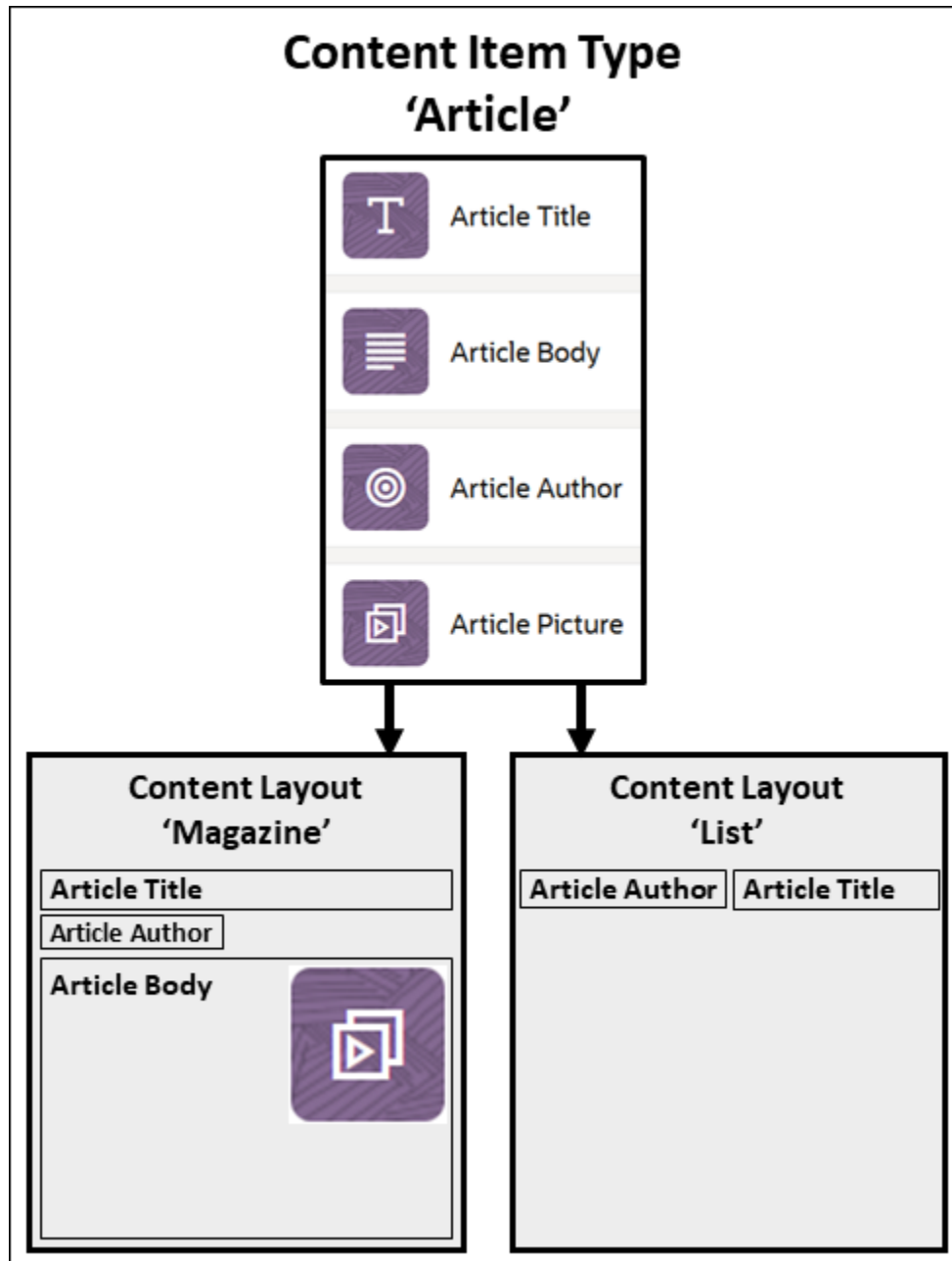
This will open auto-generated forms for creating assets of a specific content type, showing all data fields defined for that content type. For example, the Article form would look as shown in the following image, where each field type shows up with a specific editor (as configured when the content type was created), so content authors can create content in a way that makes sense.



Once created, all content items appear in the associated asset repository. From there, they can be published so they become available in their assigned channels.



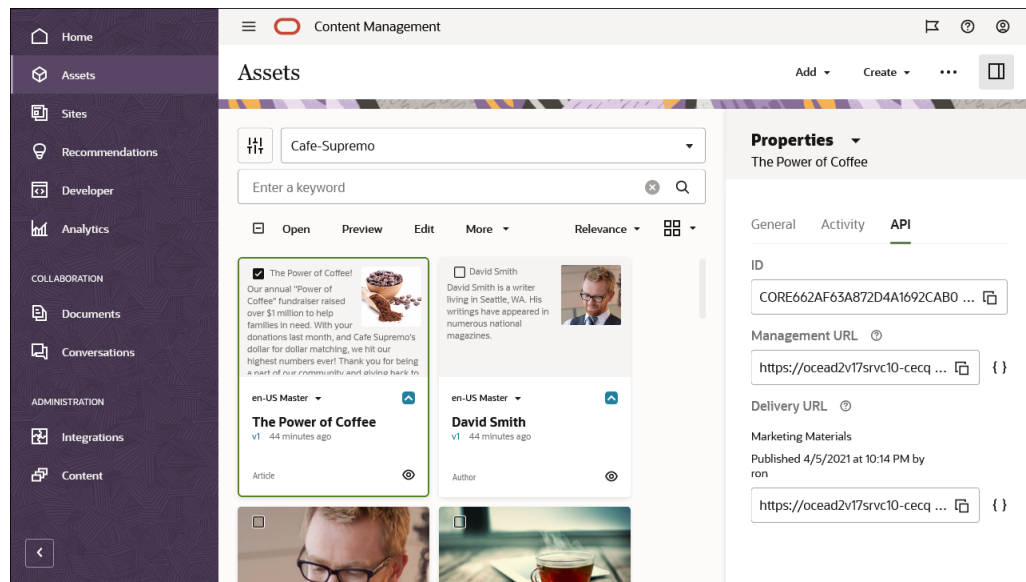
It's important to note that a content type only dictates what content of that type consists of, and not how the content is presented. For that purpose, a content type can have any number of content layouts associated with it, which determine how the content appears and what information is used in that particular layout. This allows for easy reuse of information.



Content API for Content Items

Content items behave exactly like any other asset in Oracle Content Management: they can be part of a workflow, they can be discovered and routed using API calls, and so on.

Once published, each content item is available as a REST resource. The address to the resource can be found as part of the content item properties.



You can see the full JSON response data for a content item by clicking the {} brackets next to the delivery URL. Here's an example of what the JSON response looks like for a content item (note "ContentType" in the typeCategory field and the content item type name in the type field):

```
{
  "id": "CORE662AF63A872D4A1692CAB0E9FA0AEFFE",
  "type": "Article",
  "typeCategory": "ContentType",
  "name": "The Power of Coffee",
  "description": "Article 'The Power of Coffee' for marketing material",
  "slug": "3000000150001-the-power-of-coffee",
  "language": "en-US",
  "translatable": true,
  "createdDate": {
    "value": "2021-04-06T05:14:02.433Z",
    "timezone": "UTC"
  },
  "updatedAt": {
    "value": "2021-04-06T05:14:02.433Z",
    "timezone": "UTC"
  },
  "fields": {
    "article_title": "The Power of Coffee!",
    "article_author": {
      "id": "CORE9D70A006F41D43268922AB55DBA0A100",
      "type": "Author",
      "typeCategory": "ContentType",
      "name": "David Smith",
      "links": [
        {
          "href": "https://.../content/published/api/v1.1/items/CORE9D70A006F41D43268922AB55DBA0A100?channelToken=1c92bb5b68b245da87ffb8672ff2fddb",
          "rel": "self",

```

```
        "method": "GET",
        "mediaType": "application/json"
    }
}
},
"article_picture": {
    "id": "CONTA37AC23CE5284C46ACF4D3C3B10A9950",
    "type": "Image",
    "typeCategory": "DigitalAssetType",
    "name": "Coffee Beans and Ground Coffee.jpg",
    "links": [
        {
            "href": "https://.../content/published/api/v1.1/items/
CONTA37AC23CE5284C46ACF4D3C3B10A9950?
channelToken=1c92bb5b68b245da87ffb8672ff2fddb",
            "rel": "self",
            "method": "GET",
            "mediaType": "application/json"
        }
    ]
},
"article_body": "Our annual \"Power of Coffee\" fundraiser raised
over $1 million to help families in need. With your donations last month,
and Cafe Supremo's dollar for dollar matching, we hit our highest numbers
ever! Thank you for being a part of our community and giving back to our
community. Our annual \"Power of Coffee\" fundraiser raised over $1 million
to help families in need. With your donations last month, and Cafe Supremo's
dollar for dollar matching, we hit our highest numbers ever! Thank you for
being a part of our community and giving back to our community."
},
"links": [
    {
        "href": "https://.../content/published/api/v1.1/items/
CORE662AF63A872D4A1692CAB0E9FA0AEFFE?
channelToken=1c92bb5b68b245da87ffb8672ff2fddb",
        "rel": "self",
        "method": "GET",
        "mediaType": "application/json"
    },
    {
        "href": "https://.../content/published/api/v1.1/items/
CORE662AF63A872D4A1692CAB0E9FA0AEFFE?
channelToken=1c92bb5b68b245da87ffb8672ff2fddb",
        "rel": "canonical",
        "method": "GET",
        "mediaType": "application/json"
    },
    {
        "href": "https://.../content/published/api/v1.1/metadata-catalog/items/
CORE662AF63A872D4A1692CAB0E9FA0AEFFE?
channelToken=1c92bb5b68b245da87ffb8672ff2fddb",
        "rel": "describedby",
        "method": "GET",
        "mediaType": "application/schema+json"
    }
]
```

```
]
}
```

Learn More...

- [Manage Asset Types](#)
- [Create a Content Type](#)
- [Add and Remove Assets](#)

Asset Repositories

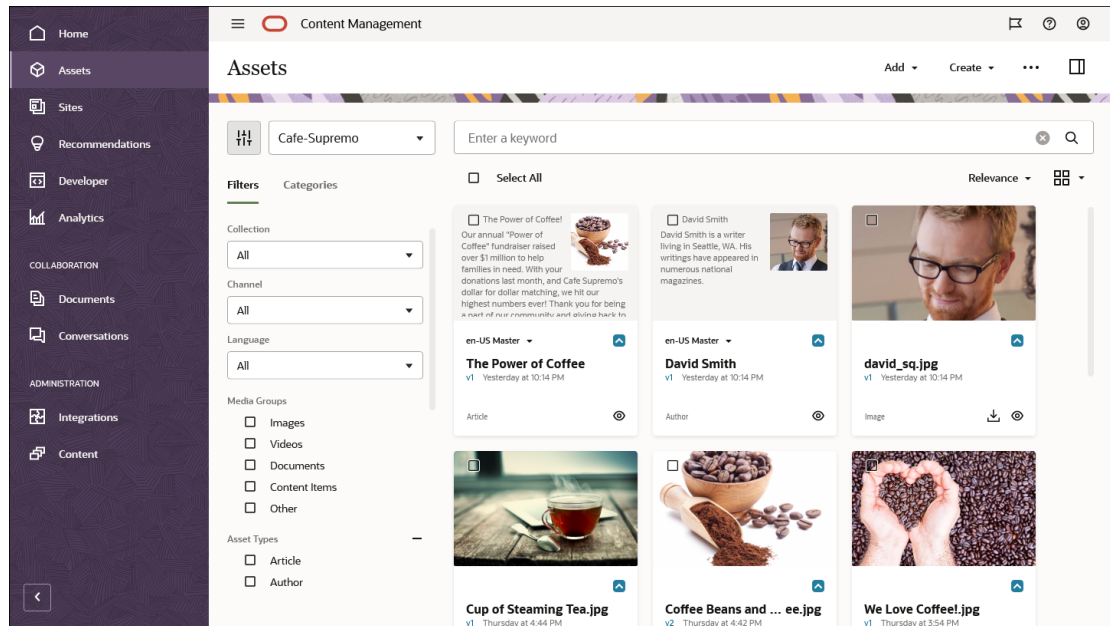
An asset repository is basically a big "bucket" in which content is managed. It's a container of all assets that an organization or team needs to work with—a conceptual entity that helps them organize and manage content.

Consider different departments in a company, such as sales, finance, and marketing. All these departments have their own teams of people working on content. Content from the finance department may not be relevant (and sometimes not even accessible) to people in the sales department, at least during part of the content life cycle. Content used by the marketing department can be accessible to teams working in other departments, but restricted only to review and not to modify or to publish.

Asset repositories help model the content in such scenarios. An organization could create separate repositories for each department and assign their respective teams as users of that repository with specific rights. When users sign in to Oracle Content Management, they are assigned to relevant repositories, and access privileges are granted to them for repositories, to allow for contribution, review, or approval. Some users might be involved with more than one department and may need access to content in multiple repositories.

All aspects of the content life cycle—including management, workflow, publishing, and revision tracking—are available in the context of asset repositories that assets are part of.

It's important to note that content delivery has no concept of repositories; published content is published to a channel regardless of which asset repository content originates from.



Administratively, a number of other entities are explicitly assigned to asset repositories:

- **Content types** are assigned to a repository to control which kinds of content can be created in that repository. This is helpful in controlling which types matter to a specific domain that the repository is supposed to work with.
- **Channels** are assigned to a repository to control the publishing targets for content in that repository.
- **Taxonomies** can be assigned to a repository, which allows authors to classify content in that repository. Additionally, the management interface shows classified content in a category view.

Please note that these are many-to-many relationships:

- A repository can publish to any number of channels, and a channel can be enabled for any number of repositories.
- Asset types are shared between repositories for which they are enabled.
- Taxonomy definitions can be shared between repositories.

Asset repositories are silos, which means that content from one repository can't be seen or referenced by another repository. It must be copied into another repository. On the other hand, assets from multiple repositories can be published to the same channel. This means that even though content in repositories is managed in silos, they can be consumed together in clients.

An asset repository is also a place for a number of other configuration settings:

- An asset repository controls which languages are available for content authors to create content in. Each repository also has a default language, which is the language that's assumed for all content if no language is specified.
- Configuration of content connectors, which are used to bring data into Oracle Content Management from an external system.
- On/off control of smart content discovery, which includes the ability to automatically tag content and search by them, recommend assets based on the content topic, and much more. This switch enables or disables certain background processing of content for

enrichment and discovery as well as behavior of the user interface and supporting APIs to surface relevant content.

Learn More...

- [Manage Repositories](#)

Videos

Oracle has partnered with Kaltura to provide built-in, advanced video management capabilities called Video Plus within Oracle Content Management.

This integration makes it easy to create, edit, and publish video content directly within Oracle Content Management, and then deliver that video content to any channel. The Video Plus integration feature needs to be enabled in the Oracle Content Management administration web interface: an administrator clicks the **Integrations** option in the left navigation menu and then opens the **Applications** page. Make sure the **Kaltura Video Management - Video Plus** option is enabled.

All video management, collaboration, and workflow are done within Oracle Content Management. Under the covers, though, the video content is converted on the Kaltura platform, and then delivered from Kaltura. This ensures that you can manage your videos as part of the central Oracle Content Management asset hub, include them as part of your sites and experiences, and deliver those videos in an optimized way to all your channels.

If you're creating your site in Oracle Content Management using Site Builder, then Oracle offers out-of-the-box components that you can use to deliver the videos to your sites. However, with more and more people taking advantage of Oracle Content Management as a headless content management system, the site you're building can be created using a tool or platform outside of Oracle Content Management. You can embed the video from Kaltura within any page and still take advantage of the Kaltura features in a way that's more customized for your audience.

At a high level, you can embed a video from Kaltura in a number of ways:

- Using the web browser's default HTML5 video support
- Using another custom video player
- Using the Kaltura JavaScript player

Embed the Kaltura player

The example below shows how to embed the Kaltura player in your page and then pull or load the published video content from Oracle Content Management. The process is pretty straightforward:

1. [Obtain the configuration parameters for the video](#)—partner ID, player ID (or user interface configuration ID), and entry ID for the video—through the REST API for Content Management.
2. [Add the JavaScript for the Kaltura player to your page.](#)
3. [Pass the configuration parameters](#) to the Kaltura player on your page.

Obtain the Configuration Parameters

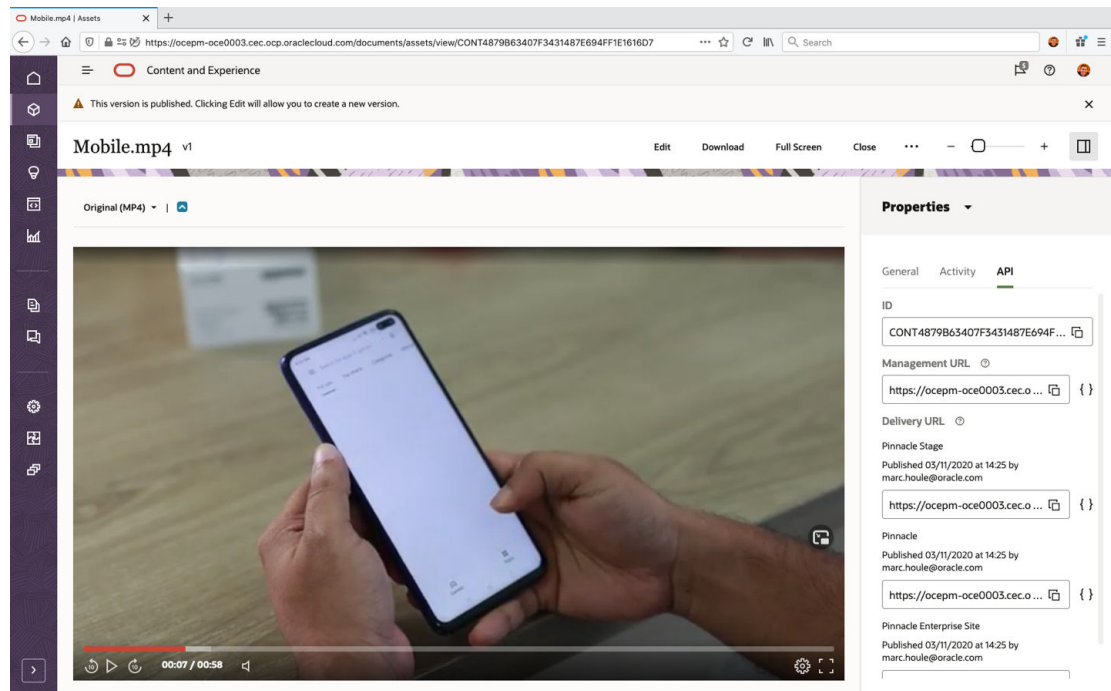
Before loading the player JavaScript library, you need to know the following IDs:

- Partner ID

- Player ID (or UI configuration ID)
- Entry ID of the item you want to include

If you have the ID for the video you want to embed, you can use the REST API for Content Management to extract all of this information.

If you don't have the video ID, you can use the REST API for Content Management or REST API for Content Delivery (for published content). Alternatively, you can get that information from the Oracle Content Management web interface directly; for example, by looking at the API properties of the Video Plus asset.



Then use the REST API for Content Management to get the video:

```
GET http://{host}/content/management/api/v1.1/items/{item_ID}
```

This will return data in the following format (not all data values are shown):

```
{
  "id": "CONT4879B63407F3431487E694FF1E1616D7",
  "type": "DigitalAsset",
  "name": "Mobile.mp4",
  "description": "",
  "fields": {
    "metadata": {
      "width": "1920",
      "height": "1080"
    },
    "advancedVideoInfo": {
      "provider": "kaltura",
      "properties": {
        "duration": 58,

```

```

        "videoStripProperties": "n15,h168,w300",
        "extension": "mp4",
        "searchText": "Mobile.mp4 ",
        "name": "Mobile.mp4",
        "status": "READY",
        "entryId": "1_9ijq4y5e",
        "endpoint": "https://www.kaltura.com",
        "partner": {
            "id": "2735841"
        },
        "player": {
            "id": "46290582"
        }
    },
    "renditions": [ . . . ],
    "mimeType": "video/mp4",
    "fileGroup": "AdvancedVideos",
    "version": "1",
    "fileType": "mp4"
},
{
    "status": "READY",
    "entryId": "1_9ijq4y5e",
    "endpoint": "https://www.kaltura.com",
    "partner": {
        "id": "2735841"
    },
    "player": {
        "id": "46290582"
    }
},
{
    "renditions": [ . . . ],
    "mimeType": "video/mp4",
    "fileGroup": "AdvancedVideos",
    "version": "1",
    "fileType": "mp4"
},
}

```

Note the entryID, partnerID, and playerID values that were returned (shown in bold above). You'll use these values to configure the player later.

Add the JavaScript Embed Code for the Kaltura Player

Add the embed code for the Kaltura player.

Pass the Configuration Parameters

Update the entryID, partnerID, and playerID placeholders with values that are specific to the Video Plus asset you want to embed. Additionally if you published the videos to a secure channel in Oracle Content Management, you will need to update the `ks` token field. But if you published your videos to a public channel, the token field is optional.

 **Note:**

Please obtain the token by using this [API call](#).

```
<div id="my-player" style="width: 640px;height: 360px"></div>

<script type="text/javascript" src="http://cdnapisec.kaltura.com/p/
{partnerID}/embedPlaykitJs/uiconf_id/{playerID}">
</script>

<script type="text/javascript">
try {
  var config = {
    targetId: "my-player",
    provider: {
      partnerId: {partnerID},
      uiConfId: {playerID},
      ks: {token}
    }
  };
  var kalturaPlayer = KalturaPlayer.setup(config);
  kalturaPlayer.loadMedia({entryId: '{entryID}'});
} catch (e) {
  console.error(e.message);
}
</script>
```

And that's it. The playerID value is the player configuration for the Kaltura player, as defined for playback within Oracle Content Management. However, you can modify the settings for the Kaltura player and do something different than the Oracle Content Management default.

Learn More...

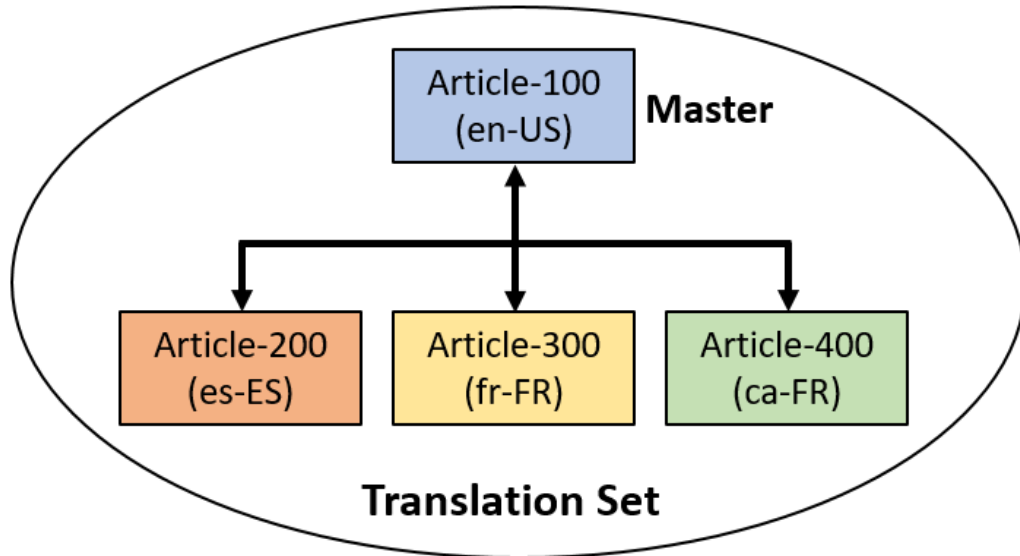
- [View and Manage Digital Assets](#)

Multilingual Content and Translations

Assets in Oracle Content Management can exist in any number of languages. When content authors create assets, they can choose from any languages that are enabled for the repository. A repository will always have a default language for assets, which is the language that's set if none is selected.

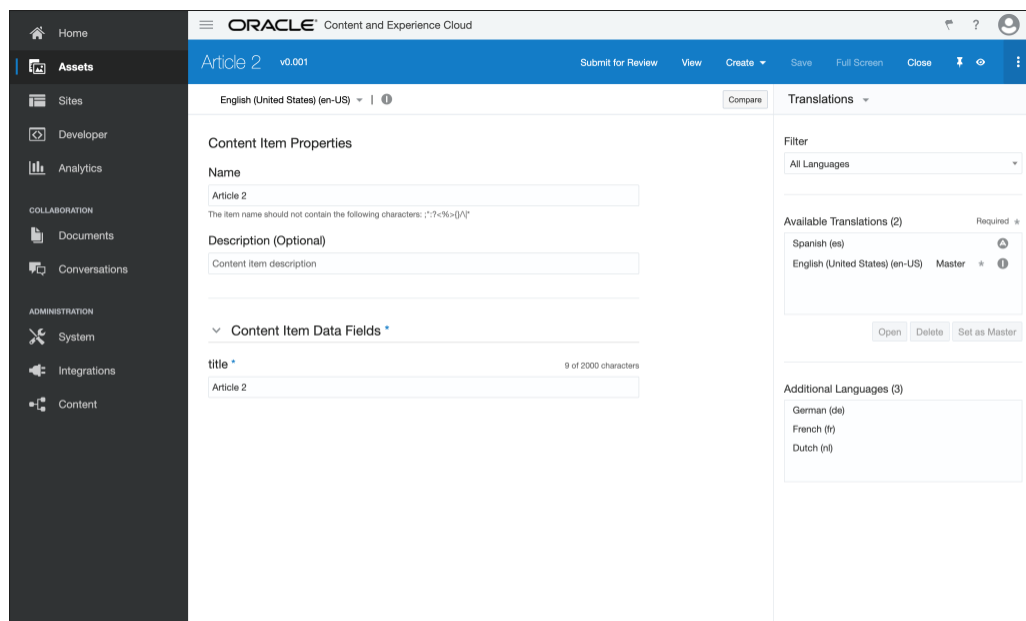
Assets can also be translations of each other. Suppose you have an article in English and want it translated to Spanish. You can then create a Spanish article, which is a translation of the English asset. An asset can have as many translations as there are languages enabled for the repository. All translations of an asset need to be of the same type.

A collection of translations of the same content item is called a *translation set*. A translation set has a master asset, which is usually the asset that was the source for the other translations. You can change translation sets and mark any asset of a set to be the master.



Alternatively, assets can be marked as nontranslatable. These assets don't have the option to add translations.

Translations of assets are managed in Oracle Content Management, as shown in the following image. In this example, the translation set contains two assets, with English as the master and Spanish its translation.



Each translation of an asset is a complete asset itself. Even though all translations are shown in the Oracle Content Management interface as one item, each translation is an asset in its own right behind the scenes. This means that translations can be created, updated, deleted, or published independently of any other assets in a translation set. (Publishing of translations is subject to localization policies defined for the publishing channel.)

When translations are published, they're available in the content delivery API. Furthermore, published translations continue to retain information about the translation relationship they have with the other assets in the translation set that have also been published.

Language Attributes of Assets

The content delivery and management APIs express the language and translatability of an asset in a translation set as standard fields in the API payload.

The following example shows the `language` and `translatable` fields.

```
{
  "id": "CORE6BEF829DE27F41AA9ACC0A3B6825D6C9",
  "type": "Article",
  "name": "Article 2",
  "description": "",
  "slua": "1481786051364-article-2",
  "language": "en-US",
  "translatable": true,
  "createdDate": {
    "value": "2019-12-05T14:55:18.788Z",
    "timezone": "UTC"
  },
  "updatedDate": {
    "value": "2019-12-05T14:55:18.788Z",
    "timezone": "UTC"
  },
  "fields": {
    "title": "Article 2"
  },
  "links": [
    {
      "href": "http://www.oracle.com/content/oraclecorp.com:8080/content/published/api/v1.1/items/CORE6BEF829DE27F41AA9ACC0A3B6825D6C9?channelToken=abb5f9190dd54a99979a7fa04b5506ba",
      "rel": "self",
      "method": "GET",
      "mediaType": "application/json"
    }
  ],
}
```

Discover Available Translations

You can discover the available published translations in the content delivery API by invoking the following `variations` resource endpoint.

```
GET
https://. . ./content/published/api/v1.1/{id}/variations/language?
channelToken=. . .
```

This produces a collection that is the translation set. Notice that `items[]` in the following example has two elements, which is the number of assets in this translation set. You can follow the links to each of the assets in the translation set.

```
{
  "setId": "98F6BD881DF46650E053020011ACE0E4",
  "masterItem": "CORE6BEF829DE27F41AA9ACC0A3B6825D6C9",
  "items": [
    {

```

```
    "id": "CORE6BEF829DE27F41AA9ACC0A3B6825D6C9",
    "value": "en-US",
    "links": [
      {
        "href": "http://. . .oraclecloud.com:8080/content/
published/api/v1.1/items/. . .?channelToken=. . .",
        "rel": "self",
        "method": "GET",
        "mediaType": "application/json"
      }
    ]
  },
  {
    "id": "CORE8D451A7D43FE401EAC96A80E111406B4",
    "value": "es",
    "links": [
      {
        "href": "http://. . .oraclecloud.com:8080/content/
published/api/v1.1/items/CORE8D451A7D43FE401EAC96A80E111406B4?
channelToken=abb5f9190dd54a99979a7fa04b5506ba",
        "rel": "self",
        "method": "GET",
        "mediaType": "application/json"
      }
    ]
  }
],
"links": []
}
```

Fetch a Specific Translation of an Asset

In a client application, you'll often want to switch to translated content automatically without listing translations.

This can be done in a single call, like the following one:

```
GET https://. . ./content/published/api/v1.1/{id}/variations/
language/es?channelToken={. . .}
```

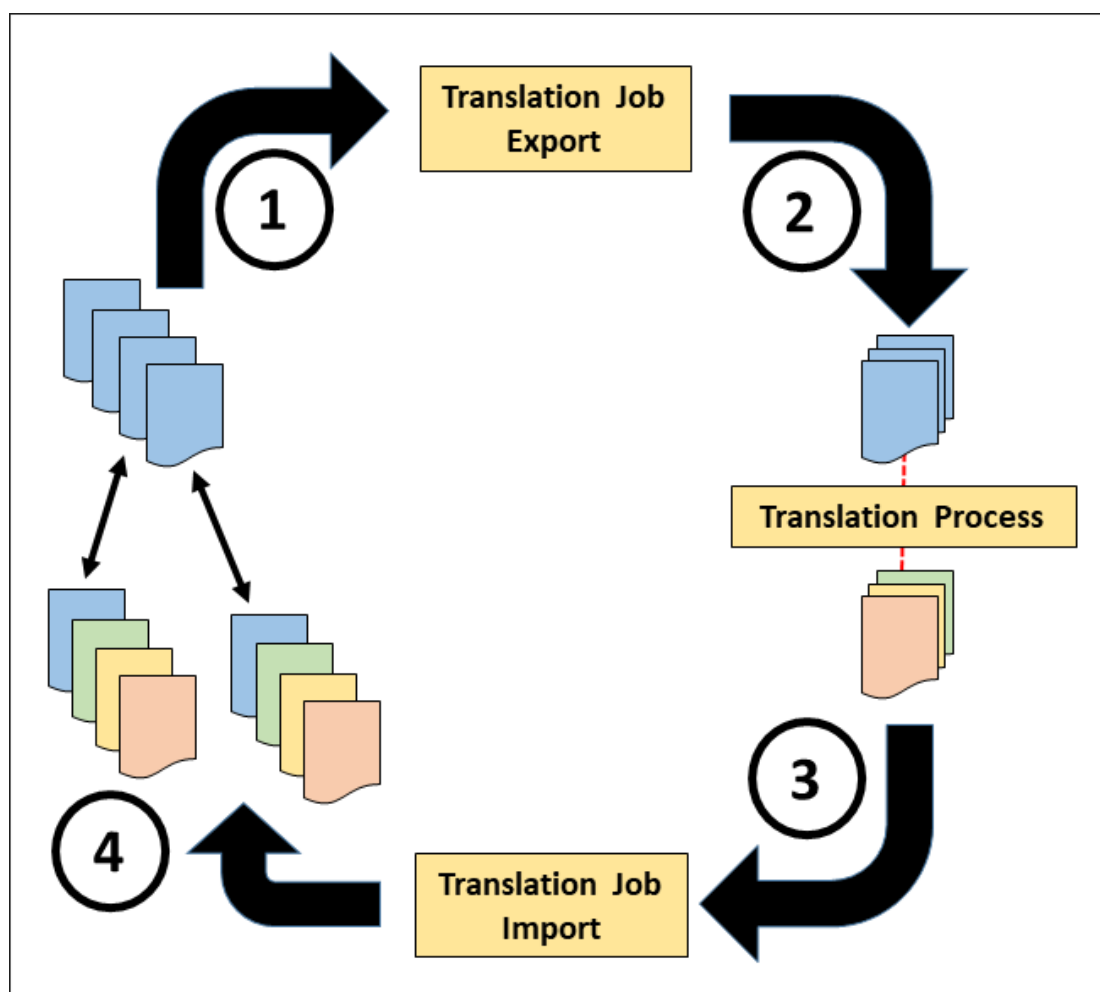
The response for this request would be the Spanish (`es`) asset in the translation set. If the client asks for a translation that doesn't exist, the result would be a 404 error response, as expected. However, you can configure Oracle Content Management to fall back to a default language instead of returning a 404 error response. This can be done by associating a localization policy with the channel. If a localization policy with a default language is associated with the channel, then requests for nonexisting translations will return the asset of the default language.

Translation Jobs

It's often necessary to translate a large number of assets in bulk, possibly using third-party tools or translation teams. Oracle Content Management supports this process through translation jobs.

This process involves a number of steps:

1. First, you add assets to be translated to a translation job, which is a trackable activity through the life cycle of asset translation.
2. Next, the translation job exports the translatable assets into an export file. This is an archive (zip file) of all contents of the asset that needs to be translated. A third party or an integration can inspect this archive and provide the desired translations. These translations are themselves added to the same archive as part of the translation process.
3. The archive with the translated assets is then imported back into Oracle Content Management. Upon import, translated assets are created and linked to their master/original assets and with each other to create translation sets.
4. The translations are now in Oracle Content Management, linked to their master asset and available for further processing and publishing.



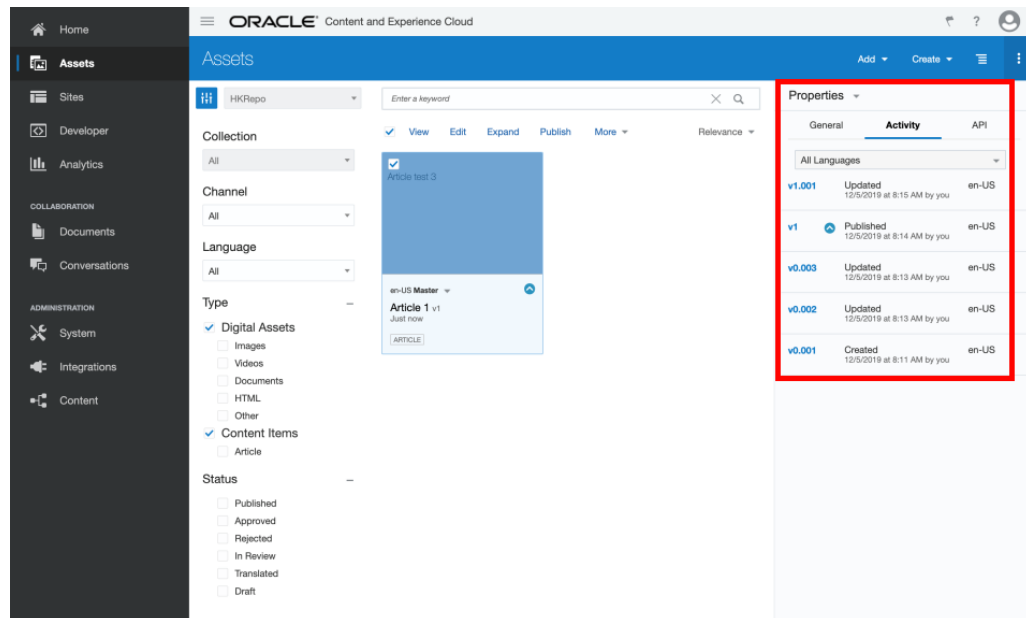
Learn More...

- [Work With Language Assets](#)
- [Review and Add Languages to a Content Item](#)
- [Localize Content Items](#)
- [Manage Asset Translation Jobs](#)

Content Versions

Changes to content in Oracle Content Management are tracked as revisions. This means Oracle Content Management maintains a snapshot, or version, of each update to an asset.

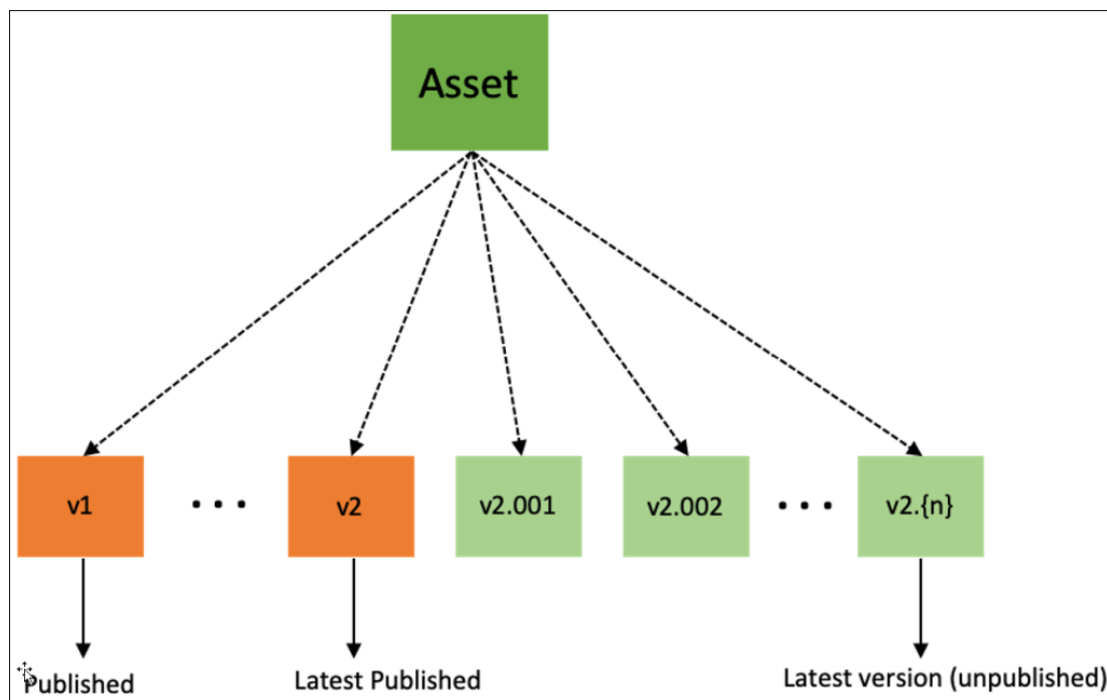
The content snapshots of an asset are readily available to the content author in the **Properties** panel on the **Activity** tab, as the following image shows.



Content authors can see content versions either for all languages or just for the currently selected asset (**Current Language**).

Please note the numbering of content versions, which follows a very specific format: *{published version}.{local version}*. For example, v1.002 means that this asset has been published at version 1 and two subsequent updates exist for this asset, but those changes have not yet been unpublished (local versions). If an asset was never published, its published version is 0.

When an asset is published, its published version number is set to the next number. For example, if an asset goes through updates marked v0.001, v0.002, and v0.003, and is then published, the new version for the current asset will be v1.000 (marked simply as v1). Subsequent numbering continues with v1.xxx as local updates are made to the asset (v1.001, v1.002, and so on).



Even though all content versions are retained in Oracle Content Management, only the current (latest) version is available for content management operations. This means that only the latest version of the asset can be approved or published or associated with other assets. Older versions exist for the purposes of tracking and review by content authors, not for actual use of that content.

Learn More...

- View and Manage Content Items
- View and Manage Digital Assets

Publishing and Channels

Publishing content is the act of making that content available to clients. It means that the content is accessible from delivery APIs in read-only form. Publishing is a *copy* operation—that is, users can freely change an asset after publishing without affecting the client.

A channel is a logical entity to which a publishing operation is initiated.

The act of publishing involves three key elements:

1. The actual **assets** that are being published
2. The **channel** to which the assets are being published
3. The **policies** that govern the publishing operation

Publishing

Any asset that's managed in an asset repository can be published to one or more channels. Once published, content is available to the delivery API in read-only form.

It's important to note that even though published content retains the same identifier as managed content, a published asset is essentially a copy (or snapshot) of the managed

asset. This means that after a piece of content has been published, it can continue to change in the management context, but those changes are not reflected in the published entity until the content is republished.

Publishing assets involves dependency management at its core. One of the key tenets of publishing is to never cause broken links in the published content. Say there is a blog asset that's published. If that asset has references to an image or other assets, then these must also be published along with the blog asset. This ensures that the published content is always self-contained and complete.

Publishing always carries delta data only since the last publishing operation to the same channel. If a referenced asset is already published to the same channel, then only the originating asset is published. This is the case even if the referenced asset has changed since the last publishing operation. This way each asset can have its own change management life cycle, while referential integrity is maintained at the same time.

Channels

Channels are logical entities to which content can be published. Channels can be thought of as containers for configuration and policies related to publishing content.

Channels can be public or secure. **Public channels** allow access to published content by any user. They are intended primarily for use in applications such as public websites and apps.

Secure channels allow content access only to users who are logged in. Secure channels are intended for use in more closed contexts, such as intranets, employee portals, and secure applications.

Each channel has a **channel token** that a client must present to get access to published content. The token can show up either as a query parameter or as an `http` header. This token is not an encrypted entity (that is, it does not represent secure or shared knowledge between a client and server); it's merely an identifier of the channel.

Policies

Policies control the nature of the validation process that happens before assets are published.

There are two kinds of policies:

- **Workflow policies**—A channel can be configured to accept any published content or only approved published content. In some situations, it might be necessary to safeguard against inadvertent publishing of content that might not be ready. A workflow policy helps ensure that each asset that ultimately gets published, either directly or indirectly (through a reference), is explicitly approved by an approving authority.
- **Language policies**—There might be situations where the publishing of content requires translations of that content to be published at the same time. For example, there might be legal requirements to publish content in a set of languages and ensure that each translation is up to date and published together with the other translations. A language policy ensures that such constraints can be met and safeguards against accidental publication of content without the required language translations. Language policies are managed as a separate

configuration set and assigned to a channel. See Multilingual Content for further details.

Publishing Process

Publishing involves a number of steps. It's useful to look at this from a flow perspective to understand how and when policies come into play.

Publishing starts with the user initiating a publish operation, either for a single asset or a set of assets. The first step in the publishing process is to gather all necessary assets for publication. This involves recursively walking through the content relationships to gather all unpublished related content.

Once all content to be published is gathered, a verification step happens to ensure that workflow policies, language policies, user rights, and so on are all validated.

After the content and the required publication permissions have been validated, the content is actually published, which basically creates a copy of the content. All published content is stored safely so that no broken links exist in published content. Once the publishing process is complete, the delivery API would be able to serve clients with the newly published content.

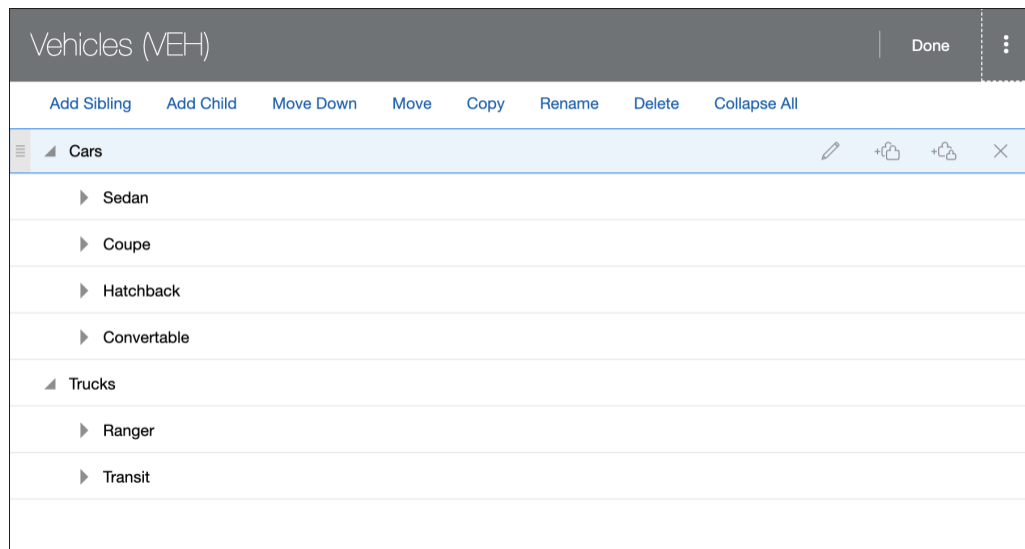
Learn More...

- [Publish Assets](#)
- [Manage Asset Publishing Jobs](#)
- [Manage Publishing Channels](#)
- [Manage Workflows](#)
- [Manage Localization Policies](#)
- [Content Delivery](#)

Taxonomies

A taxonomy is a hierarchical grouping of related concepts. In Oracle Content Management, taxonomies help content authors and client applications classify content into well-defined categories.

Let's have a closer look by taking a vehicles taxonomy as an example.



In this example, the Vehicles taxonomy has two top categories (Cars and Trucks), and these categories in turn have several children. Of course, those child categories can each have their own child categories, and so on. Such a structure of logical entities essentially represents a hierarchical set of categories.

The following topics describe Oracle Content Management taxonomies:

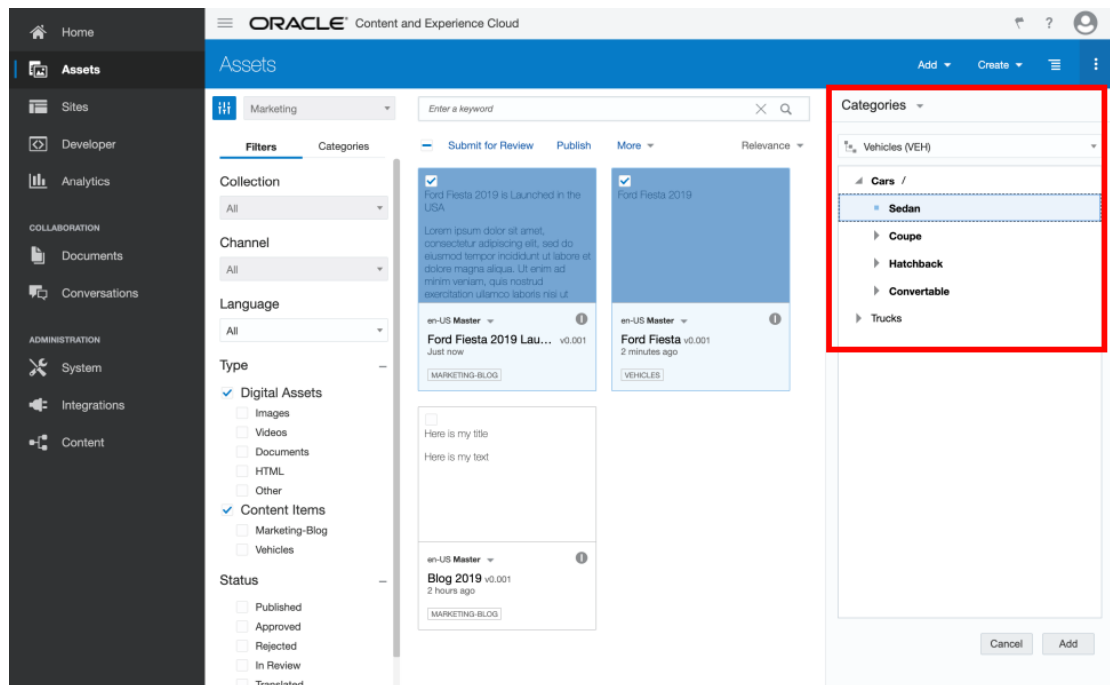
- [Taxonomies from a Management Perspective](#)
- [Taxonomy Life Cycle](#)
- [Taxonomies from a Delivery Perspective](#)
- [Discovering the Structure of a Taxonomy](#)
- [Discovering Asset Categorization](#)

Taxonomies from a Management Perspective

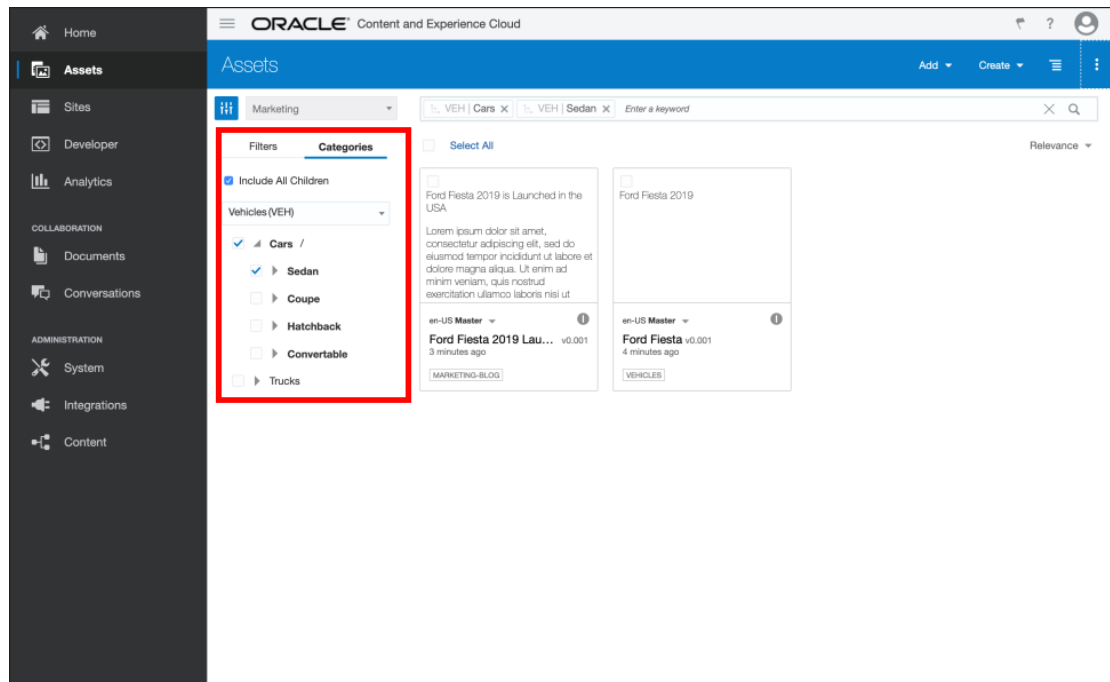
Once a taxonomy and categories are defined, content authors can classify content into categories of that taxonomy.

For example, an asset called `Ford Fiesta` would be classified under `/Vehicles/Cars/Sedan`. Any number of assets of any kind can be classified into a category. A category is merely a logical placeholder for content that belongs to a specific concept.

Adding assets to a category is simple: select the assets to be added to a category and choose the category to add them to.

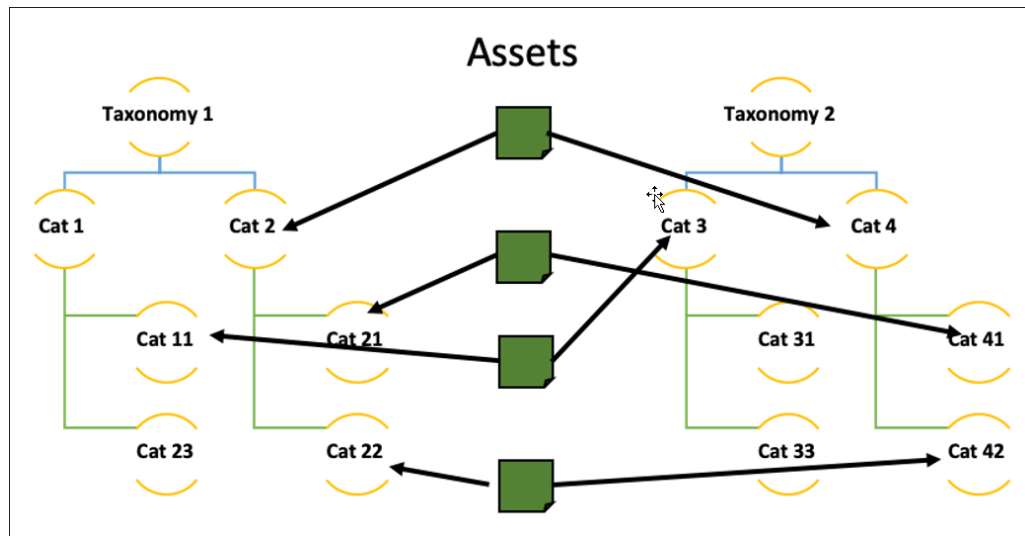


Once assets are added to categories, the Oracle Content Management web interface shows them in an intuitive tree-like navigation structure. Content authors can select a specific category to view or manage its content. This lets you use taxonomies as an organization tool for content management.



An Oracle Content Management instance can have as many taxonomies as needed. A single repository can have multiple taxonomies, and the same taxonomy can be enabled in multiple repositories. In addition, assets in a management context can belong to multiple taxonomies.

This *many-to-many* relationship between taxonomies, repositories, and assets helps content architects design and use sophisticated content-classification models.



Taxonomies themselves might change over time. When a taxonomy changes, the categorization of the assets also changes as a result (automatic recategorization). Suppose a category is moved from one parent category to another. In that case, assets that belong to the category that moved (children of the category) continue to belong to the same parent, but would acquire a new grandparent. If a category is deleted, however, the assets belonging to that category are promoted to the parent category, as you would expect.

In this sense, categories are not containers of assets. Instead, the right way to think about categorization is in terms of relationships. Assets, by virtue of categorization, are associated with a category.

Taxonomy Life Cycle

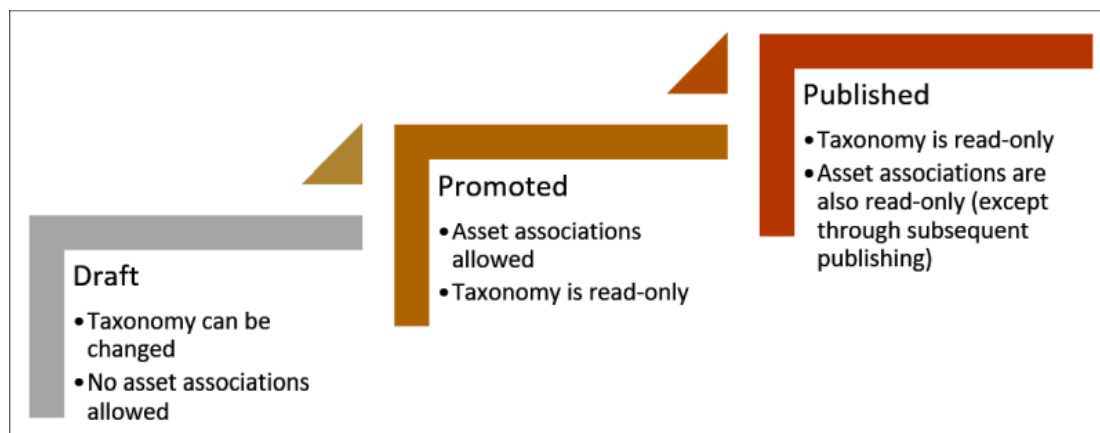
Changes to a taxonomy itself can have far-reaching consequences for the nature of categorization. Such taxonomy changes are likely to be a multistep process involving a number of people and potentially multiple groups or teams.

That's why Oracle Content Management uses an orchestrated life cycle for taxonomy changes:

- Taxonomies are always created in a *draft* state. They are initially considered to be in the process of structural definition and don't allow assets to be added to their categories.
- Once the taxonomy authors are satisfied with the structure of a taxonomy, they graduate it to a *promoted* state. This is the state at which the taxonomy can be enabled for a repository, which allows content authors to start classifying content into it. However, no changes to the taxonomy structure itself are allowed in this state.
- If the structure of a taxonomy needs to change after assets have been classified into it, that taxonomy needs to be versioned and turned back into draft mode. However, users can continue to categorize content into the promoted version of the taxonomy. The draft taxonomy can subsequently be promoted, which results in

automatic recategorization of assets into the new structure of the taxonomy (using the general rules described in the preceding text).

- A taxonomy itself can be published to channels like any other asset. Once published, taxonomy and categorization information is available in the delivery API. This helps clients present content based on taxonomy (such as product listings, faceted navigation, and some types of menus).



Taxonomies from a Delivery Perspective

Taxonomies are published, just like assets. Once a taxonomy is published, it's available in the delivery API.

There are two places where taxonomy information surfaces in the delivery API:

1. [Discovering the Structure of a Taxonomy](#)
2. [Discovering Asset Categorization](#)

Discovering the Structure of a Taxonomy

There are many APIs that help navigate the structure of a taxonomy or category by category listing.

One simple way to get all category information of a taxonomy is using an API in this form:

```
GET http://.../content/published/api/v1.1/taxonomies/
52446BF67CE74F229AF9F178448BCB80/
categories?fields=ancestors&channelToken=c240e6a4209946549a9eef53c8ed3ab0
```

Discovering Asset Categorization

The standard asset-listing resource can also list assets classified to a specific category (or categories).

This is done with this simple form of the API:

```
GET http://.../published/api/v1.1/items?q=(taxonomies.categories.id eq
"77BD937EFCA54051905DD6D525E37E58")&channelToken=c240e6a4209946549a9eef
53c8ed3ab0
```

This produces a response like the following one:

```
{
  "items": [
    {
      "name": "Ford Fiesta",
      "description": "",
      "links": [],
      "id": "CORE23B05BB961AE4EE3AE4ED9962A0440ED",
      "type": "Vehicles"
    },
    {
      "name": "Ford Fiesta 2019 Launch",
      "description": "",
      "links": [],
      "id": "COREC6C72D74EAA0477AAE966CF154CB16C7",
      "type": "Marketing-Blog"
    }
  ],
  "links": []
}
```

To make this type of request, you would have to know the ID of the category. However, there is another way to discover assets of a category by using a friendly ID (slug or API name). Each category can have a unique handle, which is a string that's both human-readable and addressable as an ID. The API name of a category can simply be added to a category at the time of creation or any time thereafter. The API name itself can be used instead of the category ID. Also note that the API name of a category, like its ID, must be unique across taxonomies.

Suppose we have `all-cars` as the API name for the `Cars` category and `all-cars.sedans` for the `Cars/Sedan` category. Then the preceding API call can also be written as follows:

```
GET http://.../content/published/api/v1.1/items?
q=(taxonomies.categories.apiName eq
"all-cars.sedans")&channelToken=c240e6a4209946549a9eef53c8ed3ab0
```

This lets client applications surface meaningful URLs instead of passing around IDs.

Learn More...

- Manage Taxonomies
- Assign Asset Categories

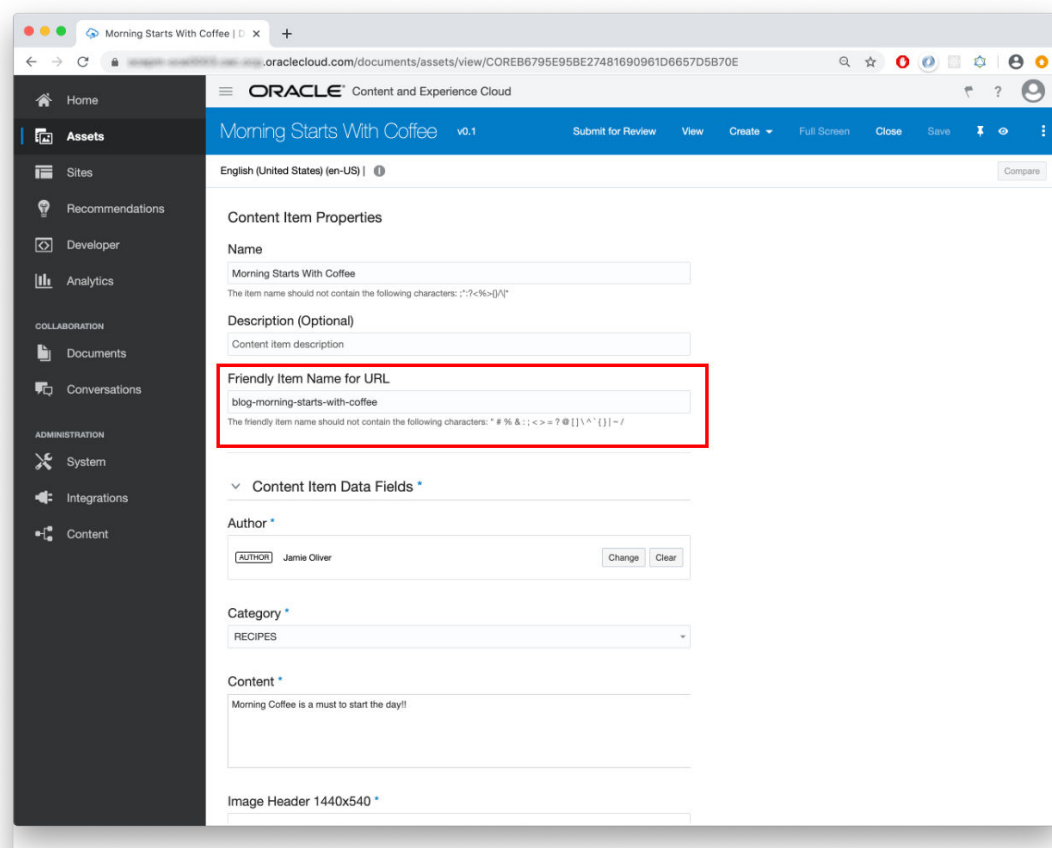
Friendly URLs for Assets

Ofentimes an [asset](#) is all that's needed to render a whole page, such as a blog post. In situations like that, it would be nice to have the URL of the web page contain the title of the blog.

Suppose I have a blog post titled "Morning Starts with Coffee". I would want the URL of the page that shows this blog post to be something like `https://.../blog-morning-starts-with-coffee.html`. This is very easy to accomplish with Oracle Content Management because you can enable friendly URLs for content types.

Once friendly URLs are enabled for a content type, a new field is added to assets of that type. Furthermore, Oracle Content Management automatically populates the friendly URL field with the asset name. As soon as I create a blog asset, its "Friendly Item Name for URL"—also called *slug* in the API—shows the name of the asset (with some automatic character replacements for better URL handling).

Of course, you can change this friendly URL to anything you want. Oracle Content Management handles all the necessary validations to make sure it's unique.



This *slug* field is now part of the asset's data. When you get this asset (say, from a search list), you can fetch the slug field and use that in your client to construct the URL of a page.

More interestingly, Oracle Content Management provides a way to fetch the asset using just the slug value. For example, instead of accessing this blog asset with

```
https://.../content/published/api/v1.1/items/  
COREB6795E95BE27481690961D6657D5B70E?  
channelToken=83e8e3bc1f5f55f9b02a8183622589ad
```

you can use the slug in the URL:

```
https://.../content/published/api/v1.1/items/.by.slug/blog-morning-  
starts-with-coffee?channelToken=83e8e3bc1f5f55f9b02a8183622589ad
```

This means that if the slug of an asset is in the URL of a page, all that the client needs to do is grab that string and request the underlying asset by slug. This saves a round trip to *look up* or search for an asset with a slug and then fetch its assets.

It's useful to note that every variation of the API that exists for an `/items/{id}` also exists for `/items/.by.slug/{value}`. These two APIs are therefore equivalent from the perspective of the content delivery API.

Learn More...

- [Create a Content Type](#)
- [Create a Digital Asset Type](#)

Quick Start

Oracle Content Management is an enterprise-grade content management system (CMS) and intelligent content platform that can serve a wide variety of content management and delivery needs. This includes headless environments, where it's leveraged primarily as a content server for consumption by other applications, including websites and other digital experiences.

In those scenarios, Oracle Content Management can support entire digital ecosystems with a single source of truth for content.

To accomplish this, Oracle Content Management offers a rich set of [application programming interfaces \(APIs\)](#) that allow arbitrary read and write queries of a variety of resources managed by the system.

It's easy to get started with Oracle Content Management as a headless CMS. This quick start tutorial is for developers who want to get up and running quickly and dive into the content delivery and management APIs provided by Oracle Content Management. It basically consists of these steps:

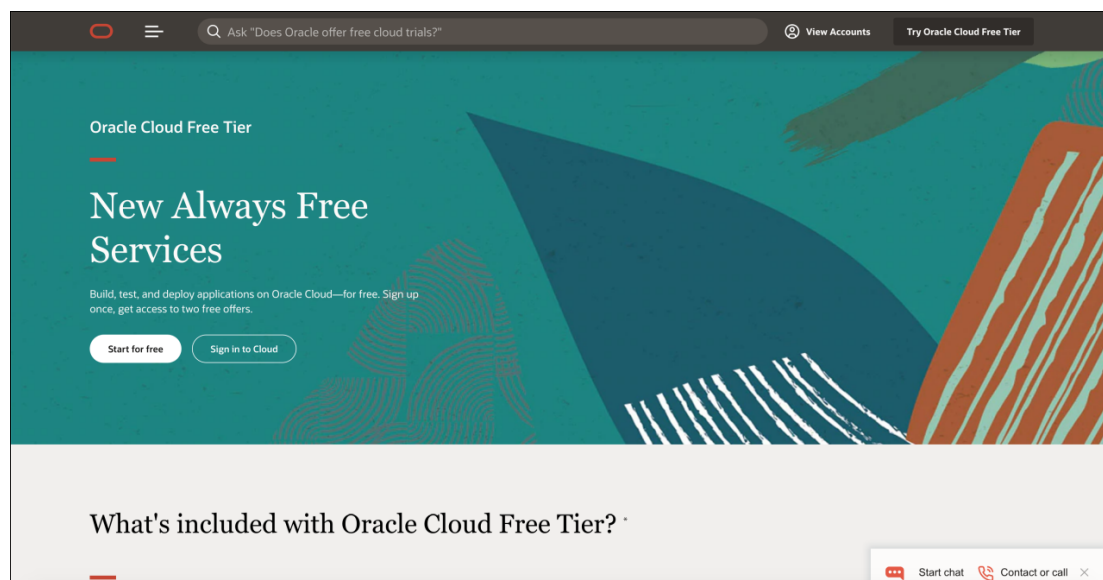
1. [Register for Oracle Cloud](#)
2. [Provision an Instance of Oracle Content Management](#)
3. [Add a content model, content, and a channel](#)
4. [Configure Oracle Content Management as a headless CMS](#)
5. [Issue your first request to Oracle Content Management](#)

This will get you from zero to productivity in no time.

Register for Oracle Cloud

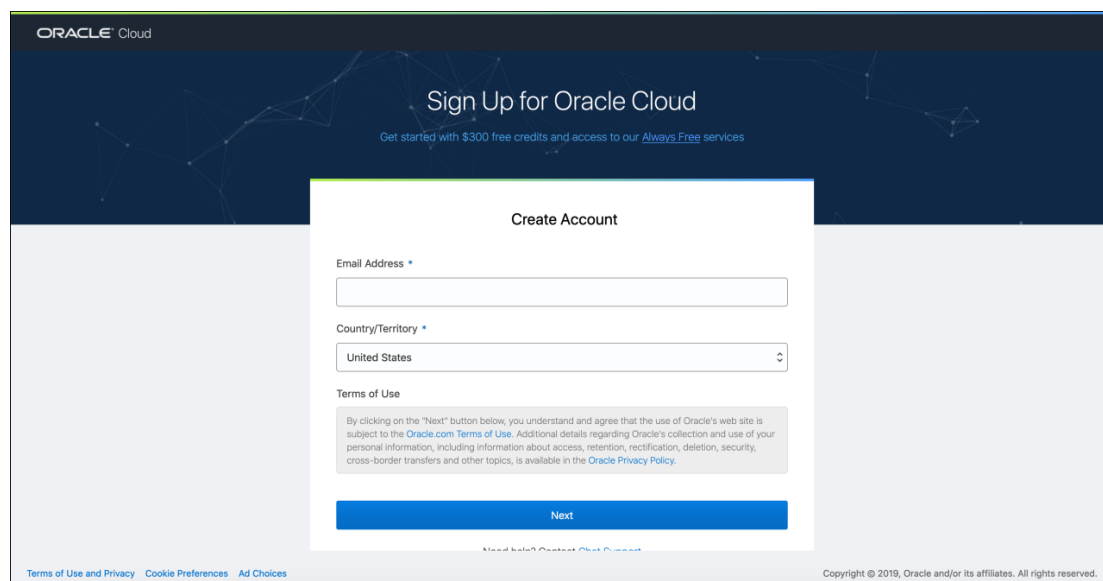
If you create an [Oracle Cloud account](#), you can access the 30-day trial that Oracle Content Management offers to all new users. This free trial provides thirty days of free access and USD 300 in free credits (known in Oracle parlance as Universal Credits).

If you already have an Oracle Cloud account, you can skip this step and proceed to [provision an instance of Oracle Content Management](#).

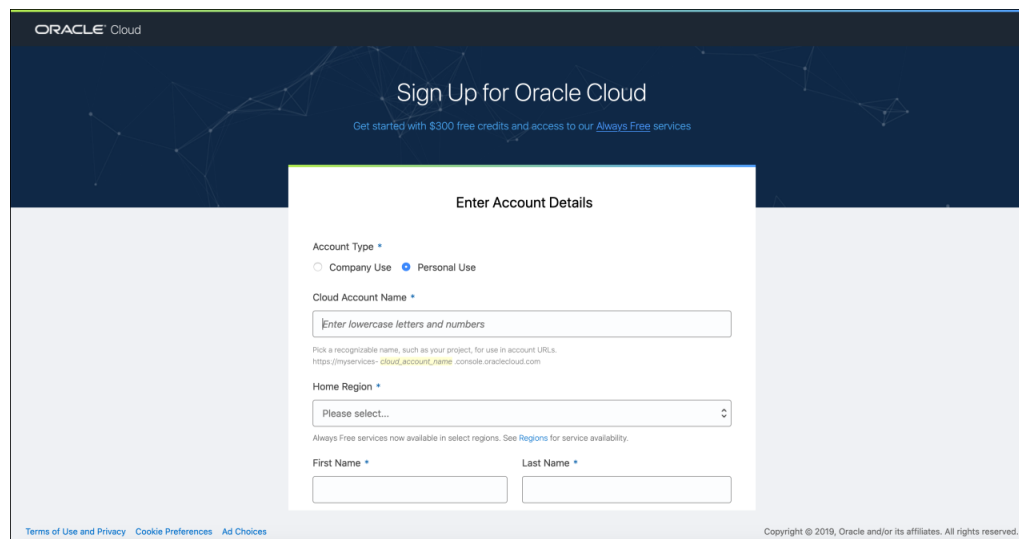


Select an Account Name and Home Region

First, navigate to [Oracle.com](#) and click **Try Oracle Cloud Free Tier**, then click **Start for free**. Enter your email address and select your location, and click **Next** to proceed.



On the following screen, provide an account name, which you'll use to access your Oracle Cloud account, and a home region, which you select from a list of Oracle data regions. You'll then be asked to provide contact information, including your mobile phone number for account verification.



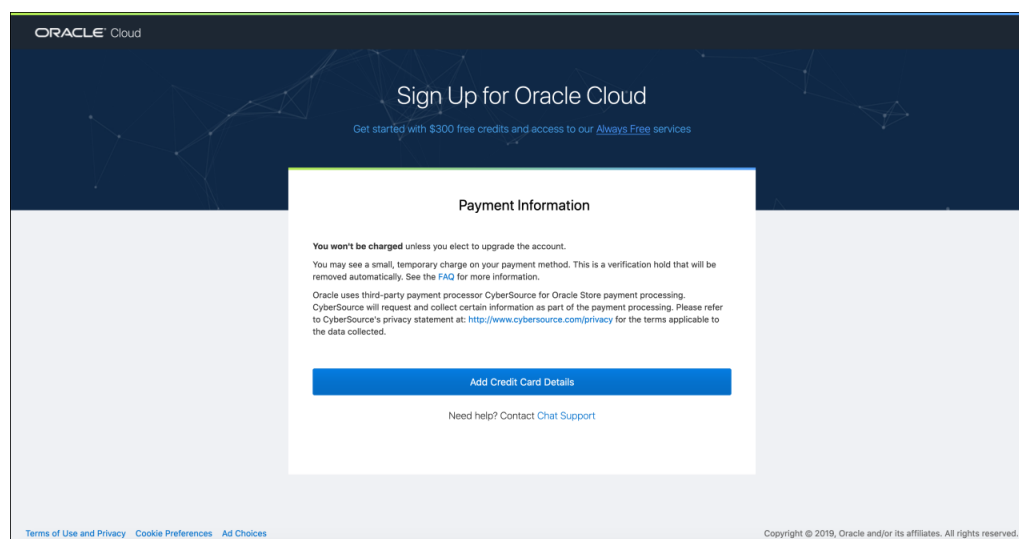
The screenshot shows the Oracle Cloud sign-up page. The main heading is "Sign Up for Oracle Cloud" with a sub-heading "Get started with \$300 free credits and access to our [Always Free](#) services". The form is titled "Enter Account Details" and contains the following fields:

- Account Type ***: Radio buttons for "Company Use" and "Personal Use" (selected).
- Cloud Account Name ***: A text input field with a placeholder "Enter lowercase letters and numbers". Below it, a note says: "Pick a recognizable name, such as your project, for use in account URLs. [https://myservices-cloud_account_name.console.oraclecloud.com](https://myservices-<u>cloud_account_name</u>.console.oraclecloud.com)".
- Home Region ***: A dropdown menu with "Please select..." and a downward arrow.
- Always Free services**: A note below the region dropdown: "Always Free services now available in select regions. See [Regions](#) for service availability."
- First Name *** and **Last Name ***: Two separate text input fields.

At the bottom of the page, there are links for "Terms of Use and Privacy", "Cookie Preferences", and "Ad Choices", and a copyright notice: "Copyright © 2019, Oracle and/or its affiliates. All rights reserved."

Provide Payment Information

Oracle Cloud requires trial users to provide payment information such as credit card details, but you won't be charged unless you choose to upgrade the account to a paid subscription.



The screenshot shows the Oracle Cloud sign-up page. The main heading is "Sign Up for Oracle Cloud" with a sub-heading "Get started with \$300 free credits and access to our [Always Free](#) services". The form is titled "Payment Information" and contains the following text:

You won't be charged unless you elect to upgrade the account.

You may see a small, temporary charge on your payment method. This is a verification hold that will be removed automatically. See the [FAQ](#) for more information.

Oracle uses third-party payment processor CyberSource for Oracle Store payment processing. CyberSource will request and collect certain information as part of the payment processing. Please refer to CyberSource's privacy statement at: <http://www.cybersource.com/privacy> for the terms applicable to the data collected.

Add Credit Card Details

Need help? [Contact Chat Support](#)

At the bottom of the page, there are links for "Terms of Use and Privacy", "Cookie Preferences", and "Ad Choices", and a copyright notice: "Copyright © 2019, Oracle and/or its affiliates. All rights reserved."

After providing payment information, you'll be asked to agree to the terms and conditions of the Oracle Cloud Services Agreement.

The screenshot shows the 'Payment Information' step of the Oracle Cloud sign-up process. The page has a dark blue header with the text 'Sign Up for Oracle Cloud' and 'Get started with \$300 free credits and access to our Always Free services'. The main content area is white and contains the following text:

Payment Information

You won't be charged unless you elect to upgrade the account.
You may see a small, temporary charge on your payment method. This is a verification hold that will be removed automatically. See the [FAQ](#) for more information.

Oracle uses third-party payment processor CyberSource for Oracle Store payment processing. CyberSource will request and collect certain information as part of the payment processing. Please refer to CyberSource's privacy statement at: <http://www.cybersource.com/privacy> for the terms applicable to the data collected.

Thank you for providing your credit card details.

Number * Expiration *

xxxxxxxxxxxx2008 12-2024

[Edit Payment Method](#)

By clicking **Complete**, I agree to the terms and conditions of the [Oracle Cloud Services Agreement V040119 for Oracle America, Inc.](#), (also available [here](#)) and this order including Service Description for Free Oracle Cloud Promotion Universal Credits - Part Number B88385.

Complete Sign-Up

Need help? [Contact Chat Support](#)

At the bottom, there are links for 'Terms of Use and Privacy', 'Cookie Preferences', and 'Ad Choices', and a copyright notice: 'Copyright © 2019, Oracle and/or its affiliates. All rights reserved.'

When you confirm to complete the sign-up, Oracle Cloud will proceed to create your account. This may take up to fifteen minutes, and you'll receive a confirmation at the email address you provided during the sign-up process.

The screenshot shows the confirmation page after completing the sign-up. The page has a dark blue header with the text 'Sign Up for Oracle Cloud' and 'Get started with \$300 free credits and access to our Always Free services'. The main content area is white and contains the following text:

Thank you for signing up for Oracle Cloud!

We are creating your account, which may take up to 15 minutes. Check your email for further instructions.

While you wait, you can read about [Oracle Cloud Infrastructure](#).

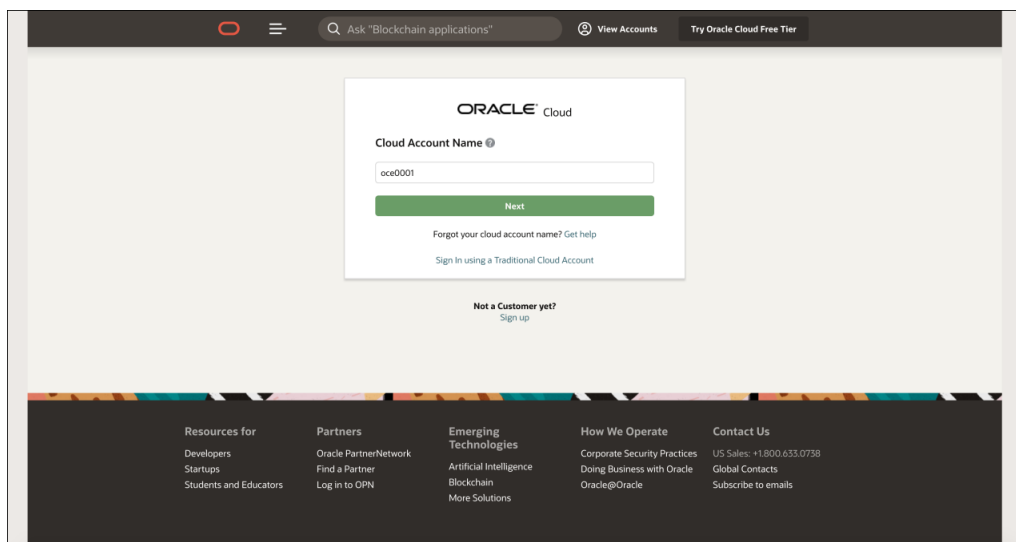
Need help? [Contact Chat Support](#)

At the bottom, there are links for 'Terms of Use and Privacy', 'Cookie Preferences', and 'Ad Choices', and a copyright notice: 'Copyright © 2019, Oracle and/or its affiliates. All rights reserved.'

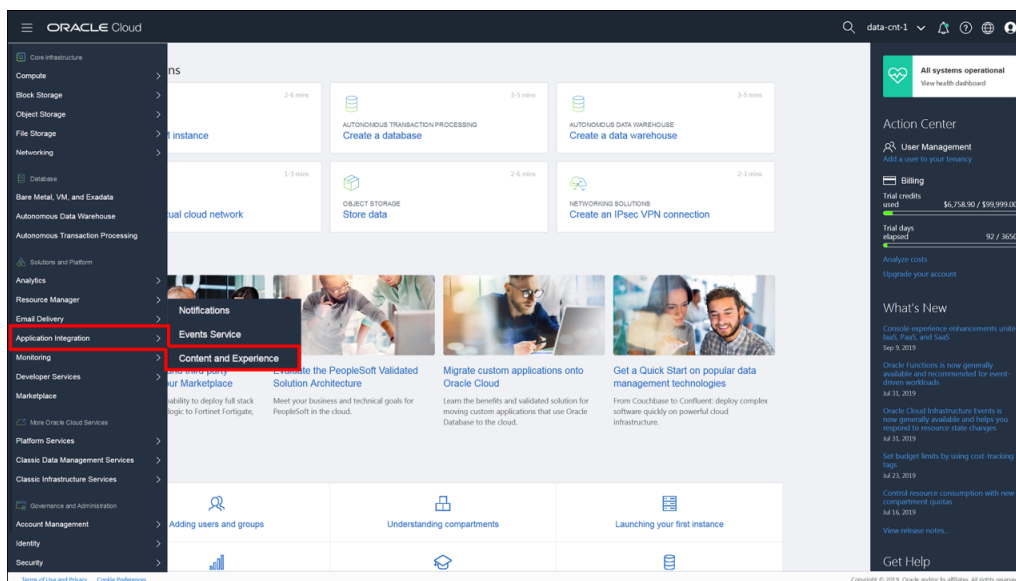
Provision an Instance of Oracle Content Management

Next, you need to create an instance of Oracle Content Management. If you already have an Oracle Content Management instance, you can skip this step and proceed to [add a content model, some content, and a channel](#).

Log in to your Oracle Cloud account. Note that your cloud account name is distinct from your user name and password. You can think of the cloud account name as representing your entire organization and your Oracle Cloud user name and password as representing you as an individual user. After logging in, you'll see your Infrastructure Dashboard.

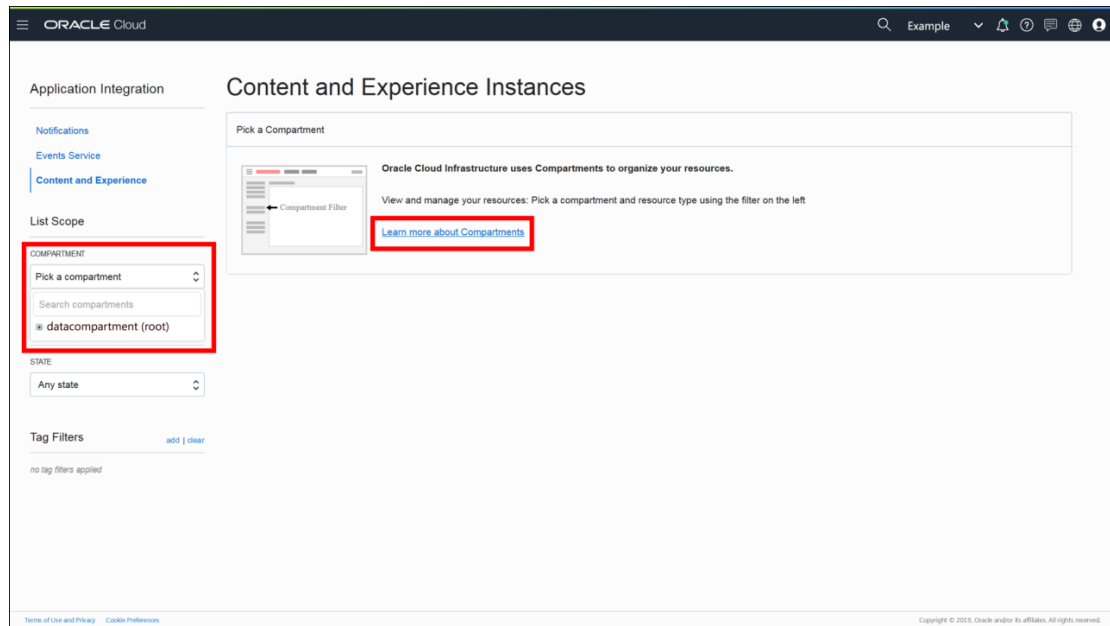


Open the hamburger menu in the upper left-hand corner (the three horizontal lines) and, under Solutions and Platform, hover over **Application Integration** and choose **Content Management**.



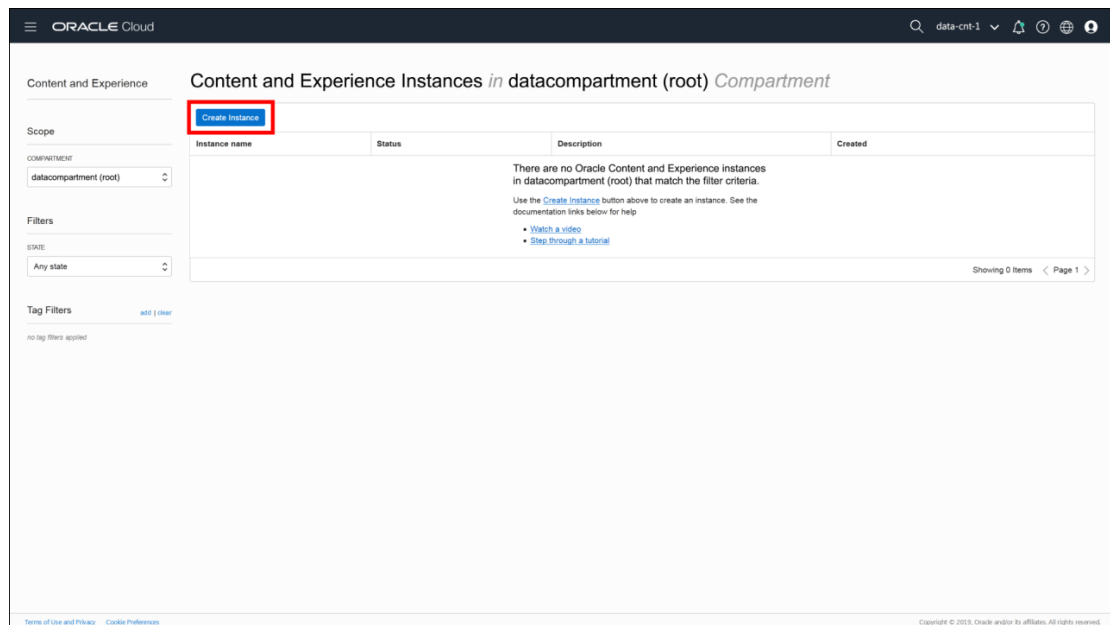
Choose a Storage Compartment

In Oracle Cloud, compartments are used to organize cloud resources for a variety of purposes, including isolation, access, and billing. Due to security reasons, it is highly recommended to create and use a new storage compartment rather than using the existing root storage compartment.

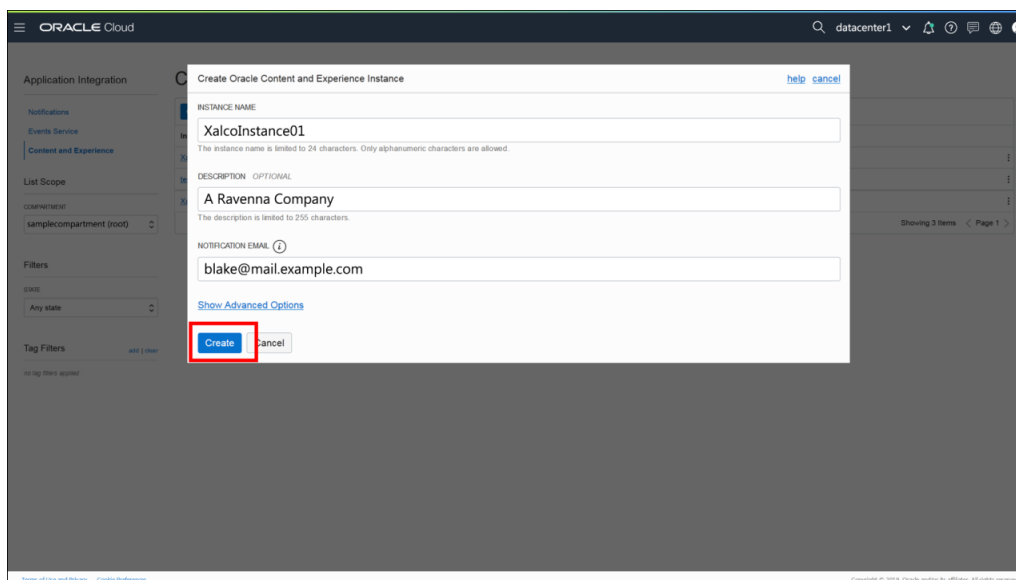


Create Your Oracle Content Management Instance

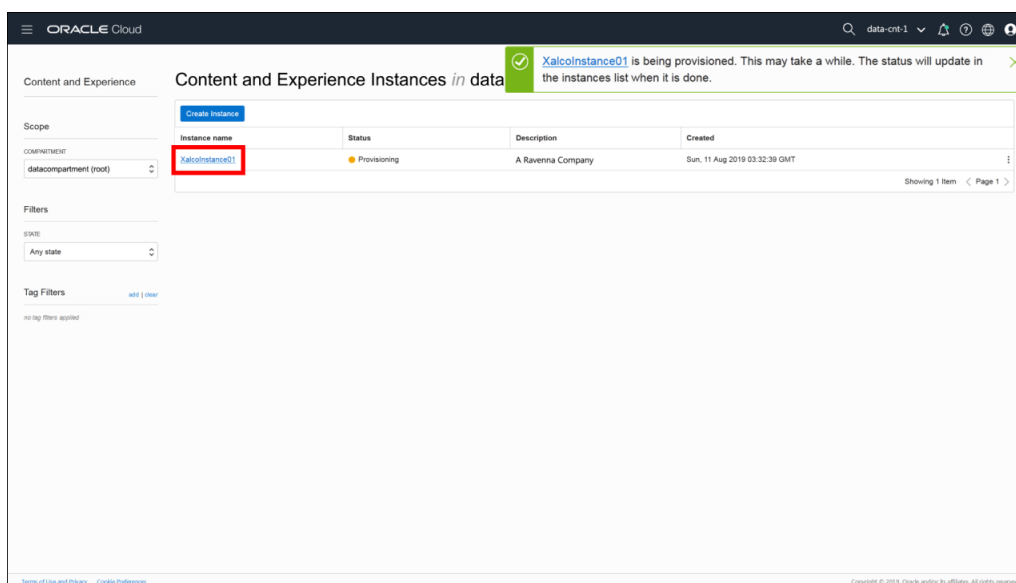
Now, click **Create Instance** to create your new service instance of Oracle Content Management.



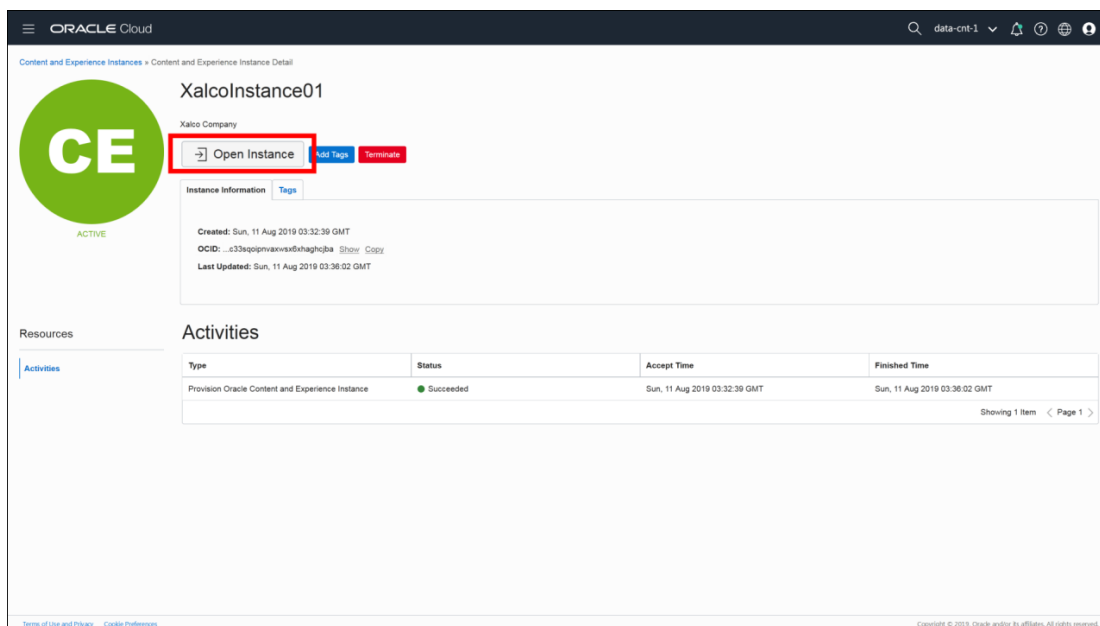
Provide a name for the instance and an optional description, and then click **Create**.



It will take some time for the instance to be created, but the instance list page refreshes automatically to keep you updated on the status of the creation process. Once provisioning has completed, you can click the instance name to access its details.



Finally, once the instance has been created, you can click **Open Instance** to navigate to the web interface of the provisioned instance. Now we're ready to do some content modeling, add some content, and publish it to a channel!



Add a Content Model, Some Content, and a Channel

By default, every new instance of Oracle Content Management comes empty, with an empty content model and no created content.

To use Oracle Content Management as a headless CMS, we need to supply a content model, create content, and publish it to a channel.

If you already have content published to a channel, you can skip this step and proceed to [configure Oracle Content Management as a headless CMS](#).

Open your instance. Note that in previous screenshots the Oracle Content Management instance we created is named **XalcoInstance01**.

Create a Content Type

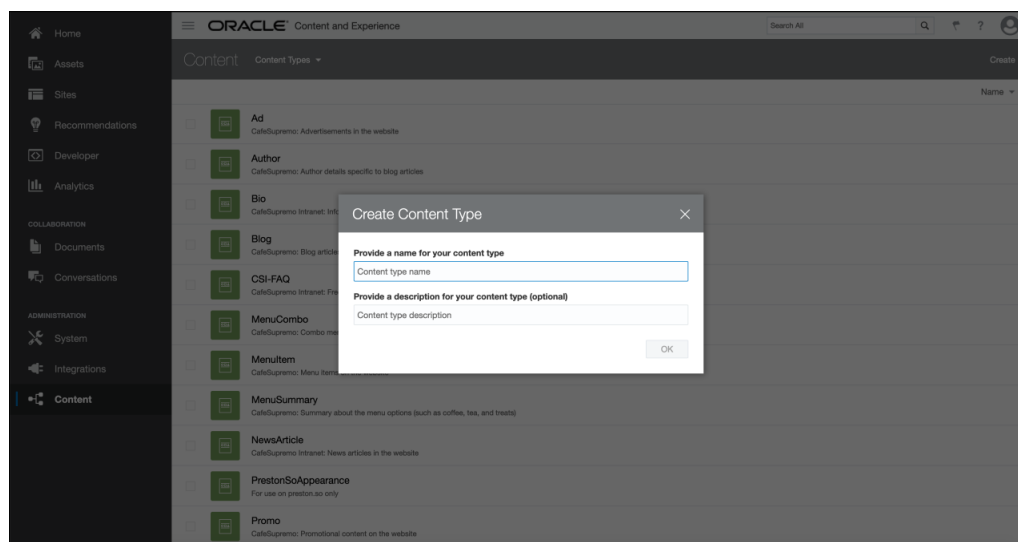
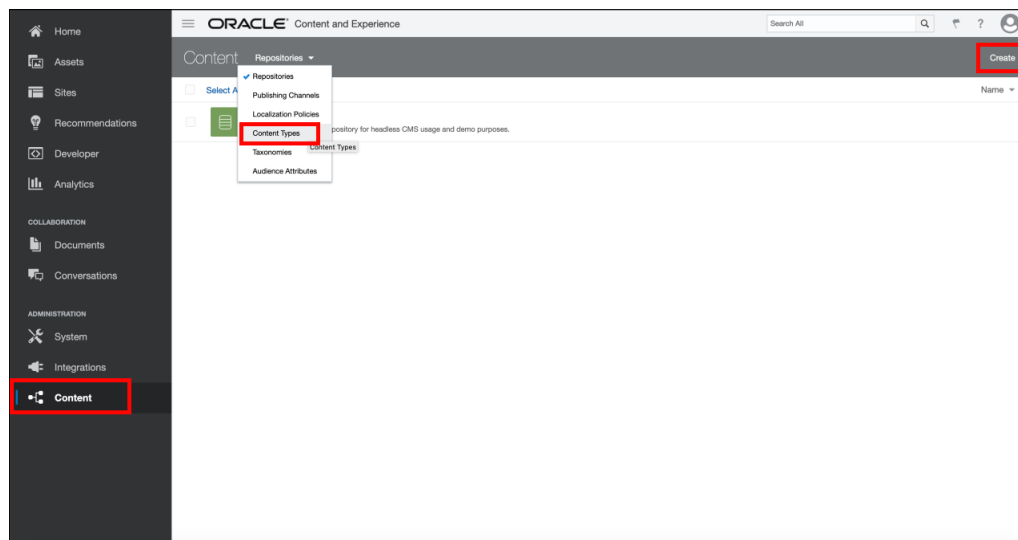
You can create a content model that contains particular [content types](#) and associated fields. Content authors choose a content type every time they want to create a new content item. Responses that developers ingest in their applications also adhere to the created content schema.

First, sign in to the Oracle Content Management web interface as an administrator. Click **Content** in the left sidebar and then choose **Content Types** from the selection list in the header.

Note:

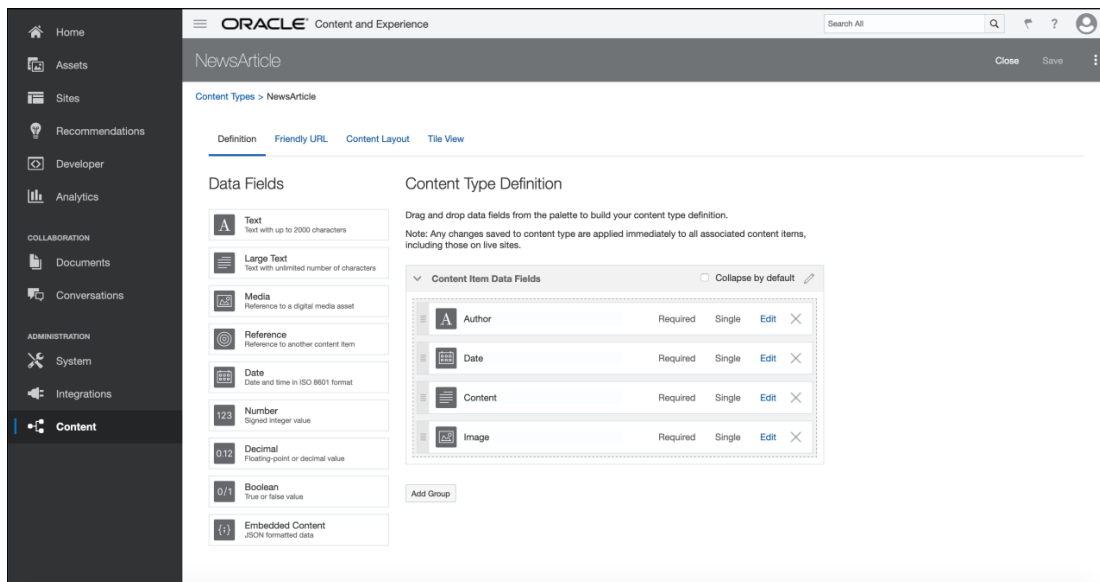
If you don't see the **Content** option in the left sidebar, your user login doesn't have the required administrative privileges.

Next, click **Create** in the upper right corner. The **Create Content Type** dialog opens, where you can provide a name for your content type and an optional description.

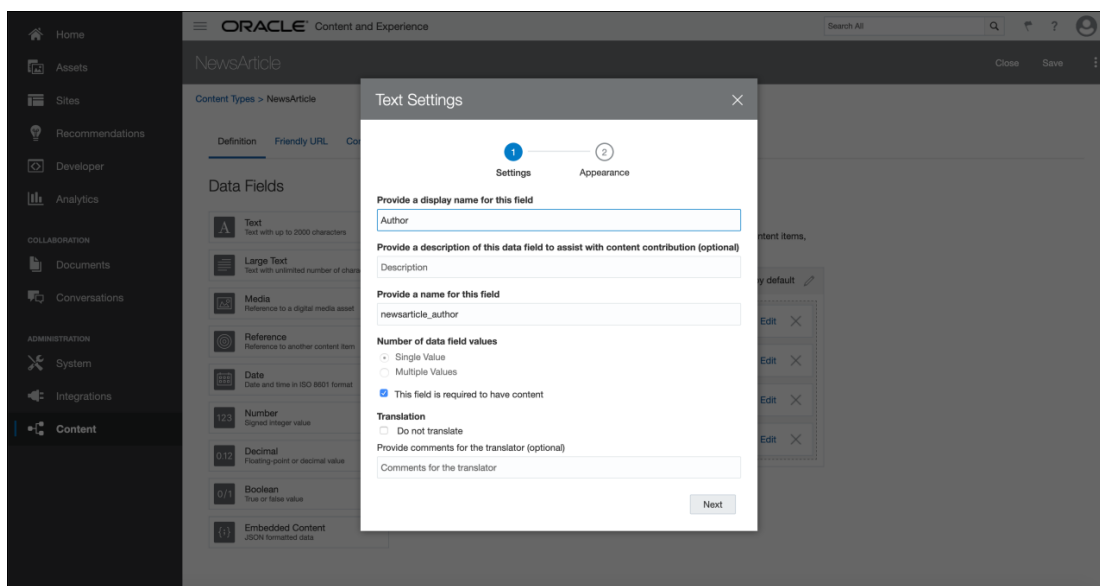


For the purposes of getting started quickly, we need only a simple content type. Let's call this content type **NewsArticle**. Give the **NewsArticle** content type four fields (each reflecting a distinct field type), with the following values filled in on the **Settings** step of the **Text Settings** dialog (we'll skip **Appearance** for now):

- The **Author** field is a required single-value Text field having the display name *Author* and the machine name *newsarticle_author*.
- The **Date** field is a required single-value Date field having the display name *Date* and the machine name *newsarticle_date*.
- The **Content** field is a required single-value Large Text field having the display name *Content* and the machine name *newsarticle_content*.
- The **Image** field is a required single-value Image field having the display name *Image* and the machine name *newsarticle_image*.



The following screenshot shows our settings for the **Author** field as an example. Depending on the field type, the structure of the form may be different.



Now that we've created our content type, we can turn to creating a publishing channel and an [asset](#) repository, which will allow us to create content items.

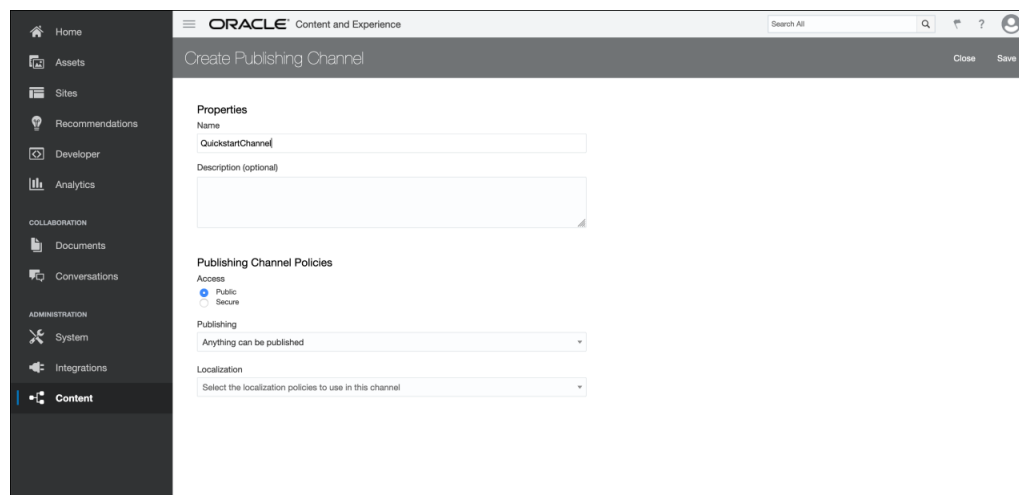
Create a Publishing Channel

In Oracle Content Management, a publishing channel represents ranges of digital experiences that need to be served content by the headless CMS. You can also think of channels as containers for configuration and policies related to publishing content.

Though channels can be public or secure depending on requirements, we'll focus only on public channels here.

Suppose you have a React application that needs to consume content from Oracle Content Management. By creating a channel with a name like JavaScript, you can publish content to the JavaScript channel and make it available to any JavaScript application that may need to consume that content, including your React application. Instead of using channels to represent sets of digital experiences, you can also designate channels for individual experiences (for example, channels named **AugmentedReality** or **DigitalSignage**).

Click **Content** in the left sidebar and choose **Publishing Channels** from the selection list in the header. You'll see an empty list without any channels created. In the upper right corner, click **Create** to create a new channel. Give the channel the name **QuickstartChannel** for the purposes of this quick start tutorial and keep the access public. Click **Save** to create the channel.



Create an Asset Repository

In Oracle Content Management, an asset repository is essentially a large *bucket* that contains all the content items (known in Oracle Content Management as assets) an organization needs to work with.

Click **Content** in the left sidebar and choose **Repositories** from the selection list in the header. You'll see an empty list without any asset repositories created. In the upper right corner, click **Create** to create a new asset repository. Give the asset repository the name **QuickstartRepo** for the purposes of this quick start tutorial.

Specify the **NewsArticle** content type under **Content Types** to indicate to Oracle Content Management that assets of the type **NewsArticle** can be created in the **QuickstartRepo** asset repository.

Finally, specify the **QuickstartChannel** channel under **Publishing Channels** to indicate to Oracle Content Management that content in the **QuickstartRepo** repository can be published to the **QuickstartChannel** channel.

For now, you can leave everything else untouched and click **Save** to create the asset repository.

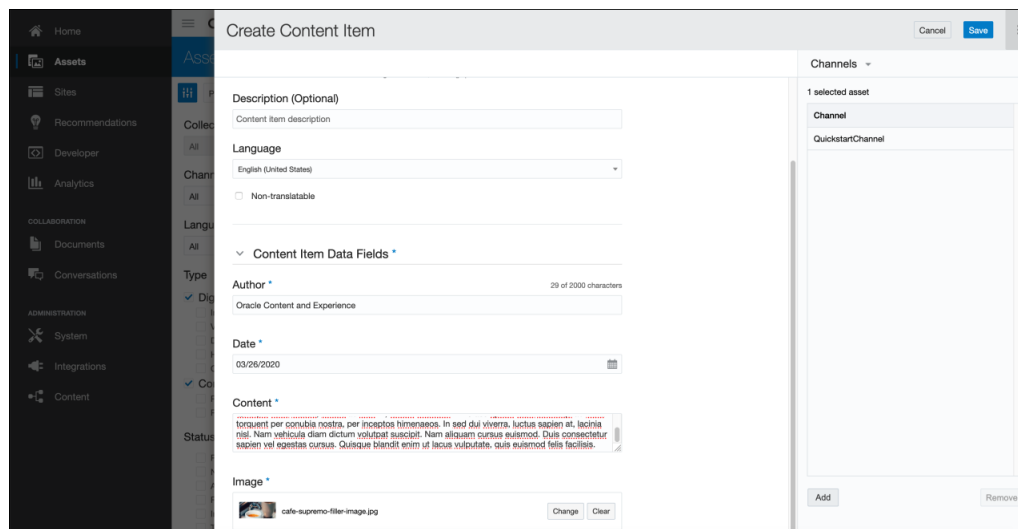
Create a Content Item (Asset)

In Oracle Content Management, an **asset** is the atomic unit of managed content and adheres to a particular content type. Each content item is an asset. For developers coming from other CMS ecosystems, assets in Oracle Content Management are most similar to what many CMSs call entities or records.

To create your first asset, click **Assets** in the left sidebar and click **Create** in the upper right corner. You'll see a selection list of available content types, including **NewsArticle**. Select this content type to continue.



In the **Create Content Item** form, enter "My First News Article" as the name under **Content Item Properties** and insert filler content under **Content Item Data Fields**, which contains the fields we defined for the **NewsArticle** content type. Then, in the **Channels** sidebar, click **Add** and select **QuickstartChannel** to identify it as the channel to which we'll be publishing.

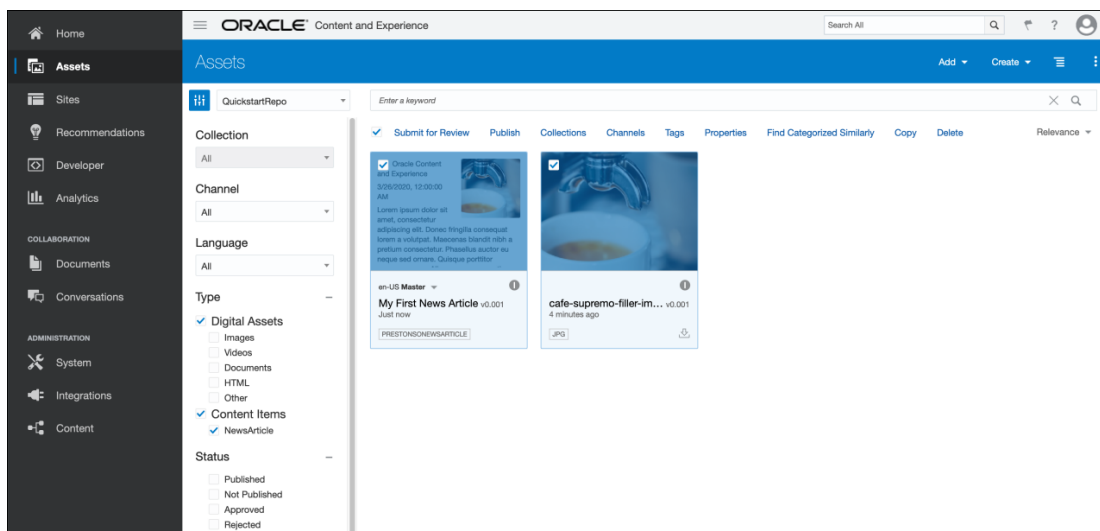


The image you choose to add will become an asset as well.

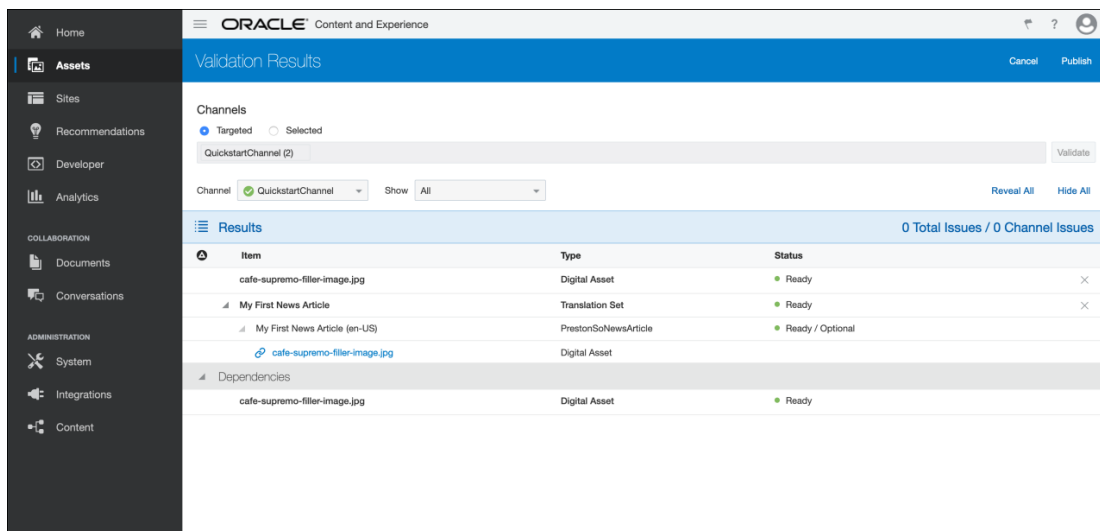
Publish Content Assets to a Channel

Now, to make the content we just created available to the **QuickstartChannel** channel, we need to **publish** both newly created assets to that channel, including the **NewsArticle** content item and the image associated with it.

Navigate to **Assets** in the left sidebar, where you'll find both of these assets available as unpublished assets. Select them both and click **Publish** in the action links that appear above the selected assets.



On the following **Validation Results** screen, verify that the assets you want to publish to the **QuickstartChannel** channel are represented. Click **Publish** in the upper right corner to verify. This will publish both assets to the channel, now making them available for public consumption.



Configure Oracle Content Management As a Headless CMS

To enable Oracle Content Management as a headless CMS, we need to configure cross-origin resource sharing (CORS) for security reasons and acquire an API access token for our publishing channel.

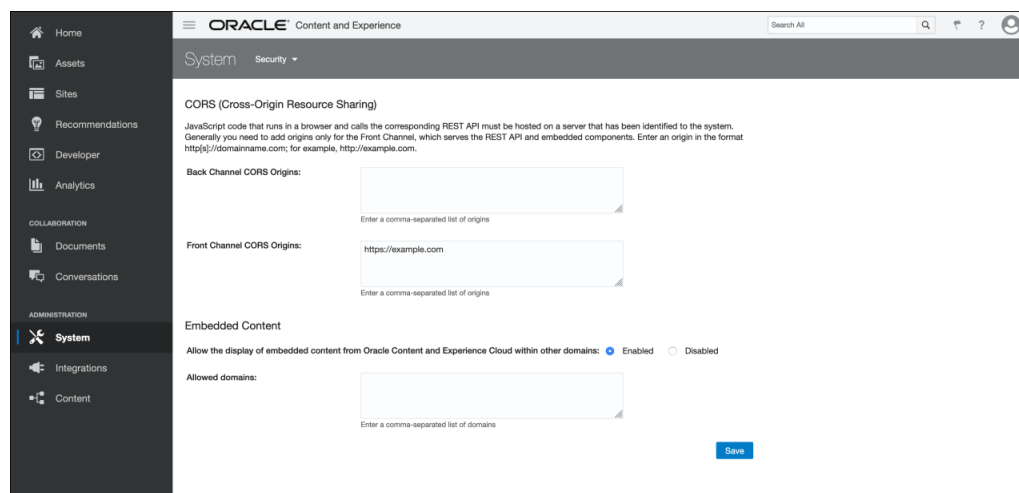
If you already have CORS configured and an API access token at hand, you can skip this step and proceed to configure Oracle Content Management as a headless CMS.

Note also that for the purposes of providing a base URL of your CMS backend, the instance URL is the domain name in your URL bar that ends with `oraclecloud.com`.

Configure Cross-Origin Resource Sharing (CORS)

Cross-origin resource sharing (CORS) prevents unauthorized requests from overloading your API and causing distributed denial-of-service (DDoS) attacks. To configure CORS, click **System** in the left sidebar and then choose **Security** in the selection list in the header.

If you already have domains ready for your consumer applications, you can insert them into the Front Channel CORS Origins field, which allows access to content through Oracle Content Management's REST APIs and embedded components.



As you can see in the preceding screenshot, we have configured CORS to allow requests originating from the `https://example.com` domain. To include additional domains that are allowed access to the Oracle Content Management REST APIs, insert a comma-separated list of domains (no quotation marks or other delimiters necessary). Any domains specified here will be able to issue requests successfully to the APIs provided by Oracle Content Management.

Acquire and Refresh API Access Tokens

As we saw earlier in this quick start tutorial, assets in Oracle Content Management must be published to a channel for that content to be available to other consumers, such as mobile or JavaScript applications.

Since channels can have differentiated sets of published assets, each channel provides its own unique API access token, which must be included in every request to target an individual channel.

Click **Content** in the left sidebar and choose **Publishing Channels** in the selection list in the header. Select the **QuickstartChannel** channel from the list of channels. Under the API Information header, you'll see two fields whose values can be copied to the clipboard: **Channel ID** and **Channel Token**. To refresh your channel's API access token, click **Refresh** to the right of the channel token.

The screenshot shows the 'Edit Publishing Channel' page in the Oracle Content and Experience interface. The left sidebar contains navigation options like Home, Assets, Sites, Recommendations, Developer, Analytics, and Content. The main content area is titled 'Edit Publishing Channel' and shows the configuration for a channel named 'QuickstartChannel'. The configuration is divided into three sections: Properties, Publishing Channel Policies, and API Information. The Properties section includes fields for Name (QuickstartChannel) and Description (optional). The Publishing Channel Policies section includes radio buttons for Access (Public and Secure), a dropdown for Publishing (Anything can be published), and a dropdown for Localization. The API Information section displays the Channel ID (RCHANNEL080C73782410460ABA4FD146E2638CFD) and the Channel Token (99e76da7c5d24c11aa806acc58a46b42a).



Note:

You can also acquire your channel API access token programmatically by issuing a request to the [REST API for Content Management](#) provided by Oracle Content Management.

Issue Your First Request to Oracle Content Management

Our next step is to issue our first request to Oracle Content Management to determine that our APIs are configured properly.

Recall that the instance URL is simply the domain at which your Oracle Content Management instance resides, adhering to the following format, where `{instance_name}` is the name of the instance and `{cloud_account_name}` is the cloud account name tied to your Oracle Cloud account:

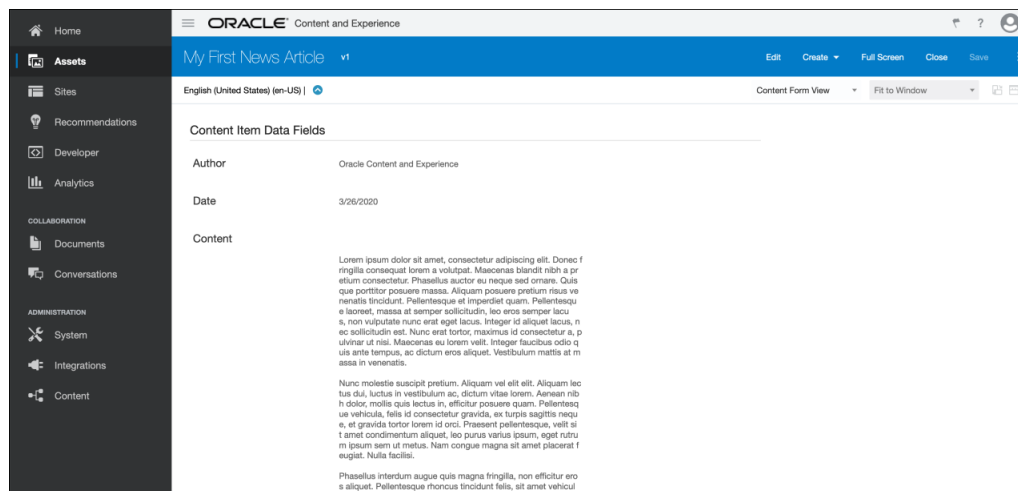
```
https://{instance_name}-{cloud_account_name}.cec.ocp.oraclecloud.com
```

Copy your instance URL and keep it handy, as this is the domain against which you'll be issuing every API request from your consumer applications. In the coming tests, we'll be retrieving the **NewsArticle** asset we created earlier and published to the **QuickstartChannel** channel.

If you want to retrieve an individual asset with a GET request, you'll need to supply the identifier of that asset in your request. To acquire the identifier of your **NewsArticle** asset, click **Assets** in the left sidebar and select your **NewsArticle** asset, as seen in the following screenshot. In the URL, you'll see a path adhering to the following format, where `{asset_id}` is the asset identifier:

```
https://{instance_name}-{cloud_account_name}.cec.ocp.oraclecloud.com/documents/assets/view/{asset_id}
```

Copy this `{asset_id}` value and keep it handy for our coming GET requests.



In the next three sections, we'll use three different approaches to issue a request to retrieve a published item: Postman, cURL, and a typical XMLHttpRequest. The resource path takes the following format, with the added `channelToken` query parameter, whose value is `{channel_token}`, the API access token we copied earlier:

```
https://{instance_name}-
{cloud_account_name}.cec.ocp.oraclecloud.com/content/published/api/
v1.1/items
/{asset_id}?channelToken={channel_token}
```

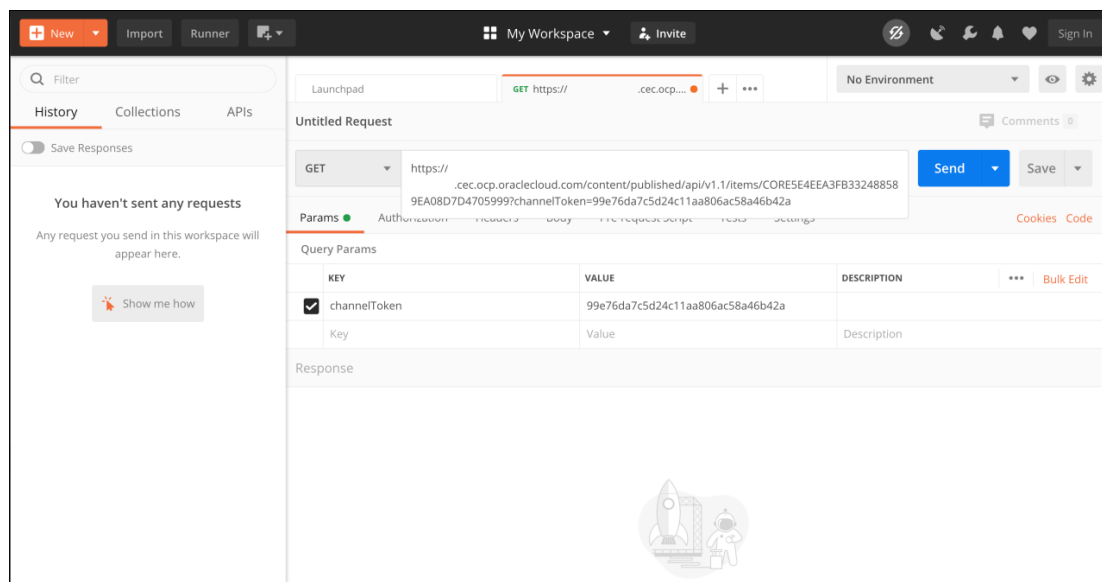
Without the `channelToken` query parameter included, you'll receive a "403 Forbidden" error.

Retrieve Content Through Postman

One of the most ubiquitous developer tools for testing API requests is Postman, a free API client and desktop application. Postman provides a convenient user interface to form your request, including request headers and request body.

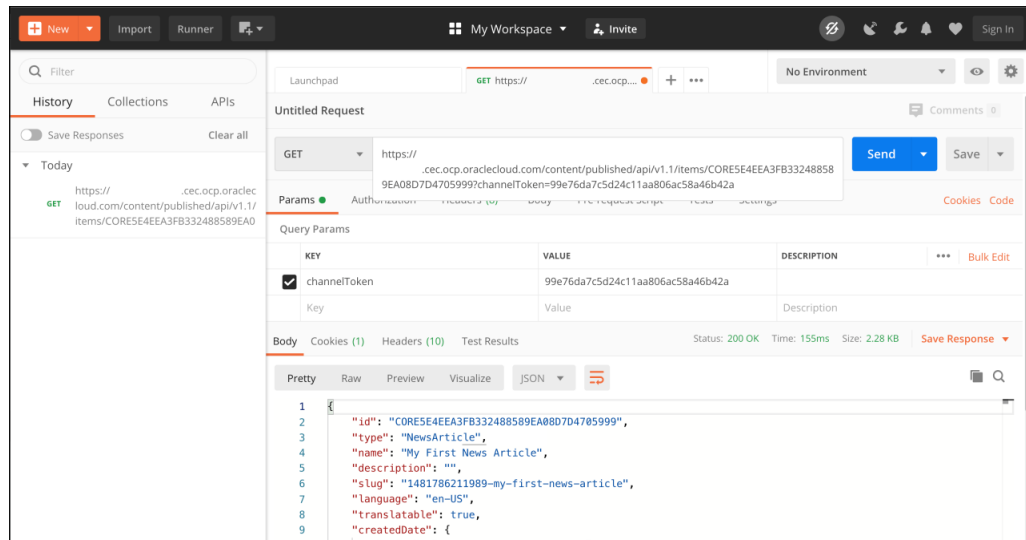
To test an API call from Postman, insert the following into the path for a new Postman GET request, supplying the `channelToken` query parameter under the **Params** tab:

```
https://{instance_name}-
{cloud_account_name}.cec.ocp.oraclecloud.com/content/published/api/
v1.1/items
/{asset_id}?channelToken={channel_token}
```



Upon issuing the GET request, you'll see a response with the code "200 OK," which begins with the following JSON:

```
{
  "id": "CORE5E4EEA3FB332488589EA08D7D4705999",
  "type": "NewsArticle",
  "name": "My First News Article",
  "description": "",
  "slug": "1481786211989-my-first-news-article",
  "language": "en-US",
  "translatable": true,
  "createdDate": {
    "value": "2020-03-26T19:15:55.710Z",
    "timezone": "UTC"
  },
  ...
}
```



Note that in the preceding two screenshots, the instance name and cloud account name have been obfuscated and should be replaced with your own details.

Retrieve Content Through cURL

To retrieve our NewsArticle asset through the command line, we can use the command-line tool cURL.

The [Oracle Content Management REST API documentation](#) contains information about using cURL to access the [REST API for Content Management](#), with useful initial steps for those unfamiliar with cURL.

To test an API call from cURL, issue the following command, where {instance_name} and {cloud_account_name} are your instance details:

```
curl -i -X GET https://{instance_name}-
{cloud_account_name}.cec.ocp.oraclecloud.com/content/published/api/
v1.1/items
/CORE54EEA3FB332488589EA08D7D4705999?
channelToken=99e76da7c5d24c11aa806ac5
8a46b42a
```

You'll receive a response that returns the JSON from the previous section, as seen in the following screenshot (note that the instance name and cloud account name have been obfuscated).



Retrieve Content Through an XMLHttpRequest

Finally, we can also retrieve our NewsArticle asset from any JavaScript application or in the browser by employing the XMLHttpRequest API.

To test an API call using XMLHttpRequest, issue a request by constructing an XMLHttpRequest and invoking the `send()` method as follows, where `{instance_name}` and `{cloud_account_name}` are your instance details:

```
var req = new XMLHttpRequest();
req.open("GET", "https://{instance_name}-
{cloud_account_name}.cec.ocp.oraclecloud.com/content/published/api/v1.1/items
/CORE5E4EEA3FB332488589EA08D7D4705999?channelToken=99e76da7c5d24c11aa806ac58a
46b42a");
req.send();
```

You'll receive a response that returns the JSON from the previous section. Note that if you're issuing a variety of GET requests with diverse query parameters, you may want to employ a utility function to handle arbitrary query parameters in lieu of inserting them into your invocation of the `open()` method directly.

Next Steps

To learn more about using Oracle Content Management as a headless content management system, see the documentation and sample websites.

You can browse the [Headless CMS section](#) of the [Oracle Content Management documentation](#), where you can find information about the [content delivery and content management APIs](#) as well as [software development kits](#) that can aid your implementation.

In addition, you can download sample websites and learn about the open-source ecosystem surrounding Oracle Content Management on the [downloads page](#).

2

Oracle Content Management REST APIs for Headless Development

REST application programming interfaces (APIs) are available in Oracle Content Management for content delivery and for management of content, conversations, documents, and users and groups.

Which REST API you use depends on what you want to do:

- If you want to fetch assets that have been published to a channel or information about the assets, use the [REST API for Content Delivery](#).
- If you want to manage assets in Oracle Content Management, use the [REST API for Content Management](#). Assets include content items as well as digital assets and their renditions.

In addition to these content APIs, Oracle Content Management offers a number of other REST APIs that let you integrate or extend Oracle Content Management functionality:

- [REST API for Activity Log](#)
- [REST API for Content Capture](#)
- [REST API for Content Preview](#)
- [REST API for Conversations](#)
- [REST API for Documents](#)
- [REST API for Self-Management](#)
- [REST API for Site Management](#)
- [REST API for Users and Groups](#)
- [REST API for Webhooks Management](#)

REST API for Content Delivery

You can use the Oracle Cloud REST API for Content Delivery to fetch assets or information about them from channels in an asset repository. Assets include content items as well as digital assets and their renditions.

The [REST API for Content Delivery](#) has several categories of endpoints, which the following table describes.

Category	Description
AutoSuggestions	Use the AutoSuggestions resource to suggest item keywords for auto-completion of a default search.
Item	Use the Item resource to get published items, previews of items, item metadata, or taxonomies of items.
Item Variations	Use the Item Variations resource to get item variations, a content item for item variations, and item variations by variation type.

Category	Description
Items	Use the Items resource to search published items or get the metadata catalog of published items.
Items by Slug	Use the Items by Slug resource to manage items by slug and to provide details about the metadata catalog preview, taxonomies, published information, and variations of an item.
Provider Tokens	Use the Provider Tokens resource to generate a provider token.
Published Item	Use the Published Item resource to get the metadata catalog preview of an item.
Recommendations	Use the Recommendations resource to access published recommendation results.
Renditions	Use the Renditions resource to get a digital assets file and metadata catalog, to get metadata for digital assets or renditions, or to get information about published assets or renditions.
Taxonomies	Use the Taxonomies resource to get the metadata of a category, published taxonomies, or published categories, to list all taxonomies, to read a published category or taxonomy, or to search published categories.
Version Catalog	Use the Version Catalog resource to get information about APIs, API versions, or API metadata.

REST API for Content Management

You can use the Oracle Cloud REST API for Content Management to manage assets that have been published to a channel in an Oracle Content Management asset repository. Assets include content items as well as digital assets and their renditions.

The [REST API for Content Management](#) has several categories of endpoints, which the following table describes.

Category	Description
AutoSuggestions	Use the AutoSuggestions resource to suggest item keywords for auto-completion of a default search.
Channel Secret	Use the Channel Secret resource to generate, refresh, or delete a channel secret.
Channels	Use the Channels resource to create, delete, read, or update a channel, to list all channels, or to list all permissions on a channel.
Collections	Use the Collections resource to create, delete, read, or update a collection, to list all collections in a repository, or to list all permissions on a collection.
Connectors	Use the Connectors resource to list all connectors.
Digital Item Renditions	Use the Digital Item Renditions resource to get a rendition of a digital item or a digital item native file with or without a file name.
File Extensions	Use the File Extensions resource to list file extensions or to read a file extension.
Item Revisions	Use the Item Revisions resource to list item revisions, list item revisions by slug, read an item revision, or read an item revision by slug.
Item Variations	Use the Item Variations resource to list all item variations of a variation type, read an item or items by variation type, or update the master item of an item variations set.

Category	Description
Items	Use the Items resource to: <ul style="list-style-type: none"> • Create or delete a content item or digital item • Generate taxonomy suggestions for content items on demand • List all suggested taxonomies and categories or all taxonomies and categories of an item • Get the workflow instance details for an item • List the time zones, channels, collections, published channels, relationships, tags, or variations of an item • Read the lock, publish, or version information of an item • Read the on-demand taxonomy suggestion and operations status or the workflows of an item • Submit an item to workflow, take action on a workflow task, update a digital item with a new file, update an item, or update the lock status of an item
Items Bulk Operations	Use the Items Bulk Operations resource to perform bulk items operations, read item operations or status, or publish item IDs.
Items by Slug	Use the Items by Slug resource to manage items by slug: <ul style="list-style-type: none"> • List all item variations of a variation type • List an item's channels, collections, lock information, permissions, publish information, published channels, relationships, tags, taxonomies, variations, version information, or workflow information • Read an item or an item variation of a variation type value by slug • Read the master of an item variation set by slug
Items Search	Use the Items Search resource to manage items search queries, get the job status for similar items, query items, or query similar items.
Language Codes	Use the Language Codes resource to create a custom language code, delete a language code, list all valid language codes, read a language code, or update a language code.
Languages	Use the Languages resource to list the names of all known language codes.
Localization Policies	Use the Localization Policies resource to create, delete, read, or update a localization policy or to list all localization policies.
Taxonomies	Use the Taxonomies resource to manage your content: <ul style="list-style-type: none"> • Create, update, or delete a taxonomy • Get, create, update, copy, or delete a category in a taxonomy • Create, copy, update, promote, read, publish, unpublish, or delete a taxonomy • List all taxonomies or list all categories in a taxonomy • Copy, create, or delete a category • Read a taxonomy, a category, or the copy category, draft creation, the promote status • Create a new draft version • Read the promote, publish, or unpublish job status • Search categories • Update a category's properties, including moving it in the tree <p>After taxonomies are promoted, they can be assigned to repositories. After taxonomies are assigned to repositories, users can apply categories to assets.</p>
OAuth Tokens	Use the OAuth Tokens resource to generate an OAuth token.
Permission Operations	Use the Permission Operations resource to perform permission operations on a resource or to read permission operations status.

Category	Description
Provider Tokens	Use the Provider Tokens resource to generate a provider token for an asset for a specific version.
Recommendations	Use the Recommendations resource to manage recommendations: <ul style="list-style-type: none">• Create, delete, list, update, read, and publish recommendations• Read a recommendation's published and unpublished item IDs• Approve or reject a Recommendation• Create, delete, list, read, and update audience attributes
Repositories	Use the Repositories resource to create, delete, read, or update a repository, to list all permissions on a repository, or to list all repositories.
Tokens	Use the Tokens resource to read a a Cross-Site Request Forgery (CSRF) valid token.
Types	Use the Types resource to create, delete, read, or update a type or to list all types, all data types, or all permissions on a type.
Workflow Roles	Use the Workflow Roles resource to add or remove members of a process role, get members or details of a process role, or list the roles of all registered workflows.
Workflow Tasks	Use the Workflow Tasks resource to list workflow tasks assigned to the current user, or to read a workflow task.
Workflows	Use the Workflows resource to reregister or deregister a workflow, list all workflows, list all permissions on a workflow, read a workflow, or update a workflow.

3

Oracle Content Management SDKs

Oracle Content Management provides software development kits (SDKs) that help you integrate Oracle Content Management functionality and simplify your application development:

- [Content SDK for JavaScript](#)
- [Content SDK for Java](#)
- [Content SDK for Swift](#)
- [Sites SDK](#)
- [Translation Connector SDK](#)

Content SDK for JavaScript

The Content SDK for Oracle Content Management is a light-weight JavaScript wrapper that interacts with the Content REST APIs.

This read-only SDK retrieves structured content, digital assets, and content layouts that are managed in Oracle Content Management. The SDK can be used in web browsers or NodeJS projects.

The Content SDK for JavaScript consists of three main modules:

- **ContentSDK**: The main entry-point object. The ContentSDK object lets you create client objects to access content based on your requirements.
- **ContentDeliveryClient** : A client object that is set up to access published content items and digital assets.
- **ContentPreviewClient** : A client object that is set up to access content types, draft content items, and draft digital assets.

The Content SDK for JavaScript is available as an Oracle open-source project on [GitHub](#).

The reference guide can be found here, [Content SDK for JavaScript](#).

Mobile SDKs

For mobile application development, consider using the SDKs below:

- [Content SDK for Java](#)
- [Content SDK for Swift](#)

Content SDK for Java

Oracle Content Management provides a Content SDK for Java/Android. The read-only SDK is a package of libraries for retrieving published content items, digital assets, and content layouts that are managed in Oracle Content Management.

The SDK is a light-weight Android binding that interacts with the [REST API for Content Delivery](#). The SDK can easily be integrated with any third-party Android mobile application. The SDK lets you fetch content from the server on the fly, without the need for rebuilding the app to modify content. The SDK also provides a wide range of advanced utilities and features such as response caching, a search request builder, and request/response modeling.

The Content SDK for Java is available as an Oracle open-source project on [GitHub](#).

The reference guide can be found here, [Content SDK for Java/Android](#).

 **Note:**

The SDK will work in stand-alone Java applications as it does not contain any direct dependencies on the Android SDK; however, most examples are shown as they would be coded in an Android application as it is assumed that is the primary target platform. This SDK will also work for Android apps written in Kotlin.

Content SDK for Swift

Oracle Content Management provides a Content SDK for [Swift](#), which is a powerful and intuitive programming language for iOS, iPadOS, macOS, tvOS, and watchOS. The read-only SDK is a package of libraries for retrieving published content items, digital assets, and content layouts that are managed in Oracle Content Management.

The SDK is a light-weight iOS binding that interacts with the [REST API for Content Delivery](#), and can be easily integrated with any third-party iOS mobile application. The SDK lets you fetch content from the server on the fly, without the need for rebuilding the app to modify content. The SDK also provides a wide range of advanced utilities and features such as response caching, a search request builder, and request/response modeling. There are two main libraries:

- **OracleContentCore**, which provides base functionality, including network transport capabilities, that's shared and required by multiple Oracle Content libraries.
- **OracleContentDelivery**, which provides data models and APIs for consuming content from Oracle Content Management.

The Content SDK for Swift is available as an Oracle open-source project on [GitHub](#).

The reference guides can be found here:

- [Content SDK for Swift/iOS \(Core\)](#)
- [Content SDK for Swift/iOS \(Delivery\)](#)

Sites SDK

The Sites SDK for Oracle Content Management is a JavaScript library that provides a set of functions which enable components to have a more integrated experience with Oracle Content Management.

The [Sites SDK](#) is available for download from the Oracle Content Management server:

`http://{server}/_sitesclouddelivery/renderer/app/sdk/js/sites.min.js`

Translation Connector SDK

The Translation Connector SDK for Oracle Content Management is a sample NodeJS implementation of the [Translation Connector REST API](#). The sample accepts an Oracle Content Management translation job zip file, translates all the resources in the file, and returns a new zip file containing all the translations.

The Translation Connector SDK is part of Content Toolkit, which is available on [GitHub](#).

The reference guide can be found here, [Translation Connector SDK](#).

4

Starter Site CLI for React Development

The Starter Site CLI for Oracle Content Management is a quick way to get started with React development, and it requires no build configuration.

In this command-line interface, the `create site` command creates a generic site that is built in React and is an independently runnable application. You can generate a site to run in development mode and in production mode.

- [Install the Starter Site CLI](#)
- [Run CLI Commands](#)
- [Get Content from Oracle Content Management](#)
- [Set Up the Oracle Content Management Server Connection](#)
- [Create a Site](#)
- [Build a Site](#)
- [Run a Site in Development Mode](#)
- [Run a Site with Oracle Content Management Server Content](#)
- [Build a Site for Production](#)
- [Run a Site in Production Mode](#)
- [Structure of the React JS Site Template](#)
- [Generated Components](#)
- [Starter Site Runtime](#)

Install the Starter Site CLI

To install the Starter Site CLI for React development, you can download a zip file, unzip it, and use the `npm install` command.

Follow these steps for installation:

1. Get the files for the Starter Site CLI from here:

```
git clone git@github.com:oracle/content-and-experience-toolkit.git
```

Or you can download from GitHub: <https://github.com/oracle/content-and-experience-toolkit>

2. `cd content-and-experience-toolkit/react-starter-sites`

3. If you are behind a corporate web proxy, configure `npm` to work with your proxy:

```
npm config set proxy http://proxy.company.example.com:8080
npm config set https-proxy http://proxy.company.example.com:8080
```

4. Run the `npm install` command:

```
npm install -g
```

For a Mac, run the following command instead:

```
sudo npm install -g
```

If you want to reinstall the CLI, uninstall it first:

```
npm uninstall -g cecss-cli
```

For a Mac, use the following command instead to uninstall the CLI:

```
sudo npm uninstall -g cecss-cli
```

Run CLI Commands

After installation, you can run the command `cecss -h` to see the usage.

```
Usage: cecss <command> [options]
Run 'cecss <command> -h' to get the detailed help for the command.
Commands:
  cecss create-site <name>          Creates the site <name> for
the
content from local or from OCE server.
  cecss export-server-content <channel> Create content template based
on
the channel <channel>, then export and download the archive from OCE
server.
  cecss list-server-content-types    List all content types from
server.
  cecss list-server-channels        List all channels from server.
  cecss develop                     Start development server.
Watches
files, rebuilds, and hot reloads if something changes.
  cecss build                       Build an OCE starter site.
  cecss serve                       Serve previiously build OCE
starter
site.
Options:
  --help, -h    Show
help
```

```
[boolean]
  --version, -v Show version number
```

Get Content from Oracle Content Management

To create a site to show Oracle Content Management content, you need to specify the source of the content.

There are three ways to get Oracle Content Management content:

- **Oracle Content Management templates**

If an Oracle Content Management template contains content types and content items, you can export it from the Oracle Content Management server and use the template zip file to create a site.

- **Published content from a channel**

You can use the command `cecss export-server-content` to export all published content items from a channel, and then use the generated zip file to create a site.

- **Live content on an Oracle Content Management server.**

Example content has been provided in `StarterBlog_export.zip`.

Set Up the Oracle Content Management Server Connection

Some CLI commands require Oracle Content Management server configuration.

Configure the Oracle Content Management server before you use the following commands:

```
cecss create-site <name > -s
cecss export-server-content <channel >
cecss list-server-content-types
cecss list-server-channels
```

To configure the Oracle Content Management server, create the file `.cec_properties` under the user's home directory and configure the server as follows:

```
cec_url=<the oce server url>
cec_username=<user name>
cec_password=<password>
cec_env=pod_ec
```

Create a Site

You can use the `create-site` command and its options to create a site.

Run the command `cecss create-site -h` to see the usage and examples:

```
Usage: cec create-site <name>
```

Creates the Site `<name>` for the content from local or from OCE server. By default, it creates a `StarterSite`. Optionally specify `-f <source>` to create

from different source.

Options:

```
--from, -f      <source> Source to create from
--content, -c  <content> The absolute path of the local OCE
template zip
file
--server, -s   flag to indicate to use the content types from server
--navtypes, -n <navtypes> The comma separated list of content types
from
server to be used as site navigation
--types, -t   <types> The comma separated list of content types on
the
server, if not specified, all content types will be used
--help, -h    Show
help
```

[boolean]

Examples:

```
cecoss create-site NewSite -c StarterBlog_export.zip
cecoss create-site NewsSite -c ~/Downloads/NewsTemplate.zip
cecoss create-site NewsSite -f ~/Downloads/ReactSiteTemplate.zip -c
~/Downloads/NewsTemplate.zip
cecoss create-site BlogSite -s -n Blog
cecoss create-site BlogSite -s -n Blog -t Blog,Author
```

Content Type Restriction

If the name of a content type contains the character "-" or starts with a digital character, the generated React component for this type won't compile, and thus the site won't be runnable. Do not use a content type whose name violates the restrictions.

Create a Site with Local Content

The local content can be the zip file of an exported Oracle Content Management template or a zip file of exported channel content. For each content type in the zip file, a React component will be generated and the type will also be used for site navigation.

```
cecoss create-site NewsSite -c ~/Downloads/NewsTemplate.zip
```

The site created will be placed in a folder with the same name as the site.

Create a Site with Oracle Content Management Server Content

To create a site with content types from an Oracle Content Management server, you need to configure the Oracle Content Management server first. At least one content type should be specified to use for site navigation.

- Create a site for all content types on the Oracle Content Management server, specifying which types to use in the navigation:

```
cecoss create-site serverSite -s -n Article
cecoss create-site serverSite -s -n Article,Author
```


- Create a site with certain content types on the Oracle Content Management server. The content types used for navigation will be included automatically.

```
cecsc create-site serverSite -s -n Article,Author -t Employee
```

Build a Site

Before a site can be run, dependencies need to be fetched. This can be done with the `npm install` command.

If a site is later edited, there is no need to run `npm install` again unless you want to add new dependencies.

After you create a site, run the `npm install` command.

```
cd <site name>  
npm install
```

Run a Site in Development Mode

Running a site in development mode starts a NodeJS server and allows changes to the site to be automatically deployed to the running server.

```
cd <site name>  
cecsc develop
```

This command will start a hot-reloading development environment, and the site can be viewed in a browser using the following URL:

```
http://localhost:9090/
```

After JavaScript, HTML, and CSS files are updated under `src`, the saved changes will live reload in the browser.

Run a Site with Oracle Content Management Server Content

For a site created with content types from an Oracle Content Management server, it's required to set up the site to use the Oracle Content Management server to get the content. Also, a site created from a local content zip file can be changed to run with content from an Oracle Content Management server in the same way.

For each created site, an empty `.cec_properties` file is generated in the site directory:

```
#  
# To show the content on Oracle Content Management server, cec_url and  
# cec_channel_token must  
# be set.  
# If the channel is secure, cec_username and cec_password are also required.  
# Only published items will be displayed.  
# If the Oracle Content Management instance is a development env, set  
# cec_env to dev_ec.
```

```
#
cec_url=
cec_username=
cec_password=
cec_env=pod_ec
cec_content=server
cec_channel_token=
```

Edit the file and set the url and channel token, and username, password if needed.

You can get the channel token from the Oracle Content Management server or use the CLI:

```
cecscs list-server-channels
```

Build a Site for Production

The `build` command will generate an optimized production build of the site:

```
cecscs build
```

Run a Site in Production Mode

Running in production mode optimizes a site and starts an optimized server, without hot-update monitoring.

```
cd <site name>
cecscs serve
```

This command will start a local node server for the site, and the site can be viewed in a browser using the following URL:

```
http://localhost:8080/
```

To run the site on a different port, use:

```
cecscs serve -p <port>
```

Structure of the React JS Site Template

The `StarterSite.zip` site template is used to create a site by default. This template comes with the CLI.

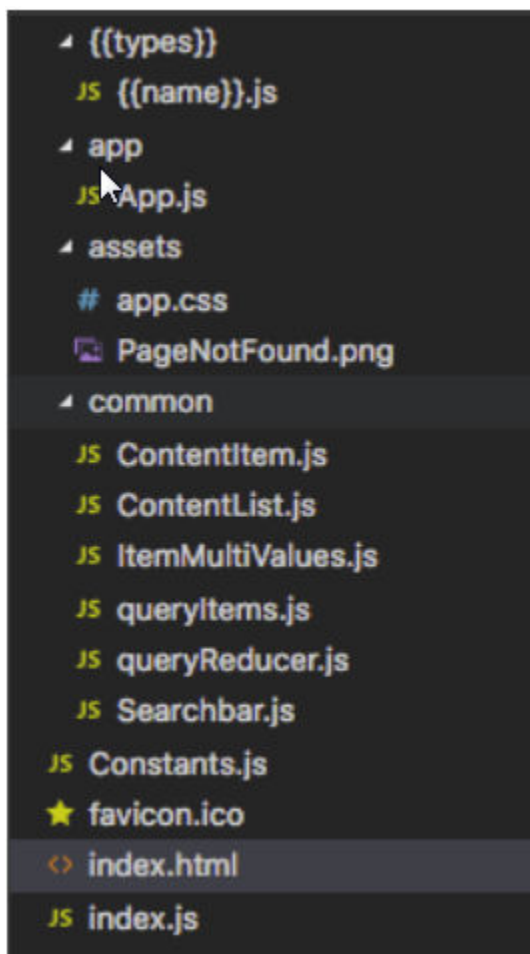
You can find it here:

- Windows:

```
C:\Users\<userid>\AppData\Roaming\npm\node_modules\cecscs-cli\data
```

- Mac

```
/usr/local/lib/node_modules/cecss-cli/data
```

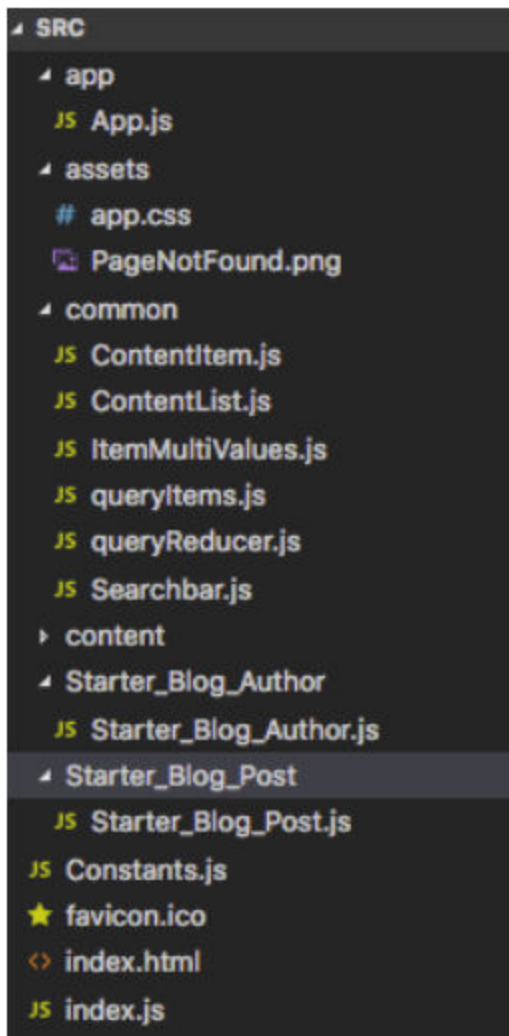


In the template:

- `index.html` is the page template.
- `index.js` is the JavaScript entry point.
- `Constants.js` is where constants are defined.
- `app/App.js` is a React component that is the main parent component of the Simple Page App. The React router is used to decide which component to show and which to hide.
- `assets/app.css` is the css used by the site.
- `{{types}}/{{name}}.js` is a placeholder. React components will be generated for each content type to render the content item, content list, or search result of this content type, based on parameters
- `common/ContentItem.js` is a React component that will render an item with passed-in layout, and will also be responsible for kicking off the item query.
- `common/ContentList.js` is a React component that will render the content list with passed-in layout, and will also be responsible for kicking off the items query.

- `common/ItemMultiValues.js` is a React component that will render an item's field with multiple values.
- `common/queryItems.js` contains JavaScript APIs that create Redux actions to fetch an item or items.
- `common/queryReducer.js` is the reducer that saves the query result into the Redux store when it receives the fetch success action. It'll also set a loading flag to true when the fetch begins, and false when it finishes or fails.
- `common/Searchbar.js` is a React component that will render the search field, and also be responsible for kicking off the search for content items.

After a site is created, all the source code is under `<site name>/SRC/`.



In the SRC directory:

- `content` is the folder that contains the Oracle Content Management content.
- `Starter_Blog_Author/Starter_Blog_Author.js` contains React components generated based on the placeholder in the site template for the content type `Starter_Blog_Author`.

- `Starter_Blog_Post/Starter_Blog_Post.js` contains React components generated based on the placeholder in the site template for the content type `Starter_Blog_Post`.

Generated Components

For each content type, React components are generated to render the content item, content list, or search result, based on the parameters.

The component can be called as follows:

```
<Starter_Blog_Author />
```

Supported parameters for the component:

- `id`
- `search`
- `limit`
- `orderBy` (`name:asc` | `name:des` | `updatedAt:des` | `updatedAt:asc`)

The parameters should be passed in inside the `match.params` object:

```
{
  ...
  match: {
    params: {
    }
  }
}
```

For example:

```
class Starter_Blog_PostDetail extends React.Component {
  render() {
    var item = this.props.item;
    if (!item) {
      return (
        <div />
      );
    }
    var authorId = item.fields['starter-blog-post_author'] ?
item.fields['starter-
blog-post_author']['id'] : '';
    var authorProps = {match: {params: {id: authorId}}};
    return (
      <div>
        <div className="Starter_Blog_Post">
          <span>{item.fields['starter-blog-post_title']}</span>
          <span>{item.fields['starter-blog-post_summary']}</span>
        <div>{renderHTML(item.fields['starter-blog-post_content'])}</div>
          <span>{item.fields['starter-blog-post_category']}</span>
          <ItemMultiValues type='image' values={item.fields['starter-blog-
post_download_media']}>

```

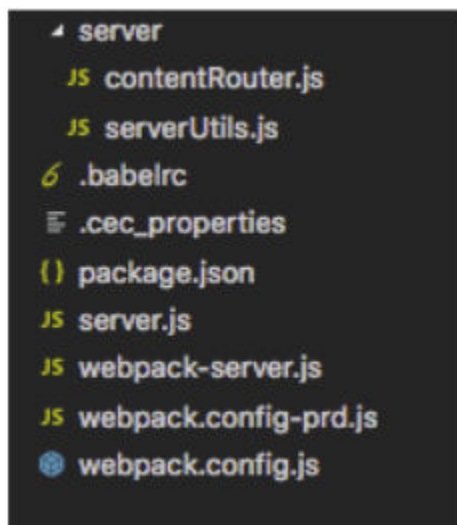
```
        </div>
        <hr/>
        <Starter_Blog_Author {...authorProps} />
    </div>
    );
  }
}
```

Starter Site Runtime

When a site is running in development mode, two servers are running. One is a node Express server with Webpack middleware that helps with live and hot reloading. The other is a node Express server that handles content query.

In production mode, there is only one server running. All the client-side code will be bundled into static files using Webpack, and it will be served by the node Express server.

Details about the starter site runtime follow.



In the starter site runtime:

- `.babelrc` is the Babel config. Babel is used to compile and also to convert JSX to JavaScript.
- `package.json` contains the site's metadata.
- `server.js` is the entry point to the node Express server, which handles content query.
- `webpack-server.js` is the entry point to the node Express server with Webpack middleware, which handles live and hot reloading.
- `webpack.config.js` is the Webpack configuration for development mode.
- `webpack.config-prd.js` is the Webpack configuration for production mode.

5

Connecting to Headless Experiences

Oracle Content Management provides a way to connect content repositories and publishing channels to headless experiences developed and managed outside of Oracle Content Management and automatically trigger deployments based on content changes or published status.

Content providers can leverage the advantages of repository asset management such as powerful tools to organize, retrieve, translate, collaborate on, approve, and publish content. Then, without leaving Oracle Content Management, they can preview, in context and with content, their headless applications.

Experience developers can work with tools they have and configure headless experiences to automatically build based on changes to content in associated repositories or publishing status of content in associated publishing channels, to drive continuous integration/continuous deployment (CI/CD).

The following topics describe how to create and configure connections to experiences outside of Oracle Content Management:

- [Create an Experience Object](#)
- [Configure Experience Object Properties](#)
- [View Connected Headless Experiences](#)
- [Sharing an Experience Object](#)

Create an Experience Object

To create an experience object in Oracle Content Management that connects to a headless experience outside of Oracle Content Management, you need to be an enterprise user with the developer role and have a good understanding of the 3rd party application used to create and manage the experience you are connecting to.

 **Note:**

Only one experience object can be created on an Oracle Content Management Starter Edition instance. For unlimited objects, upgrade to Premium Edition.

1. Click **Experiences** on the side navigation panel.
2. On the Experiences page, click **Create**.
3. Enter a name, optional description, and default URL of the headless experience you are connecting to, then click **Create**. The default URL is typically the site URL of the headless experience you're connecting to and is necessary to view the experience in Oracle Content Management.

Note: there are some limitations in the type of information that can be used in each field.

Field	Limitation
Name (required)	<ul style="list-style-type: none"> Maximum string length is 256 characters. The string must not contain the characters; " : ? < % > { } / # \ *
Description (optional)	<ul style="list-style-type: none"> Maximum string length is 500 characters. The string may contain any characters.
Experience URL (required to open experience preview)	<ul style="list-style-type: none"> Maximum string length is 4000 characters. The string may contain any characters valid in a URL. The string should parse to be a valid URL.

Configure Experience Object Properties

Once an experience object has been created, you must configure it to include outgoing targets that automate headless experience deployment based on content lifecycle events from Oracle Content Management. When a target is triggered, Oracle Content Management sends a request to the URL endpoint you specify in the outgoing target when you define the target. A URL endpoint is typically implemented in your headless experience to receive and respond to an outgoing target, typically automating a CI/CD workflow. For example, a notification about updated content might initiate a new build of the experience, flush caches, or take any other actions needed for the experience to use the updated content.

1. On the Experiences page, select the experience object and click **Properties** in the actions bar.
2. Under the properties tab, you can review information about the experience object, modify the description or default URL, and enable or disable the headless experience connection. Disabling it prevents content lifecycle events from triggering targets in the experience object.

Up to ten targets per experience can be configured to handle different deployment scenarios based on certain content lifecycle events. For example, one target may be triggered when a new version of an unpublished asset is updated in a repository, initiating a preview of the experience, while a second target may be triggered when an asset is published or republished to a channel, initiating a new build.

The response received back from the endpoint is processed using rules defined in the [Analyze tab](#) when configuring the experience object properties. The response could include:

- the target identifier that was included in the request
- a status message or code
- the URL at which the updated headless experience can be viewed

Add Outgoing Targets to Experience Objects

To automate deployment and preview of headless experiences from within Oracle Content Management, you must add targets and specify trigger events on the **Outgoing** tab of the experience object properties. You can configure up to ten targets to handle different deployment scenarios based on certain content lifecycle events.

1. With the experience object properties panel open, click the **Outgoing** tab.

2. To add an outgoing target, click **Add**.
3. Enter a name for the target, the URL endpoint, and specify which method to use, **GET** or **POST**.

 **Note:**

URL endpoints must use the secure https protocol.

4. Choose the Oracle Content Management event that triggers the target to deploy and specify the repositories (for asset change events) or channels (for asset published events).
 - **Changes**—the target is triggered when an asset is changed in a specified repository.
 - **Publishes**—the target is triggered when an asset is published in a specified publishing channel.
5. Click **Headers** and **Body** to add any additional required information, if necessary.
6. Click **Save**.
7. Click **Test** to manually trigger the target. Triggering the target sends a request to the target endpoint to solicit a response, typically in the form of a JSON body payload. If the request is valid, you can refresh the events listed on the Events tab to see it listed with additional information. If the request is invalid, an error message is displayed.

Add a TARGET_IDENTIFIER Token

You can add the `{{TARGET_IDENTIFIER}}` token to an outgoing target. This token will add the target ID to the request when the target is triggered either automatically by content lifecycle events or manually by clicking **Test**. Where you add the `{{TARGET_IDENTIFIER}}` token depends on the 3rd party application you use to manage your headless experience. In some cases it may be added to the URL endpoint, to the HTTP body, or HTTP header.

When an incoming payload is received, the payload is evaluated to see if a target ID exists. If a target ID is found that matches an outgoing target ID in an Oracle Content Management experience object, then that payload is associated with the outgoing target that has that ID.

Once a payload is associated with a target, then the event item is logged on the Events tab with the proper title. If the payload is associated to a target and a URL is matched with the URL introspect string configured on the Analyze tab of a target, then that URL is stored as the preview URL for that target.

If a target ID is not found in an incoming payload that matches an outgoing target ID, the payload cannot be associated to a target and it is displayed as Unknown in the Events tab.

Enable and Disable Incoming Webhook

Oracle Content Management makes it easy for your content contributor to view the latest versions of your headless experience. When your headless experience is updated, your deployment system can notify Oracle Content Management by sending a request to the experience webhook. All incoming requests are logged to the Events tab.

You can review the incoming webhook URL and enable or disable it on the **Incoming** tab of an experience properties panel.

Analyze and Extract Payload Information

The Analyze tab allows you to extract information from an incoming response payload and display it on the Events tab to provide information important to content editors and contributors, such as when their content is saved, published, or goes live.

When a target is triggered and a request is sent to the endpoint URL of your headless experience, the experience returns a response payload. The response payload is combined with the request and the combined data structure is displayed as an event item on the Events tab, with the response payload in an `httpBody` section of an event item.

On the analyze tab, you can encode the JSON path to the information you want to extract within double-braces `{{}}` to create an introspect string that extracts the requested data from the `httpBody` section of a payload. The extracted information is displayed in the event listing on the Events tab.

For example, let's say a section of the payload is as follows:

```
"httpBody": {
  "data": {
    "url": "https://sample.com",
    "status": "Building",
    "name": "Sample Target"
  }
}
```

You would specify the JSON paths to the URL, status, and name data on the Analyze tab as the following URL Introspect strings:

```
{{data.url}}
```

```
{{data.status}}
```

```
{{data.name}}
```

The screenshot shows a dialog box titled "Properties for Experience 'Xalco'". At the top right are "Close" and "Save" buttons. Below the title bar are tabs for "Properties", "Outgoing", "Incoming", "Analyze", and "Events". The "Analyze" tab is selected and underlined. The section is titled "Analyze Payloads" and contains the instruction: "Use this tool to introspect notification payloads and extract relevant information for Experience viewers in the event stream." There are three input fields, each containing a placeholder string: the first is "{{data.url}}", the second is "{{data.status}}", and the third is "{{data.name}}". To the right of the second and third fields are "X" icons. At the bottom left is an "Add" button.

The URL, deployment status, and name of the deployment is then listed dynamically on the Events tab.

The screenshot shows the same dialog box, but with the "Events" tab selected and underlined. The "Preview Target" section is expanded, showing a blue link "https://sample.com", the text "Building", and "Sample Preview".

 **Note:**

The URL introspect string has special status and is reserved for identifying a URL to preview the deployment of the associated target. This URL value is stored and used to preview the latest results of a triggered target when viewing an experience object.

You can also hard-code string values. One reason to do this is to label the JSON path result to easily identify the information for a content contributor. For example, when extracting the name of a deployment, you could enter

```
Name: {{data.name}}
```

in an introspect string on the Analyze tab. This would result in **Name: Sample Preview** being displayed on the Events tab for that target.

Similarly, you can combine encoded JSON paths and hard-coded strings in one introspect string entry. For example,

```
Name: {{data.name}} Status: {{data.status}}
```

would result in **Name: Sample Preview Status: Building** being displayed on the Events tab for that target while the preview was building.

If there is a spelling or syntax error when encoding a JSON path in an introspect string, the string interpretation will fail and the information will not be extracted from the payload or displayed on the Events tab. If you are using more than one JSON path in an event string, an error in one will cause all to fail.


View Event Information for an Experience

Event information for a headless experience is listed on the Events tab of an experience object properties panel. You can expand an event in the list to see the complete payload and review details about incoming and outgoing notifications useful to a headless experience developer.

The Events tab logs all information contained in a payload. You can refine what information is displayed in the event list in order to provide useful information to content providers by using introspect strings on the [Analyze tab](#). This helps you and the content provider see useful information quickly, such as build status or preview URL, without having to expand and review the entire payload to find the information.

View Connected Headless Experiences

Once a headless experience outside of Oracle Content Management is connected, content providers can view headless experience events within Oracle Content Management. This automates their workflow by publishing or building a preview of the headless experience when they trigger an event, either by editing or publishing an asset, depending on how the experience is configured.

To view a headless experience, select the experience object and choose **View** from the right-click menu or click  on the actions bar.

Once the experience panel is displayed, choose the target you want to view from the drop-down list. The headless experience is displayed in an iframe. If the experience cannot be displayed properly in an iframe for security reasons, click **Launch** to view the experience in a new browser tab.

Launch Properties After a Successful Experience Creation

Users typically need to customize an experience after creating it. Oracle Content Management launches the properties and switches to the **Outgoing** tab after you complete the create wizard.

Set Security Admin Settings Through APIs

If you have a service administrator role, you can set security admin settings through APIs.

You can also update CORs lists to standardize CORS handling across Docs, Caas, and OSN.

Sharing an Experience Object

If you have created an experience object or are a manager of an experience object that has been shared with you, you can share it with others to allow them to use or edit the object.



Note:

Because experiences are available only to enterprise users, you can share experience objects only with other enterprise users.

1. Open the **Experiences** page and select the experience object you want to share.
2. Select or click **Members** in the right-click menu or action bar.
3. Click **Add Members** in the side panel.
4. Enter the names or email addresses of the people with whom you want to share the experience object. You can delete or modify the optional message sent to new members.
5. Select the role you want the added member or members to have. The selected role is assigned to all members that you list in this process. You can go through the process several times if you want to specify different roles for different members, or change the roles individually once they've been added.

The following roles can be assigned to experience object members.

- **Owner**—The owner is the person who created the experience object and cannot be changed. They have full control of the object, object properties, and members.
 - **Manager**—A manager has full control of the object, object properties, and members.
 - **Contributor**—A contributor has full control of the object and object properties, can see the list of members, but cannot add or remove members.
 - **Viewer**—A viewer can see and use an experience object, but cannot change it in any way. They can also see the list of members, but cannot add or remove members.
 - **Remove**—If you are an owner or manager, you have the option to remove the role assigned to a member, which removes them as a member of the experience object.
6. When you've finished selecting the people you want to add, click **Add**. The people you selected are listed as members in the members side panel.

7. Click **Done** to close the members side panel.

Filtering Experience Objects on the Experience Page

You can select what experience objects are displayed on your Experiences page by selecting an option from the title menu. The following filter options are available.

- **All**—All experience objects created by you or shared with you are listed.
- **Owned by you**—Only experience objects created by you are listed.
- **Shared with you**—Only experience objects shared with you are listed.

6

Instrumenting Headless Sites with Consumption Analytics

Oracle Content Management provides consumption analytics features that enable you to track the usage and popularity of assets on sites.

The following sections describe how 'headless' sites (that is, created and managed outside of Oracle Content Management) can be instrumented with data collection capabilities to integrate with the consumption analytics feature:

- [Analytics Script](#)
- [Asset Events](#)
- [Page Instrumentation](#)
 - [Component Attributes](#)
 - [JavaScript API Calls](#)
- [Configuration Options](#)
- [Add data-asset-operation Markup for Digital Assets](#)
- [Add data-asset-operation Markup for Referenced Field Types](#)
- [Use a Site-Specific Oracle Infinity Account](#)
- [Use Your Own Oracle Infinity Tag](#)

The data collection for consumption analytics happens in JavaScript, typically running in client browsers. This is especially important for headless sites, which are manufactured server-side. Data collection will happen at the point of consumption in the browser through HTTP requests initiated from JavaScript.

Analytics Script

Include the following analytics script in the pages of the headless site:

```
<script type="text/javascript" src="//<instance>.oacedn.oraclecloud.com/_sitesclouddelivery/analytics/ocm-asset-analytics.js?channelId=<channel-id>" async></script>
```

Please note the following:

- This ocm-asset-analytics.js script will be served with a caching duration of two hours.
- When loaded, this will add a global variable, SCSAnalytics, to the window namespace.

The ocm-asset-analytics.js library delivered from the your Oracle Content Management instance has the `serviceId` and `accountId` parameters embedded in it. The only parameter you need to supply is `channelId`.

Parameter Name	Description
serviceId	The service ID of the Oracle Content Management instance that holds the site assets.
accountId	The account ID of the analytics provider.
channelId	The channel ID of the publishing channel used to obtain the site assets.

Asset Events

Four events are currently supported and recorded for assets used on site pages. The following table describes these events:

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	The media asset on the site page has started to play.
download	The user initiated a download operation on the asset.

Page Instrumentation

There are two methods of instrumenting pages for use with consumption analytics: you can augment the rendered asset markup on the page, or manually produce asset events with direct JavaScript calls.

To automatically record asset consumption events, you can add a special "data" attribute to the rendered assets markup on the page:

```
data-asset-operation
```

This attribute instructs the ocm-asset-analytics.js library 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, separated by semicolons (;).

The following table lists the `data-asset-operation` parameters:

Parameter Name	Description
page-event	<p>A page-event normally corresponds to a DOM event, like "click" and "play" events on elements. The supported page events include view, play, and click.</p> <ul style="list-style-type: none"> • 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	<p>One of the supported asset events, including:</p> <ul style="list-style-type: none"> • 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>
```

On the first line, note the `data-asset-operation` attribute. The `ocm-asset-analytics.js` library will automatically generate a "load" event for the asset when this markup appears on the page. Additionally, when the item is scrolled into view in the browser viewport, a "view" analytic event will be recorded.

The `recordAssetOperation` call will return a Boolean value indicating if the call was accepted and will be recorded in the analytics provider. The value `false` will be returned for all other conditions, including invalid parameters.

Component Attributes

To automatically record asset consumption events, you can add a special "data" attribute to the rendered assets markup on the page.

The following table describes 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

Parameter Name	Description
	<p>elements. The supported page events include view, play, and click.</p> <ul style="list-style-type: none"> • 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	<p>One of the supported asset events:</p> <ul style="list-style-type: none"> • load • view • play • download

For example, the final markup of a simple content item might look like this:

```
<div data-asset-
operation="view:COREEBE60D5159507409B97E9B5CD27937B82:view">
  Hello World!
</div>
```

JavaScript API Calls

Site JavaScript can make calls to the ocm-asset-analytics.js library to record asset events. The **SCSAnalytics** object exposes a single function to record asset events.

SCSAnalytics.recordAssetOperation(assetId, assetEvent, options) → {Boolean}

Name	Type	Required	Description
assetId	String	Yes	The ID of the content item or digital asset.
asset event	String	Yes	<p>One of the supported asset events, including:</p> <ul style="list-style-type: none"> • load • view • play

Name	Type	Required	Description
			<ul style="list-style-type: none"> download
options	Object	No	Provides options for the recording operation. Properties include: <ul style="list-style-type: none"> immediate (Boolean) - Indicates that the operation should be recorded immediately, not after a delay.

Configuration Options

JavaScript within the site page can define the `SCSAnalytics.config` object before loading the `ocm-asset-analytics.js` library. Parameters can be supplied to the library without adding to its URL query string.

You can supply the following config parameters:

Name	Type	Description
<code>enableAnalytics</code>	Boolean	Determines if the <code>ocm-asset-analytics.js</code> library performs any further actions after loading. Default: true.
<code>serviceId</code>	String	The service ID of the Oracle Content Management instance holding the site assets.
<code>channelId</code>	String	The channel ID of the publishing channel used to obtain the site assets.
<code>accountId</code>	String	The account ID of the analytics provider.
<code>useObservers</code>	Boolean	Determines if the <code>ocm-asset-analytics.js</code> library should install DOM Observers to

		automatically record analytics events. Default: true.
loadProvider	Boolean	Instructs the ocm-asset-analytics.js library to load the analytics provider's JavaScript. Set this to false if the provider's JavaScript is already included on the page. Default: true.
ready	Array	An array of functions that should be called after the analytics library is loaded and ready to record analytics events.

The following example shows how the ocm-asset-analytics.js library can be parameterized:

```
<script type="application/javascript">
window.SCSAnalytics = {
  config: {
    enableAnalytics: true,
    serviceId: "<service-id>",
    channelId: "<channel-id>",
    accountId: "<account-id>",
    useObservers: true,
    loadProvider: true,
    ready: [
      function() {
        console.log("The asset analytics library is ready.");
      }
    ]
  }
};
</script>
<script type="text/javascript" src="//
<instance>.ocecdn.oraclecloud.com/_sitesclouddelivery/analytics/ocm-
asset-analytics.js" async></script>
```

Add data-asset-operation Markup for Digital Assets

To add data-asset-operation markup for digital assets, add the following HTML code:

```

```

Add data-asset-operation Markup for Referenced Field Types

To add data-asset-operation markup for referenced field types, add the following HTML code:

```
<ul data-asset-operation="view:COREBE60D5159507409B97E9B5CD27937B82:view">
  <li>Name: Joe Bloggs</li>
  <li>Age: 39</li>
  <li>Photo: </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>
```

Use a Site-Specific Oracle Infinity Account

The `account-id` parameter can be used to configure the `ocm-asset-analytics.js` library to record asset operations in a specific Oracle Infinity account. In the script tag that includes the `ocm-asset-analytics.js` library, specify which account ID to use.

```
<script type="text/javascript" src="//<instance>.ocecdn.oraclecloud.com/
_sitesclouddelivery/analytics/ocm-asset-analytics.js?channelId=<channel-
id>&accountId=<account-id>" async></script>
```

When recording to your own Oracle Infinity account, ensure that an Infinity tag named `ocm_asset_analytics` is defined and enabled for production use.

Use Your Own Oracle Infinity Tag

The `ocm-asset-analytics.js` library can be configured to work with an existing Infinity tag loaded on the headless site page.

Use the `loadProvider` configuration value (`false`) to avoid loading the default Infinity script.

```
<!-- Include Oracle Infinity analytics -->
<script type="text/javascript" async src="//c.oracleinfinity.io/acs/account/
<infinity-account-id>/js/<infinity-tag>/odc.js"></script>
```

```
<!-- Include the ocm-asset-analytics.js library, preventing it from loading
its own copy of Infinity -->
```

```
<script type="text/javascript" src="//  
<instance>.ocecdn.oraclecloud.com/_sitesclouddelivery/analytics/ocm-  
asset-analytics.js?channelId=<channel-id>&loadProvider=false" async></  
script>
```

7

Samples

The following samples and tutorials are available to help you build sites using Oracle Content Management as a headless CMS.



React:

- Build a blog: [GitHub](#) | [Tutorial](#) | [Video](#)
 - Build an image gallery: [GitHub](#) | [Tutorial](#) | [Video](#)
 - Build a minimal site: [GitHub](#) | [Tutorial](#) | [Video](#)
 - Use GraphQL: [Tutorial](#) | [Video](#)
 - Use Video Plus assets: [Tutorial](#)
-



Vue:

- Build a blog: [GitHub](#) | [Tutorial](#) | [Video](#)
 - Build an image gallery: [GitHub](#) | [Tutorial](#) | [Video](#)
 - Build a minimal site: [GitHub](#) | [Tutorial](#) | [Video](#)
 - Use GraphQL: [Tutorial](#) | [Video](#)
-



Gatsby:

- Build a blog: [GitHub](#) | [Tutorial](#) | [Video](#)
 - Build an image gallery: [GitHub](#) | [Tutorial](#) | [Video](#)
 - Build a minimal site: [GitHub](#) | [Tutorial](#) | [Video](#)
 - Use GraphQL: [Tutorial](#) | [Video](#)
-

- Use Video Plus assets: [Tutorial](#)
-

**Svelte:**

- Build a blog: [GitHub](#) | [Tutorial](#) | [Video](#)
 - Build an image gallery: [GitHub](#) | [Tutorial](#) | [Video](#)
 - Build a minimal site: [GitHub](#) | [Tutorial](#) | [Video](#)
-

**Next.js:**

- Build a blog: [GitHub](#) | [Tutorial](#)
 - Build an image gallery: [GitHub](#) | [Tutorial](#)
 - Build a minimal site: [GitHub](#) | [Tutorial](#)
-

**Oracle Digital Assistant:**

- Build an Oracle Digital Assistant chatbot: [GitHub](#) | [Tutorial](#) | [Video](#)
 - Integrate Oracle Digital Assistant into a website: [Tutorial](#)
-

**Oracle Visual Builder:**

- Build an image gallery using Oracle Content Management components: [GitHub](#) | [Tutorial](#) | [Video](#)
-

**JavaScript:**

- Build a blog: [GitHub](#) | [Tutorial](#) | [Video](#)
-



Oracle JET:

- Build a blog: [GitHub](#) | [Tutorial](#) | [Video](#)
-