Informatica® Reference 360
January 2023

Reference 360

Informatica Reference 360 Reference 360
January 2023

Publication Date: 2023-01-13

Table of Contents

Preface

Use the Reference 360 help to learn how to create, manage, and govern reference data in Reference 360. Learn how to create reference data sets, code lists, and code values to organize reference data important to your business.

Informatica Resources

Informatica provides you with a range of product resources through the Informatica Network and other online portals. Use the resources to get the most from your Informatica products and solutions and to learn from other Informatica users and subject matter experts.

Informatica Documentation

Use the Informatica Documentation Portal to explore an extensive library of documentation for current and recent product releases. To explore the Documentation Portal, visit https://docs.informatica.com.

If you have questions, comments, or ideas about the product documentation, contact the Informatica Documentation team at infa_documentation@informatica.com.

Informatica Intelligent Cloud Services web site

You can access the Informatica Intelligent Cloud Services web site at http://www.informatica.com/cloud. This site contains information about Informatica Cloud integration services.

Informatica Intelligent Cloud Services Communities

Use the Informatica Intelligent Cloud Services Community to discuss and resolve technical issues. You can also find technical tips, documentation updates, and answers to frequently asked questions.

To access the Informatica Intelligent Cloud Services Community, see Master Data Management SaaS Community.

Developers can learn more and share tips at the Cloud Developer community:

https://network.informatica.com/community/informatica-network/products/cloud-integration/cloud-developers

Informatica Intelligent Cloud Services Marketplace

Visit the Informatica Marketplace to try and buy Data Integration Connectors, templates, and mapplets:

https://marketplace.informatica.com/

Informatica Knowledge Base

Use the Informatica Knowledge Base to find product resources such as how-to articles, best practices, video tutorials, and answers to frequently asked questions.

To search the Knowledge Base, visit https://search.informatica.com. If you have questions, comments, or ideas about the Knowledge Base, contact the Informatica Knowledge Base team at KB_Feedback@informatica.com.

Informatica Intelligent Cloud Services Trust Center

The Informatica Intelligent Cloud Services Trust Center provides information about Informatica security policies and real-time system availability.

You can access the trust center at https://www.informatica.com/trust-center.html.

Subscribe to the Informatica Intelligent Cloud Services Trust Center to receive upgrade, maintenance, and incident notifications. The Informatica Intelligent Cloud Services Status page displays the production status of all the Informatica cloud products. All maintenance updates are posted to this page, and during an outage, it will have the most current information. To ensure you are notified of updates and outages, you can subscribe to receive updates for a single component or all Informatica Intelligent Cloud Services components. Subscribing to all components is the best way to be certain you never miss an update.

To subscribe, go to https://status.informatica.com/ and click **SUBSCRIBE TO UPDATES**. You can then choose to receive notifications sent as emails, SMS text messages, webhooks, RSS feeds, or any combination of the four.

Informatica Global Customer Support

You can contact a Customer Support Center by telephone or online.

For online support, click **Submit Support Request** in Informatica Intelligent Cloud Services. You can also use Online Support to log a case. Online Support requires a login. You can request a login at https://network.informatica.com/welcome.

The telephone numbers for Informatica Global Customer Support are available from the Informatica web site at https://www.informatica.com/services-and-training/support-services/contact-us.html.

CHAPTER 1

Introducing Reference 360

Informatica® Reference 360 is a cloud-based service that organizations use to create, manage, and govern reference data. Reference data is a subset of master data that is used throughout your organization. Common types of reference data are countries, currency codes, and cost centers.

Applications in your organization typically refer to the same reference data by different code values. You can manage the different code value representations by creating crosswalks. Crosswalks contain value mappings between code values in one code list and code values in another code list. Crosswalks allow you to translate between the different code value representations used by each application.

When you manage your reference data, you see improvements in the quality of reporting, compliance with regulations, and the trustworthiness of reference data. It also helps reduce operational overhead, such as additional manual effort and QA effort to resolve issues with your reference data.

Reference 360 provides the following functionality:

- Create, manage, and govern reference data
- Create crosswalks to translate how different applications represent the same business term
- Approve changes to assets before they become part of the published reference data
- View historical information about your reference data assets
- Localization of user interface labels and error messages to Japanese, French, German, Spanish, and Brazilian Portuguese.

Key concepts

System reference data

System reference data contains a group of values that provide information about your reference data. For new organizations, values for the system reference data are empty.

You can add the values that you want to appear in the system reference data. The users can then apply the values to their reference data sets, code lists, code values, and crosswalks. For example, you might add the Low, Medium, High, and Critical values to the Priority system reference data.

You can add values to the following system reference data:

- Application
- Confidentiality
- Domain

- Priority
- Status

Related Topics:

- "Manage system reference data" on page 59

Reference data sets

All reference data in Reference 360 is categorized under reference data sets. A reference data set is a logical grouping of reference data, such as country codes, currency codes, or cost centers.

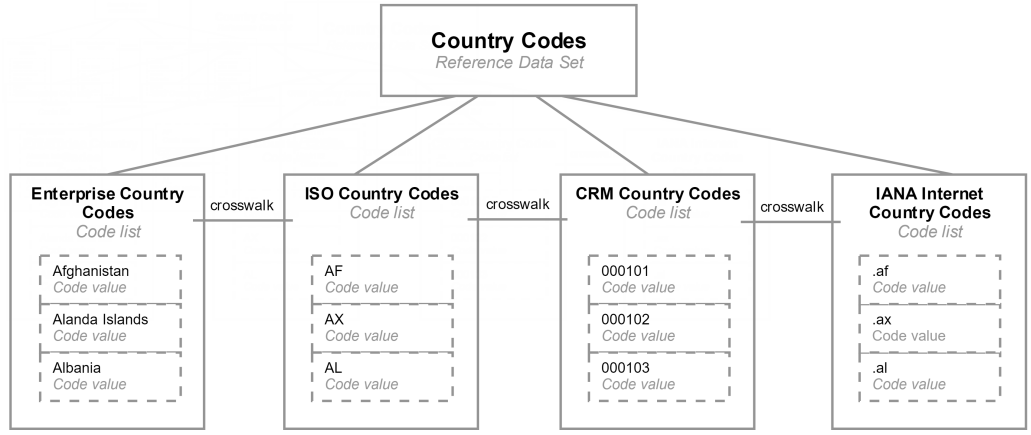
Reference data sets contain code lists. Reference data sets act as both a container and template for the code lists.

When you create a reference data set, you configure its structure definition and attributes. You cannot change the structure definition and attributes of the reference data set after creation. When you add code lists to a reference data set, the code lists inherit the structure definition and attributes from the reference data set. If you do not define the structure definition of a reference data set, you can still define the structure definition of the code lists.

The following table lists the actions you can perform on a reference data set after creation:

Action	Allowed
Delete the reference data set	Yes
Modify the general properties	Yes
Modify the structure definition	No
Modify existing attributes	No
Create additional attributes	Yes
Create required attributes	No
Delete attributes	Yes
Modify the display settings	Yes
Modify the stakeholders	Yes

For example, you might have three different applications, each with a slightly different list of country codes. In Reference 360, you create a Country Codes reference data set. Then you create three code lists in the Country Codes reference data set, one for each of the applications that contain country code values. The last step is to create crosswalks between the code lists so that you can translate how each application represents the same code value.



Related Topics:

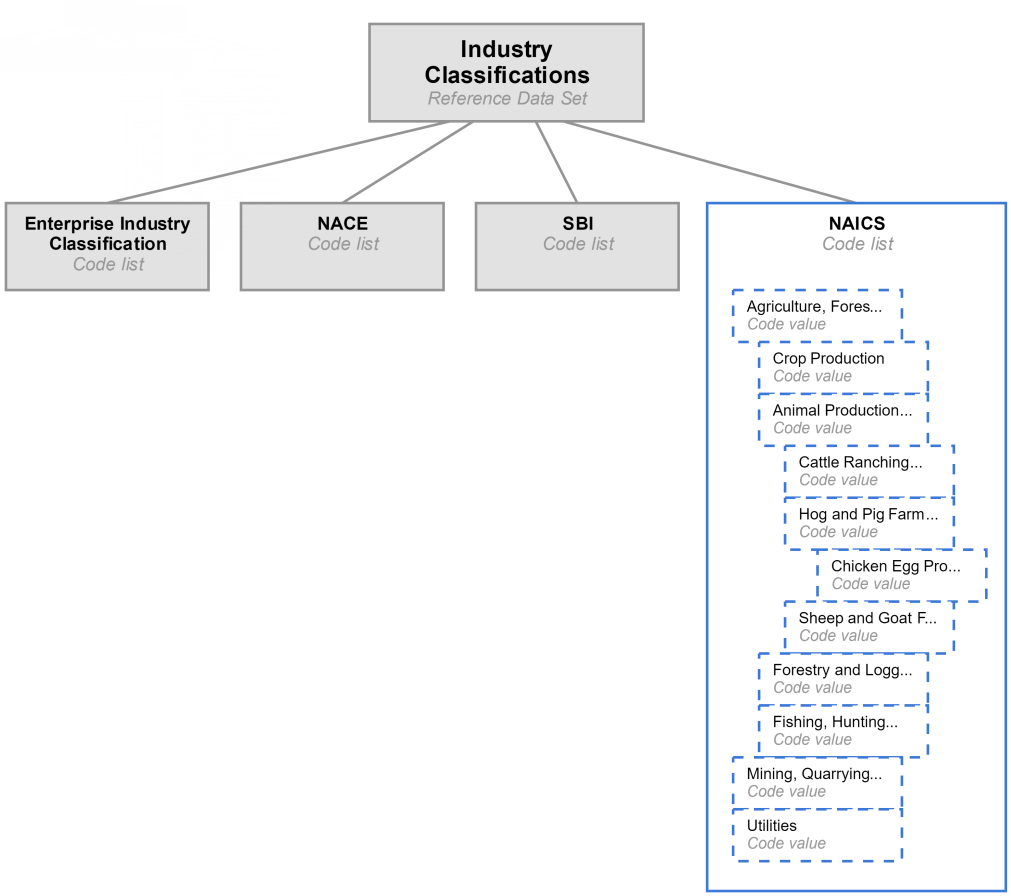
- "Creating reference data sets" on page 61

Hierarchical reference data sets

A hierarchical reference data set allows users to model hierarchical data structures. When you create a code list in a hierarchical reference data set, the code list inherits the hierarchical structure definition. In the code list, you can arrange the code values into levels to create hierarchies. For example, you might create a hierarchical reference data set for industry classification systems or cost centers.

After you create a hierarchical reference data set, you cannot change the structure definition. If you create a code list in a hierarchical reference data set, the code list must use the inherited structure definition.

For example, you want to create an Industry Classifications reference data set. When you create the set, you define the structure definition as hierarchical because you know that industry classifications contain hierarchical code values. In the reference data set, you create an Enterprise Industry Classification code list, a NACE code list, a SBI code list, and a NAICS code list. In the NAICS code list, you arrange the code values into a hierarchy to reflect the hierarchical structure of this externally mandated list.



Related Topics:

- "Hierarchical code lists" on page 17

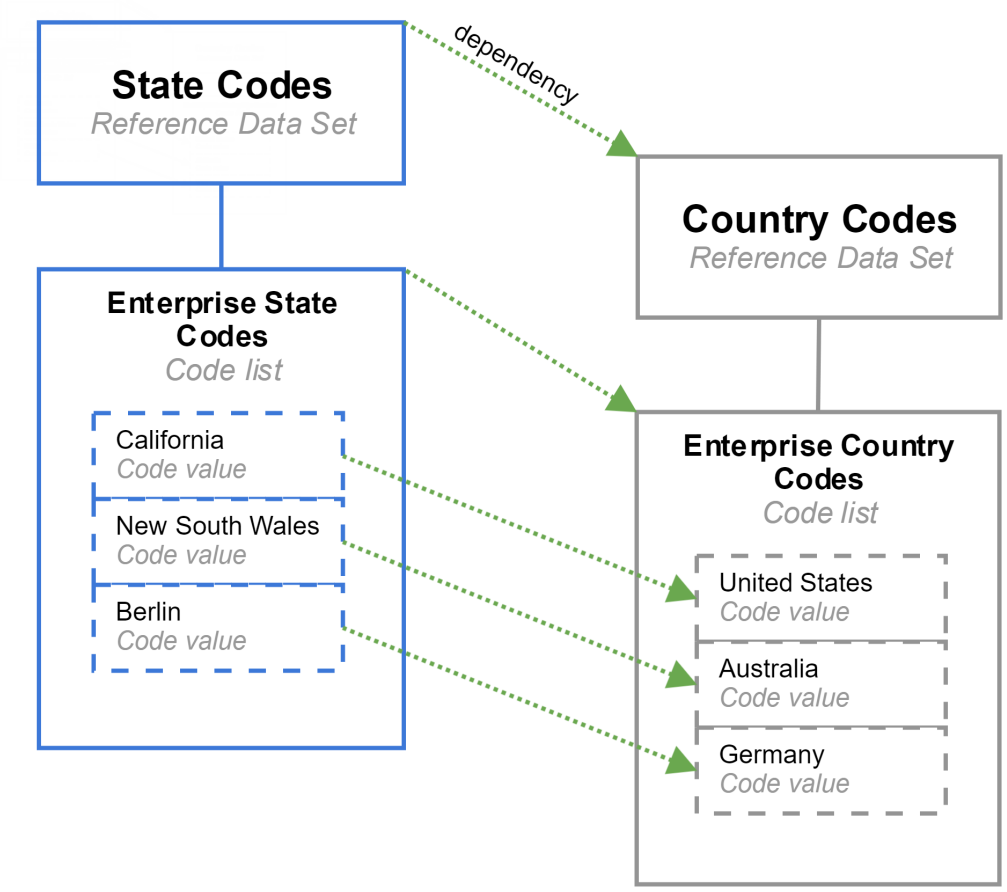
Dependent reference data sets

A dependent reference data set depends on another reference data set and passes down its dependency to its code lists.

When you configure a reference data set as dependent, the reference data set that it depends on is referred to as the parent dependency. Reference data sets can have only one parent dependency. When you create code lists in a dependent reference data set, the code lists inherit the same parent dependency.

After you create a dependent reference data set, you cannot change the structure definition. If you create a code list in a dependent reference data set, the code list must use the inherited structure definition.

For example, you have a Country Codes reference data set. When you create a State Codes reference data set, you define the structure definition as dependent and configure the Country Codes reference data set as the parent dependency. Then when you create code lists in the State Codes reference data set, the code lists inherit the dependency on the Country Codes reference data set. For each code list in the State Codes reference data set, you can choose the code list dependency. A code list dependency defines the dependent relationship between one code list and another. You might want the Enterprise State Codes code list to depend on the Enterprise Country Codes code list.



Related Topics:

- "Dependent code lists" on page 18

Code lists

Code lists are containers for code values. The code values in a code list reflect the code values that are used in a source application. Code lists from different applications might use different code values for the same business term.

A code list inherits the structure definition and attributes from the reference data set. If a reference data set does not have a structure definition, you can configure the structure definition of the code list.

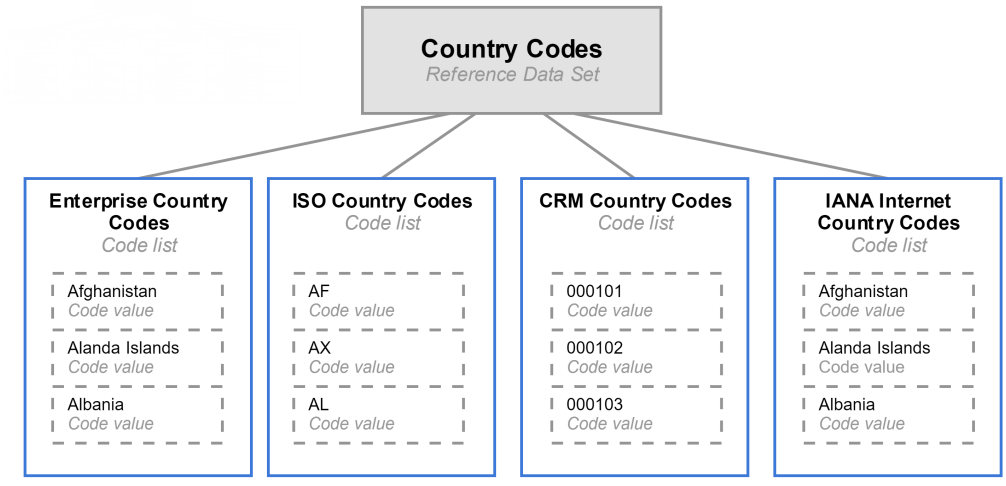
You can provide additional details for a code list, such as additional description, external URLs, and notification recipients.

The following table lists the actions you can perform on a code list after creation:

Action	Allowed
Delete the code list	Yes
Modify the general properties	Yes

Action	Allowed
Add external URLs	Yes
Modify external URLs	Yes
Delete external URLs	Yes
Add notification recipients	Yes
Modify notification recipients	Yes
Delete notification recipients	Yes
Modify the inherited structure definition	No
Modify the inherited attributes	No
Create additional attributes	Yes
Create required attributes	No
Delete attributes	Yes
Modify the display settings	Yes
Modify the stakeholders	Yes

For example, you might have a Country Codes reference data set to categorize all the country codes used in your organization. This reference data set might contain an Enterprise Country Codes code list for country code values that your organization uses for enterprise reporting. You might also have code lists for industry standard lists of values, such as ISO Country Codes and IANA Internet Country Codes. You might also have code lists for systems that store similar code values, such as country codes in your CRM system.



Related Topics:

- "Creating code lists" on page 66
- "Crosswalks" on page 19

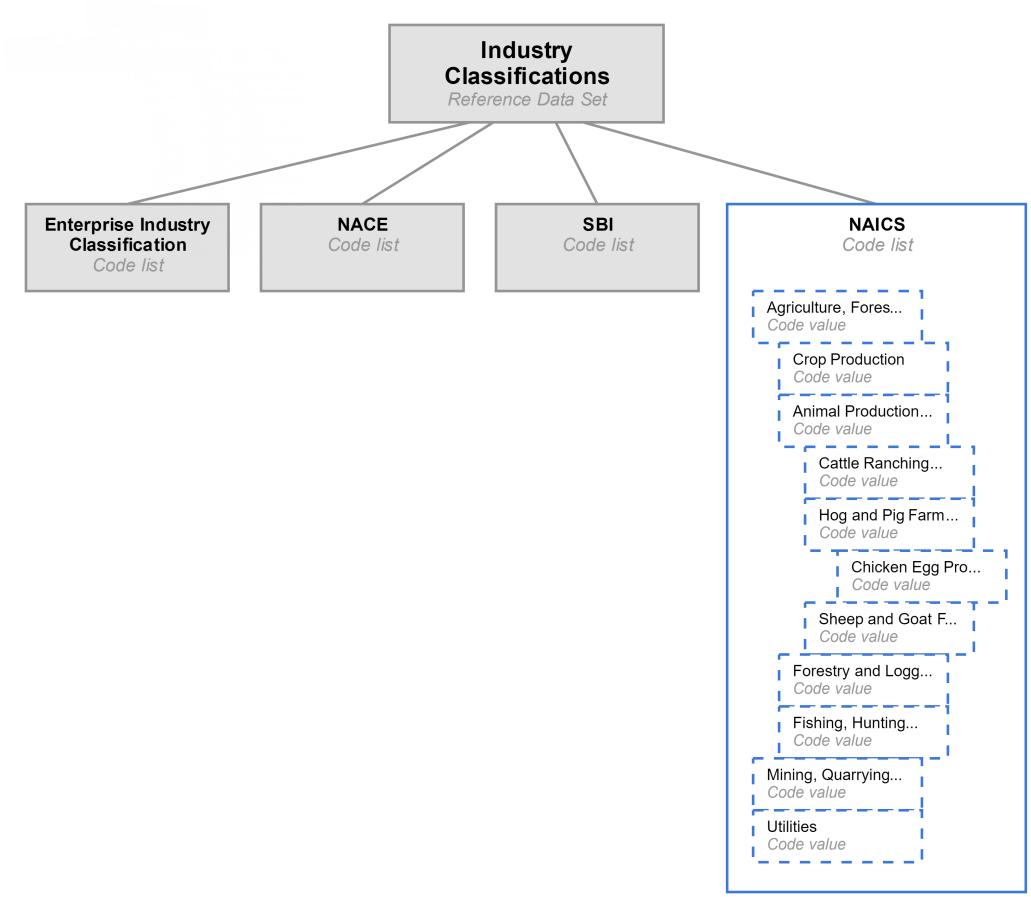
Hierarchical code lists

A hierarchical code list supports hierarchical data structures. You can arrange its code values into levels to create hierarchies.

If you create a code list in a hierarchical reference data set, the code list inherits the hierarchical structure definition from the reference data set. If you create a code list in a reference data set without a hierarchical structure definition, you can configure the structure definition of the code list as hierarchical. After you create a hierarchical code list, you cannot change its structure definition.

For more information about actions that are allowed after code list creation, see "Code lists" on page 15.

For example, you have an Industry Classification reference data set that does not support hierarchical data structures. If you need to create a NAICS code list in the reference data set that supports hierarchies, you configure the structure definition of the code list as hierarchical.



Related Topics:

- "Hierarchical reference data sets" on page 13

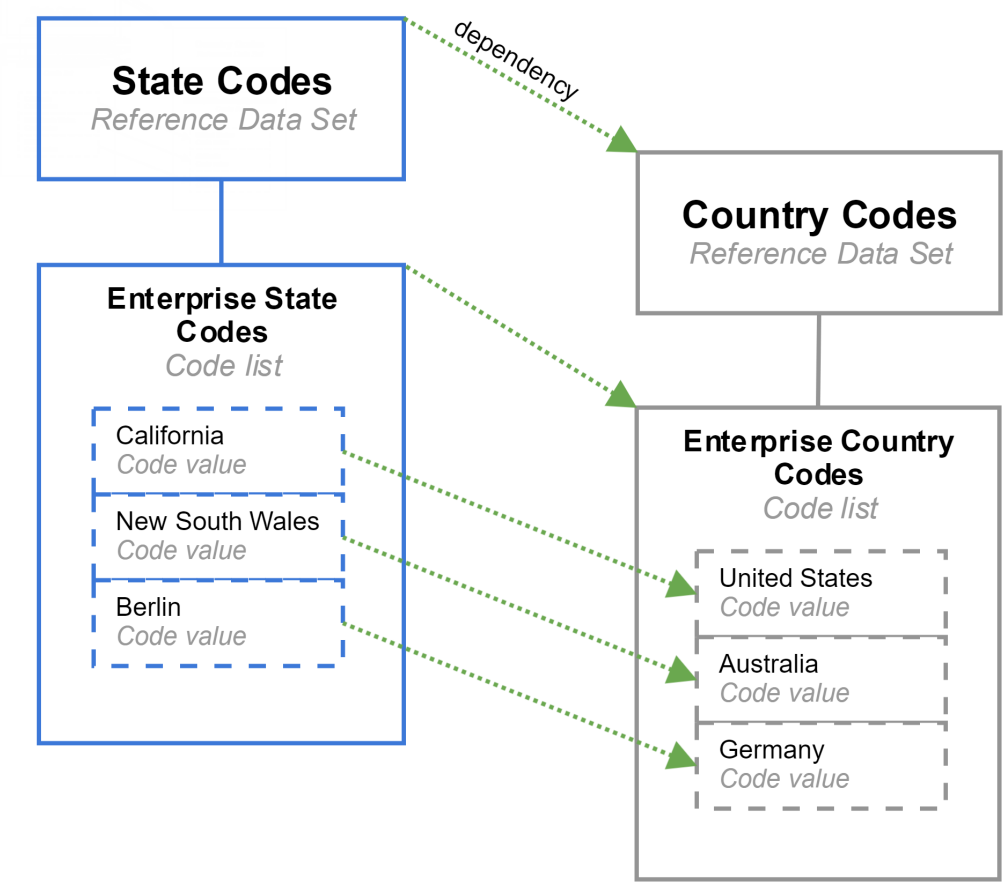
Dependent code lists

A dependent code list contains code values that depend on code values in another code list. The dependency exists between code lists that belong to different reference data sets.

If you create a code list in a dependent reference data set, the code list inherits the parent dependency from the dependent reference data set. If you create a code list in a reference data set without a dependent structure definition, you can define the code list as dependent. After you create a dependent code list, you cannot change the dependency.

For more information about actions that are allowed after code list creation, see "Code lists" on page 15.

For example, you have a Country Codes reference data set. When you create a State Codes reference data set, you define the structure definition as dependent and configure the Country Codes reference data set as the parent dependency. Then when you create code lists in the State Codes reference data set, the code lists inherit the dependency on the Country Codes reference data set. For each code list in the State Codes reference data set, you can choose the code list dependency. A code list dependency defines the dependent relationship between one code list and another. You might want the Enterprise State Codes code list to depend on the Enterprise Country Codes code list.



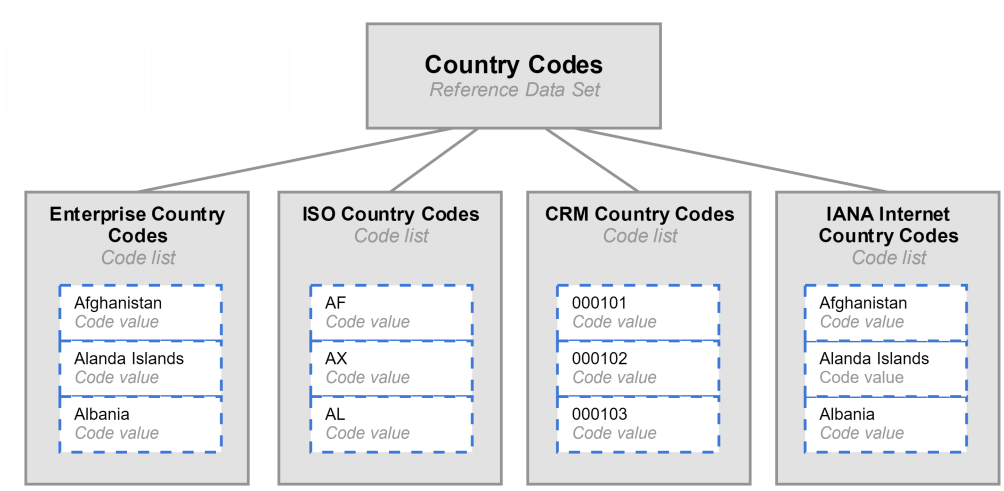
Related Topics:

- "Dependent reference data sets" on page 14

Code values

A code value is a unique value, such as a business term, code, or lookup value. Code values are organized into code lists. Each code list contains code values from a singular source application or industry standard list.

For example, you might have a Country Codes reference data set, and within it is an Enterprise Country code list. In the Enterprise Country code list, you import enterprise country codes values that your organization uses for enterprise reporting. The code list includes code values such as Afghanistan, Aland Islands, and Albania. You might also have a ISO Country Codes code list, a CRM Country Codes code list, and a IANA Internet Country Codes code list. Each of these code lists contain their own code values.



Code values consist of attributes. Each code value consists of a Name attribute and a Code attribute. For example, you might create a code value for the US country code. You define the Name attribute as US and the Code attribute as 001. For more information, see "Attributes" on page 22.

Related Topics:

- "Manage code values" on page 73

Crosswalks

A crosswalk is a visual representation of a one-way relationship between code values in a pair of code lists. A reference data set can contain many code lists, and each code list contains a variation of the same type of code values. Crosswalks provide a way to translate between the different variations each code list uses.

You can create crosswalks for code lists that belong to the same reference data set or different reference data sets. When you create a crosswalk, you configure a source code list and a target code list. You create one-way value mappings between code values in the source code list and code values in the target code list. Value mappings provide a way to translate the code values in the source code list to code values in the target code list. The crosswalk and the value mappings are associated with the source code list.

If code values in a pair of code lists are equivalent, you must create two crosswalks. When you create the crosswalks, use the same pair of code lists in reversed order as the source and target code lists. You must

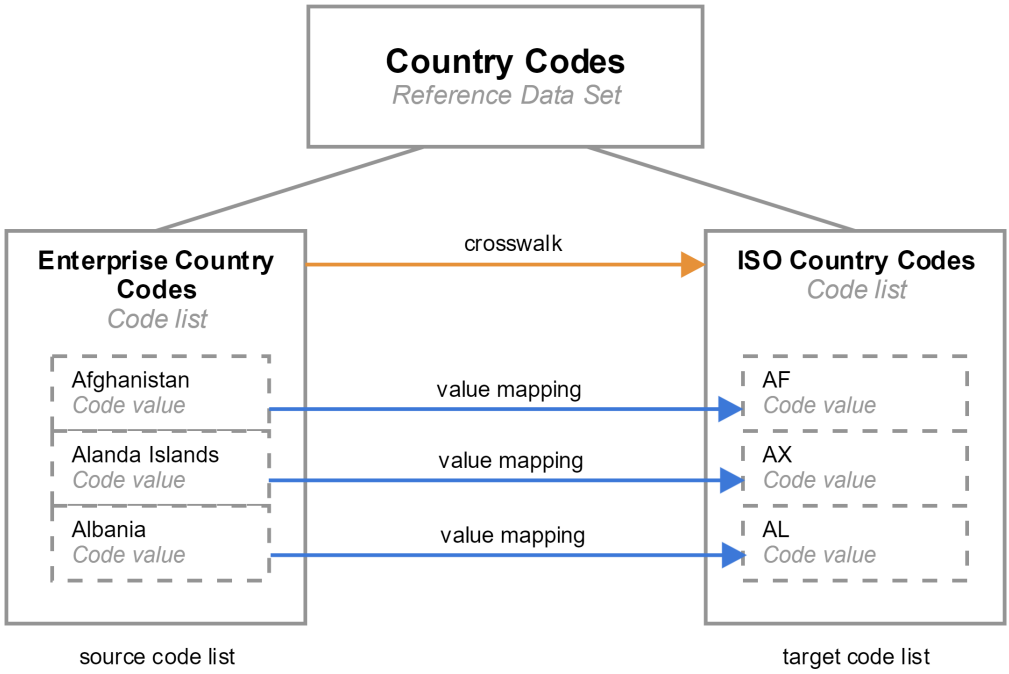
create two crosswalks because each code list can be used as a source code list and crosswalks are stored with the source code list. You cannot create multiple crosswalks for the same source and target code lists.

For example, your organization might have a Country Codes reference data set with code lists for enterprise country codes and ISO country codes. You might create the following crosswalks to map code values in the code lists:

1. Create a crosswalk with the Enterprise Country Codes code list as the source code list and the ISO Country Codes code list as the target code list.
2. Map the code value "Afghanistan" in the Enterprise Country Codes code list to the code value "AF" in the ISO Country Codes code list. The value mapping shows that the code value "Afghanistan" can be translated to the "AF" code value.
3. Create a crosswalk with the ISO Country Codes code list as the source code list and the Enterprise Country Codes code list as the target code list.
4. Map the code value "AF" in the ISO Country Codes code list to the code value "Afghanistan" in the Enterprise Country Codes code list. The value mapping shows that the code value "AF" can be translated to the "Afghanistan" code value.

After you create a crosswalk, the crosswalk is associated to the source code list as an outgoing crosswalk and to the target code list as an incoming crosswalk. Use the **Explore** panel to view code lists and the associated outgoing and incoming crosswalks. An outgoing crosswalk appears below the source code list and shows the source code values that map to the target code values. An incoming crosswalk appears below the target code list and shows the source code values that map to the target code values.

The following diagram shows a sample crosswalk:



Related Topics:

- "Create crosswalks" on page 80
- "Code lists" on page 15

Hierarchies

After you create reference data sets, code lists, and code values, you can create hierarchies. A hierarchy shows how code values are related to one another. You can arrange code values above, below, or at the same level as other code values.

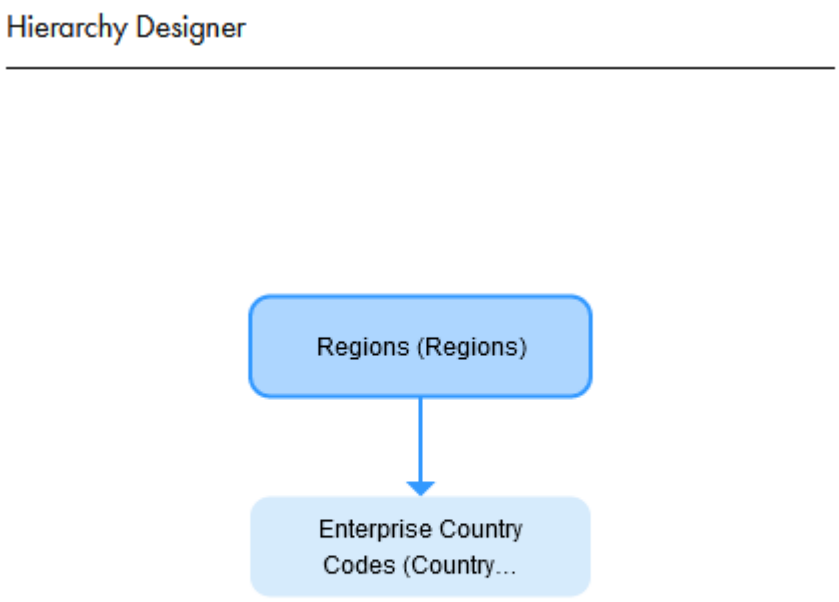
Code values in a hierarchy are top-level nodes or child nodes. Each hierarchy must contain at least one code value as the top-level node. Top-level nodes are code values arranged at the highest level in a hierarchy and do not have any code values arranged above them. Child nodes are arranged below other code values and can have code values arranged below them.

The hierarchy relationships that you can create depends on the hierarchy model. A hierarchy model consists of hierarchy relationships between code lists. You define a code list as the top-level code list and then define levels by adding code lists and relationships between the code lists. Then based on the hierarchy model, you can add code values as nodes in the hierarchy.

For example, your organization needs to track country codes by regions. You might create the following hierarchy model:

1. Create the Locations hierarchy asset.
2. Add the Regions code list as the top-level node.
3. Create a relationship from the Regions code list to the Enterprise Country Codes code list.

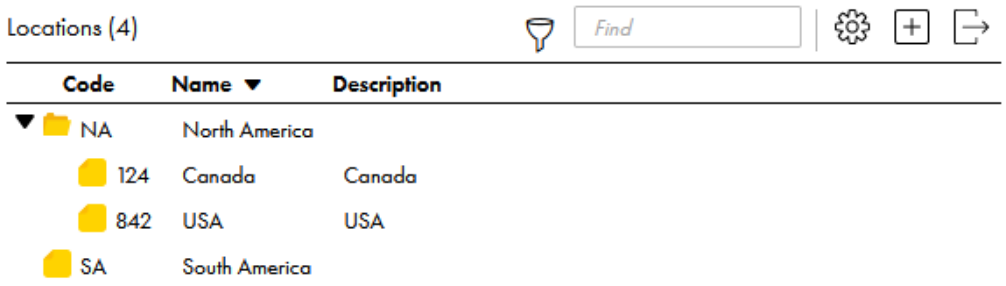
The following image shows a sample hierarchy model:



With the hierarchy model defined, you might create the following hierarchy:

1. Add the North America code value as the top-level code list.
2. Add the USA code value as a child node of the North America code value.
3. Add the Canada code value as a child node of the North America code value.

The following image shows a sample hierarchy:



Note: You must be assigned the Planner role to work with hierarchies. For more information, see "Users, groups, and roles" on page 43 .

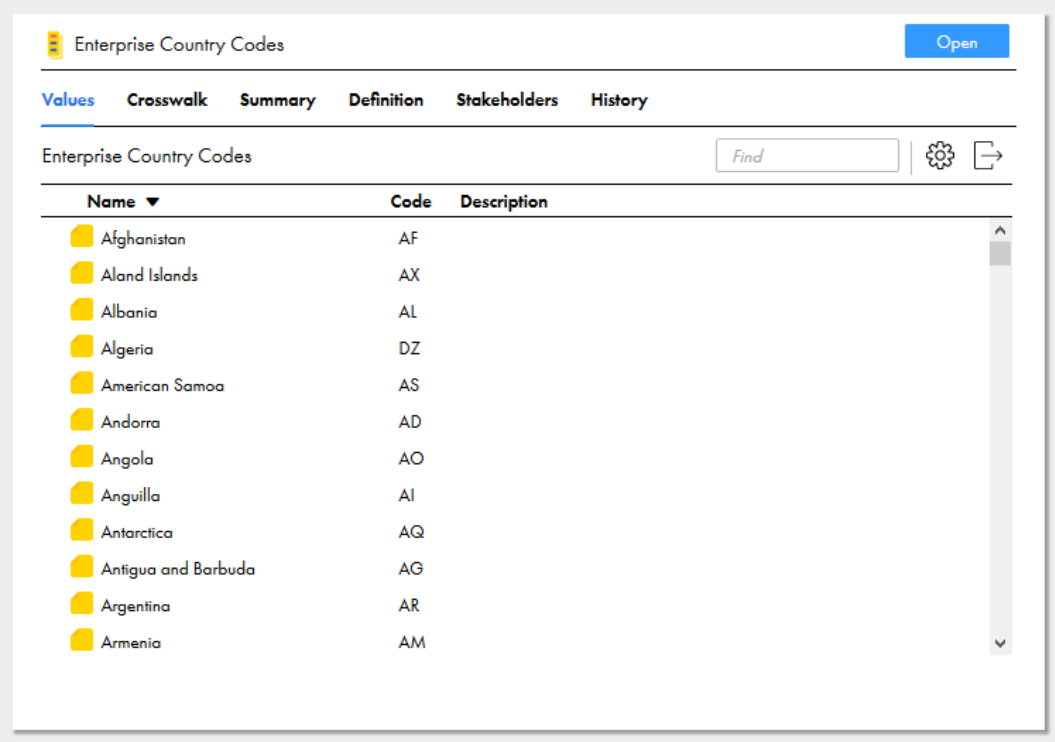
Attributes

An attribute is a component of a code value. When you create reference data sets and code lists, you define attributes that code values must consist of. Then when you add code values, you enter a value for each attribute.

By default, the attributes defined for all code values are the Name, Code, and Description attributes. The Name and Code attributes are required attributes, so you cannot delete these attributes. For each code value, you must enter a value for these attributes. The Description attribute is not required, so you can delete the Description attribute if you do not need it.

For example, you might have a code list for Enterprise Country Codes. When you add a code value for the US country code, you enter the `United States of America` value in the Name attribute. You enter the `US` value in the Code attribute. This means that the US country code value consists of the `United States of America` and the `US` attribute values.

The following image shows the Enterprise Country Codes code list with values in the Name attribute and Code attribute for each code value:



Custom attributes

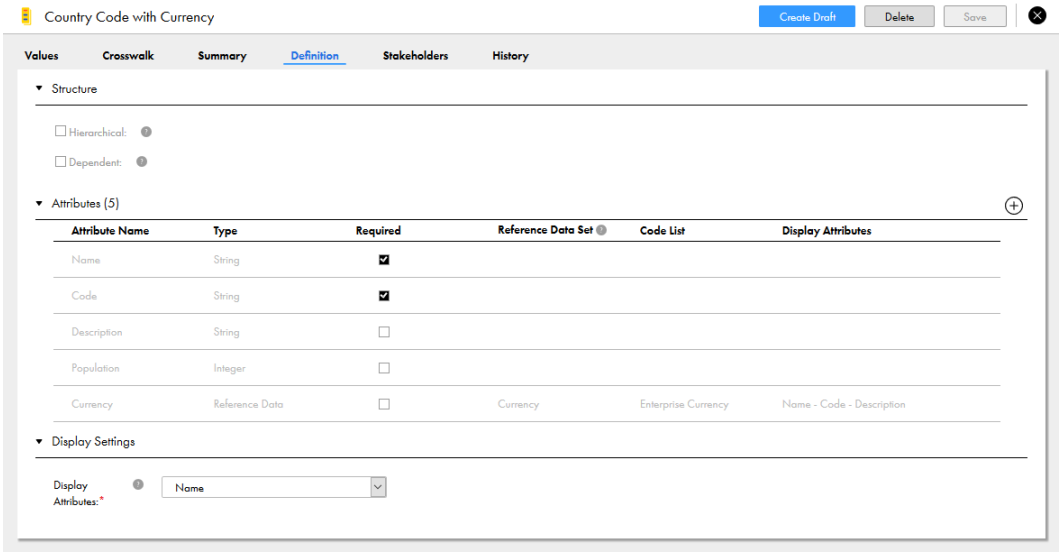
You can define additional attributes that your code values consist of. Your custom attributes can be required or optional attributes.

When you create a reference data set or code list, you can create custom attributes and define them as required or optional. If you define custom required attributes for reference data sets, the code lists inherit the attributes from the reference data set. Then when you create code values in the code lists, the code values must contains values in the custom required attributes. Also, when you create code lists, you can configure custom required attributes in additional to the inherited attributes from the reference data set.

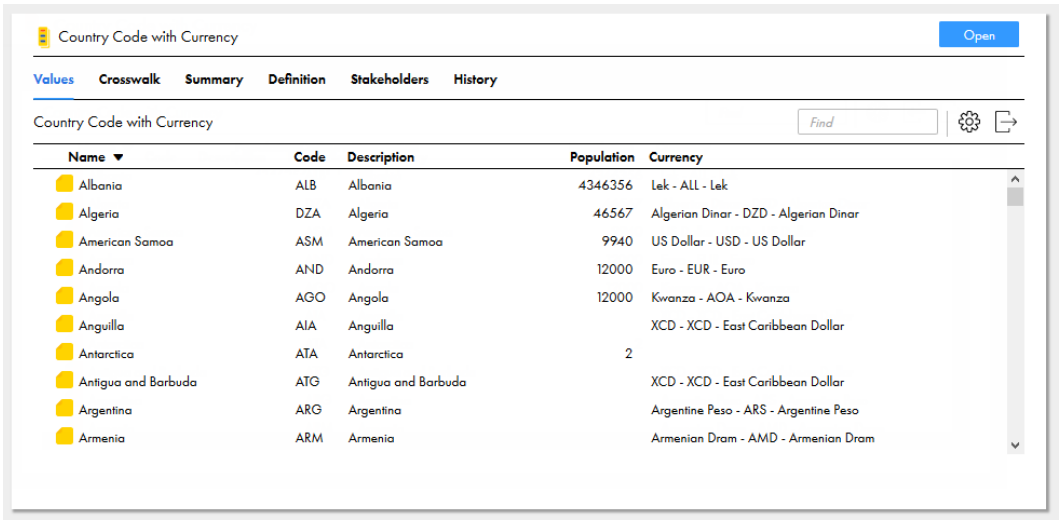
Note: You can only define custom required attributes when you create a reference data set or code list. For more information about allowed actions, see "Reference data sets" on page 12 and "Code lists" on page 15.

For example, you create a Country Codes with Currency code list. You create the optional Population and Currency custom attributes. When you create code values in the Country Codes with Currency code list, you must enter values in the required Name and Code attributes. You can choose to enter values for the Population and Currency attributes.

The following image shows an example of the attributes defined for the Country Codes with Currency code list:



The following image shows an example of the attribute values for code values in the Country Code with Currency code list:



Attribute data types

When you configure custom attributes, you define the type of data the attribute supports.

You can configure attributes to support values in one of the following data types:

- String
- Decimal
- Integer
- Boolean
- Date
- Reference Data

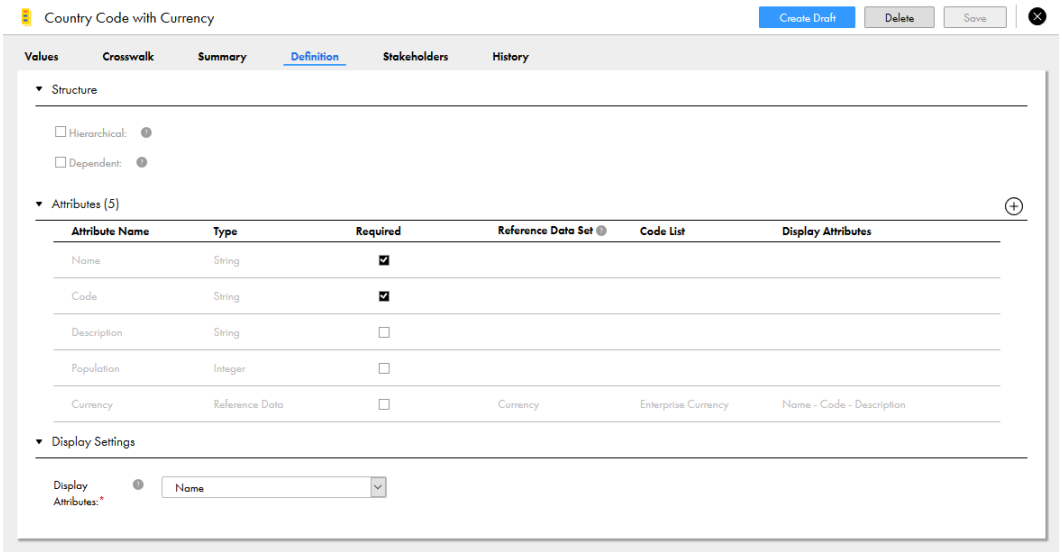
The Reference Data data type supports lookup reference data. Use the Reference Data data type to include the code values from another reference data set and code list. For more information, see "Reference data attributes" on page 25.

Reference data attributes

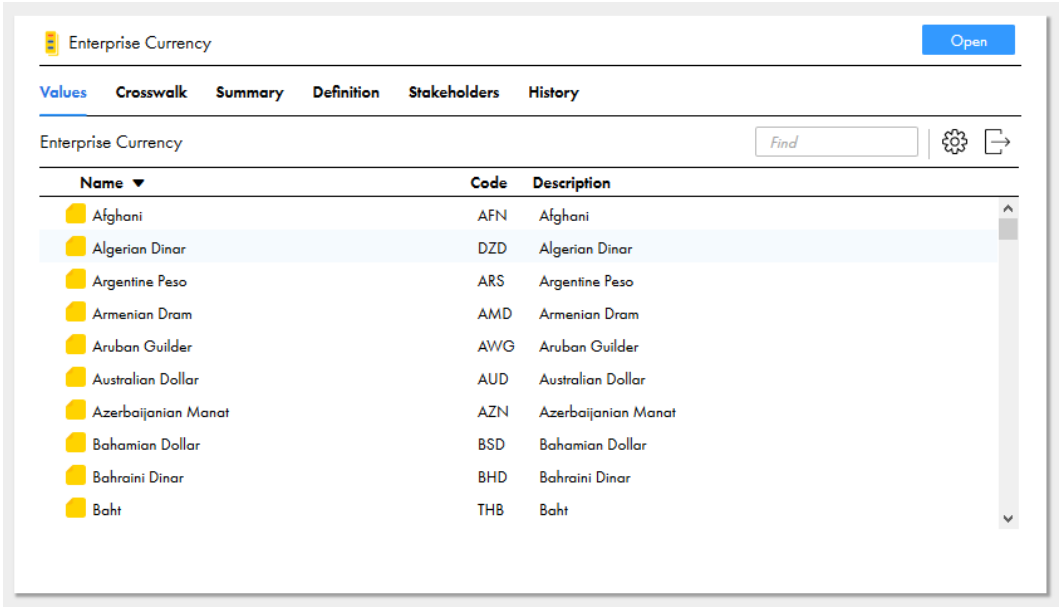
You can create custom attributes that support reference data from other assets. These reference data attributes allow you to use code values in other assets as an attribute value for code values in the code list.

When you create reference data attributes, you configure the reference data set and code list that you want to use reference data from. You also specify the attributes that you want to appear as the display attributes to represent the code values from the selected code list and reference data set.

The following image shows the Currency attribute configured as a Reference Data data type:

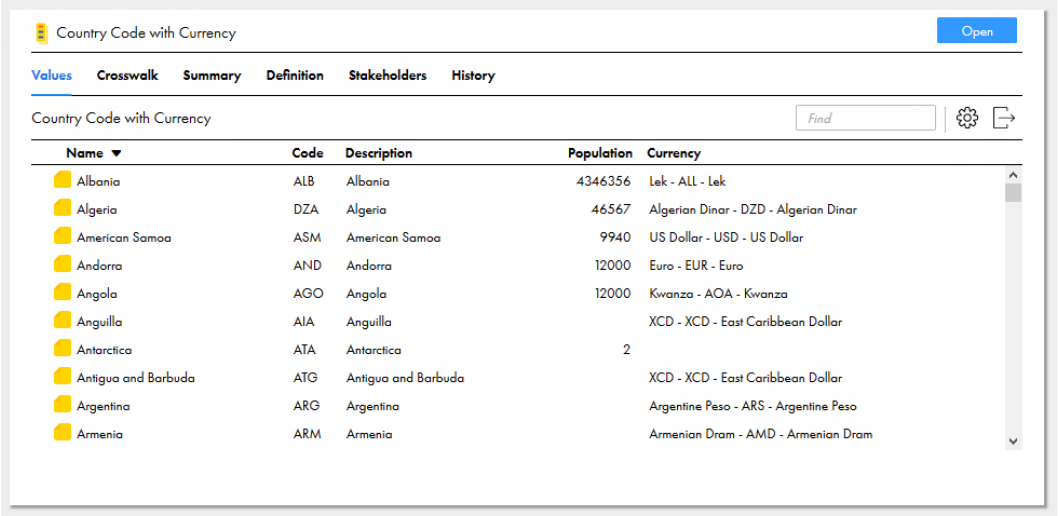


The following image shows an example of the code values in the Enterprise Currency code list that are used as reference data attributes:



When you add code values to the Country Codes with Currency code list, you can select reference data in the Enterprise Currency code list to use as the Currency attribute value.

The following image shows the `Euro - EUR - Euro` code value from the Enterprise Currency code list used as the Currency attribute value of the `Andorra` code value:



Related Topics:

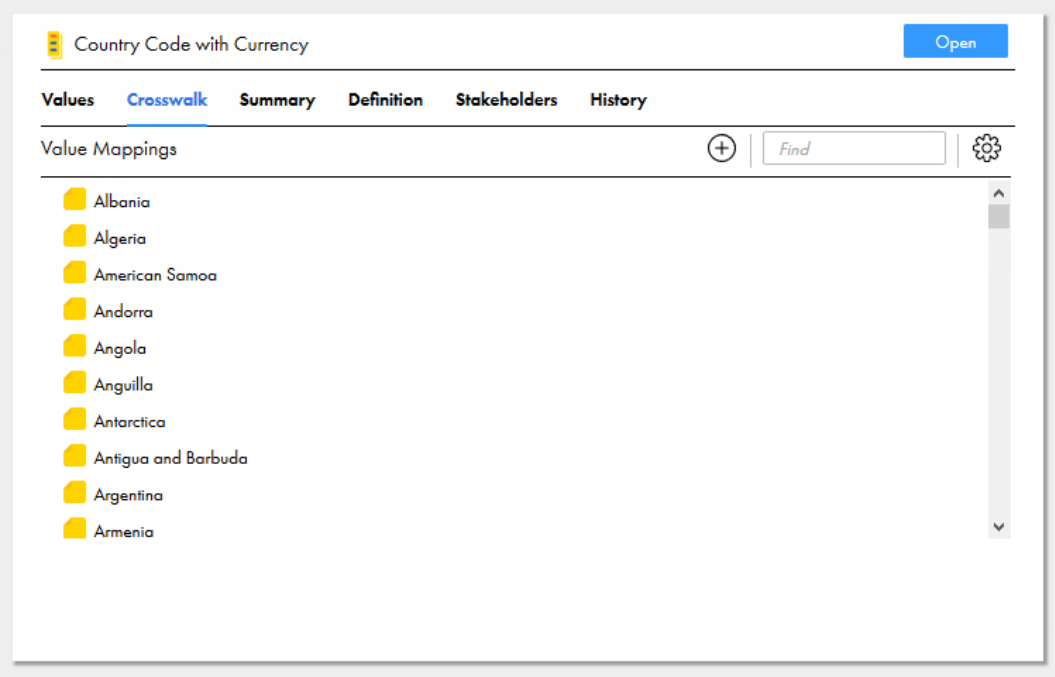
- "Exporting filtered code values" on page 247

Display attributes

Display attributes are the attributes that represent the code values in the asset throughout Reference 360. You configure display attributes for reference data sets and code lists. You can configure multiple attributes as display attributes.

For example, code values in the Country Code with Currency code list might consist of the Name, Code, Description, Population and Currency attributes. You can configure the Name attribute as the display attribute. Later when you create a crosswalk to map code values in the code list, the attribute values in the Name attribute appear for you to map.

The following image shows an example of the values in the Name attribute that represent the code values in the code list:



Batch Jobs

You can define, execute, schedule, and monitor batch jobs in Reference 360.

The batch jobs use the taskflows defined in Cloud Data Integration. Each taskflow can run multiple mapping tasks in a sequential order or in parallel. Each mapping task includes a mapping that maps the source fields with the fields of the Reference 360 assets. To connect to the Reference 360 assets, use the Business 360 connector, and create a connection in Administrator.

For example, if you want to import enterprise country codes from a flat file, create a mapping with a flat file connection as the source transformation and the Business 360 connection as the target transformation. You can then define field mappings between the source and target fields, create a mapping task that uses the mapping, and then create a taskflow and run the taskflow. For more information, see Taskflows in the Data Integration help.

Use Administrator to create a connection for the source transformation and a Business 360 connection for the target transformation. For more information about the Business 360 Connector, see Business 360 Connector in the Data Integration help.

Workflows

After you propose changes to a code list, you send your changes through an approval workflow. An approval workflow is an approval process to review and approve changes to a code list. An approval workflow consists of one approval task or multiple approval tasks linked together.

To propose changes to a code list, you can lock the code list and create a draft version of the code list. The code list is locked to you and you are the only user who can edit it. You can import, create, change, or delete code values in the code list. Your edits are associated with the draft, so you can come back to it another time to continue editing. A lock icon appears beside the locked code list, so other users know that you are working on it.

When you are finished editing the code list, you send your draft for approval. This action initiates an approval workflow and creates an approval task. Users responsible for approving the approval task are notified of the approval task. The code list remains locked to you as it moves through the approval workflow.

Note: To use workflows, you must add values to the Priority system reference data. For more information, see "Adding values to system reference data" on page 59 or "Add system reference data values" on page 236.

Workflow Configurations

You can configure the workflows that are triggered when users edit code lists. You can define the number of approval steps, approvers for each approval task, and whether comments are required for task actions.

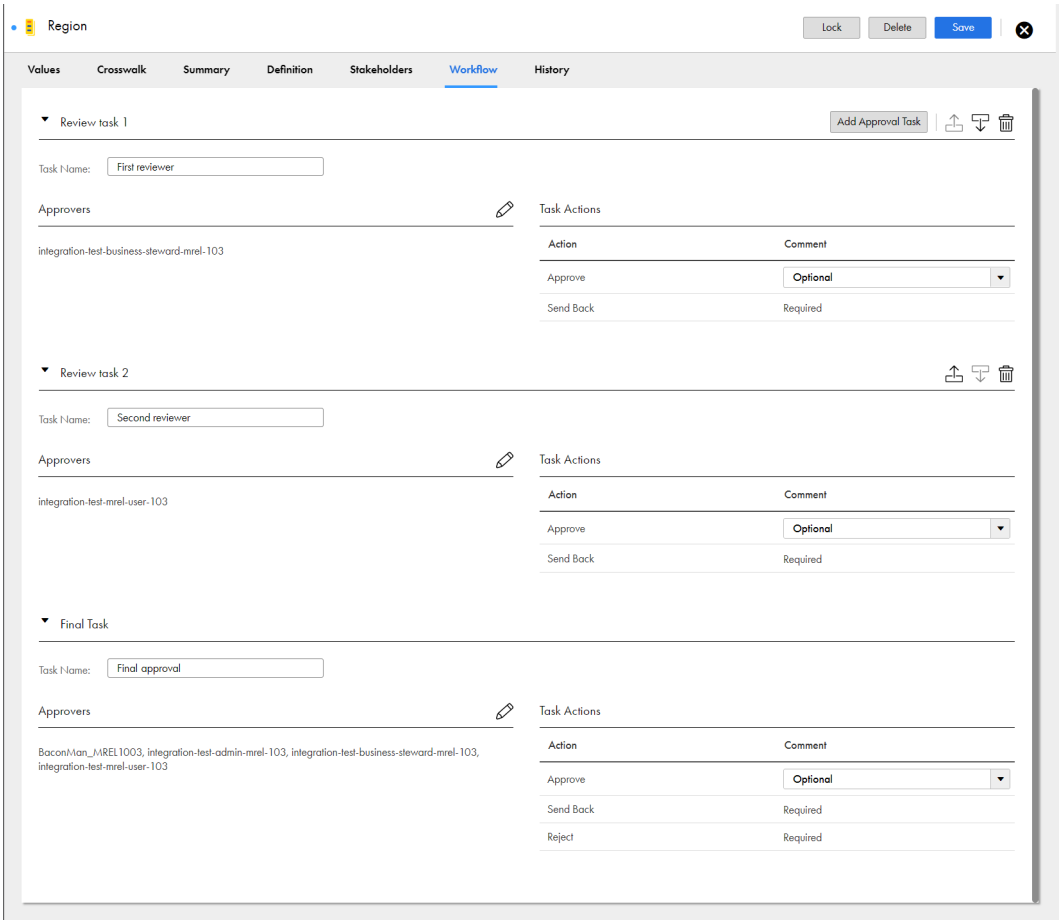
By default, one-step approval workflows are configured for all code lists. To create multi-step approval workflows, you add additional approval tasks to the workflow configuration of a code list.

The following table describes the approval workflows you can configure:

Approval Workflow	Description	Activity
One-step	One-step approval workflows contain one approval task. This workflow is the default approval workflow for all code lists.	An approver can approve the changes, reject the changes, or send the changes back to the originator. If the changes are approved, the changes become part of the active version of the code list.
Multi-step	Multi-step approval workflows contain multiple approval tasks.	The first approver can approve the changes or send back the changes to the originator. If the first approver approves the changes, the next approver can approve the changes or send the changes back to the originator. The next approver can also approve the changes or send the changes back to the originator. After the task reaches the final approver, the final approver can approve the changes, reject the changes, or send the changes back to the originator. If the changes are approved, the changes become part of the active version of the code list.

Note: To enable multi-step workflows, configure the approval implementation mode by using APIs. For more information, see "Update approval implementation mode" on page 122.

The following image shows a sample workflow configuration:



Final Approval

By default, Business Stewards are final approvers for all workflows. A Business Steward can accept and publish the changes. A Business Steward might send back the edits for additional changes or reject the changes if they are not necessary.

If approved, the changes become part of the active version of the code list. You can view your changes, as well as previous changes, on the **History** tab of the code list.

Note: Users assigned both the Reference 360 Business Steward role and the Business Steward stakeholder role for an asset can send their proposed changes for approval or directly publish their changes without approval. For more information about roles, see "Users, groups, and roles" on page 43.

Related Topics:

- "Manage workflows" on page 103

Notifications

When users send an approval request, cancel an approval request, or perform a task action, the users responsible for the asset receive a notification. The notification alerts users that a change requires their

review, a review is no longer required, or that a task action was performed by another user. Notifications appear in the user's inbox and in an email notification.

The following table lists which users receive inbox notifications following task events:

Event	Requester	Assignees (Business Stewards)
User assigns a task	-	Yes
User approves, rejects, or sends back a task	Yes	-

The following table describes the email notifications users receive following task events:

Event	Requester	Assignees (Business Stewards)
User sends approval request	Notification confirms that they successfully sent the approval request.	Notification of approval request that requires their review.
User cancels approval request	Notification confirms that they successfully canceled the approval request.	Notification of canceled approval request that no longer requires their review.
User approves, rejects, or sends back a task	Notification of the action performed on the approval request.	The assignee who performed the action receives a notification that confirms their action. All other assignees receive a notification that another user performed an action on the approval request.

Note: Assignees include users assigned the Business Steward role for Reference 360 and users assigned the Business Steward stakeholder role for an asset.

If a Business Steward sends their changes for approval, they do not receive a notification to review their own changes. Other users assigned the Business Steward stakeholder role for the asset are notified of the task. If you are the only Business Steward assigned to the asset, you can approve your own changes. For more information about roles, see "Users, groups, and roles" on page 43.

Tasks

Tasks are requests to review and approve changes to code lists. A task is created when a user sends their draft code list for approval. Tasks are assigned to users who are responsible for reviewing and approving changes to the code list. A task is associated with a code list and helps manage changes to a code list.

Use the **Workflow Inbox** tab to view and take actions on tasks. You can also send your draft code lists for approval.

The **Workflow Inbox** tab consists of a **Task** panel and a **Task Details** panel. Use the **Task** panel to view all tasks available to you. In the **Task Details** panel, view the details of a task that you selected, compare the proposed code values against the published code values, and take action on the task.

You can view the following task details:

- Assignee
- Submit date
- Priority

- Due date
- Comments

Versions

As an asset moves through a workflow, different versions of the asset are created. The version that is available depends on where the asset is in the workflow.

When you view an asset, you are viewing the active version of the asset. To propose changes to an asset, you create a draft version of the asset. When you are done making changes, you send your draft for approval. If approved, the draft version becomes the active version. If rejected, the draft version is discarded and the active version of the asset is available.

When you view an asset that was sent for approval, you can see the changes pending approval. You can compare the pending changes with the active version. If you are the last reviewer responsible for approving the draft, you can approve the changes while viewing the pending changes.

The following table describes the available versions for an asset:

Version	Description	Available to
Active	Active version of the asset that contains approved reference data.	All users viewing an asset without proposed changes.
Draft	Draft version of the asset that contains proposed changes.	Users proposing changes to an asset.

If you are a reviewer, you can compare the draft version and active version to see moved, deleted, and added reference data. For more information, see "Comparisons" on page 31.

Comparisons

You can compare the draft version of a code list with the active version to compare proposed changes with the active version. You can also compare code lists and hierarchies to identify similarities and differences.

Compare changes

You can compare the draft version of a code list with the active version. You can view the changes to code values and evaluate whether the changes are correct. You can view a comparison when you work on a draft, send a draft for approval, or review an approval task.

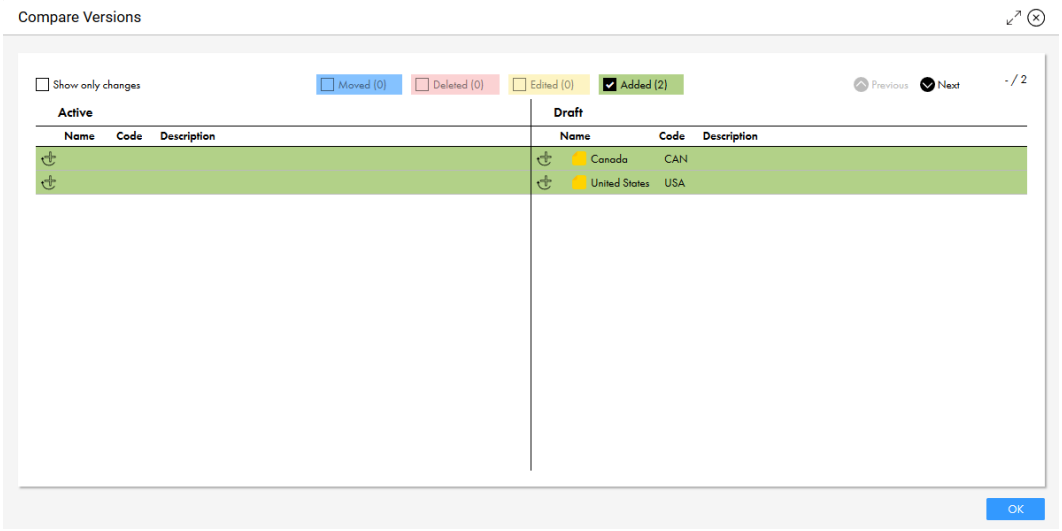
The **Compare Versions** view displays both unchanged and changed code values in a code list. When you edit or review a draft, you might want to view only changed code values. You can also filter to view added, edited, moved, and deleted code values only.

When you compare moved code values, you can navigate to the new location of the moved code value. You can navigate back to the original location to evaluate whether the move to the new location is correct.

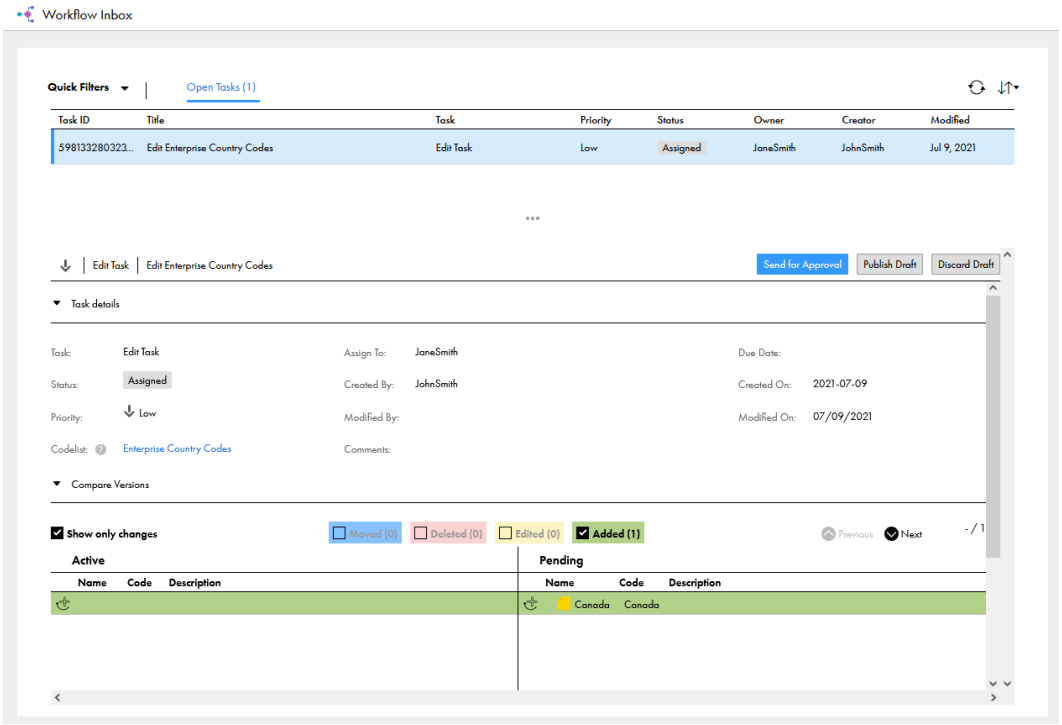
The following table lists the highlight used for each type of change:

Change	Highlight
Deleted code value	Red
Edited code value	Yellow and italicized
Moved code value	Blue
New code value	Green

The following image shows an example of the **Compare Versions** dialog box that appears when you compare your draft with the active version:



The following image shows the **Compare Versions** section of a task:



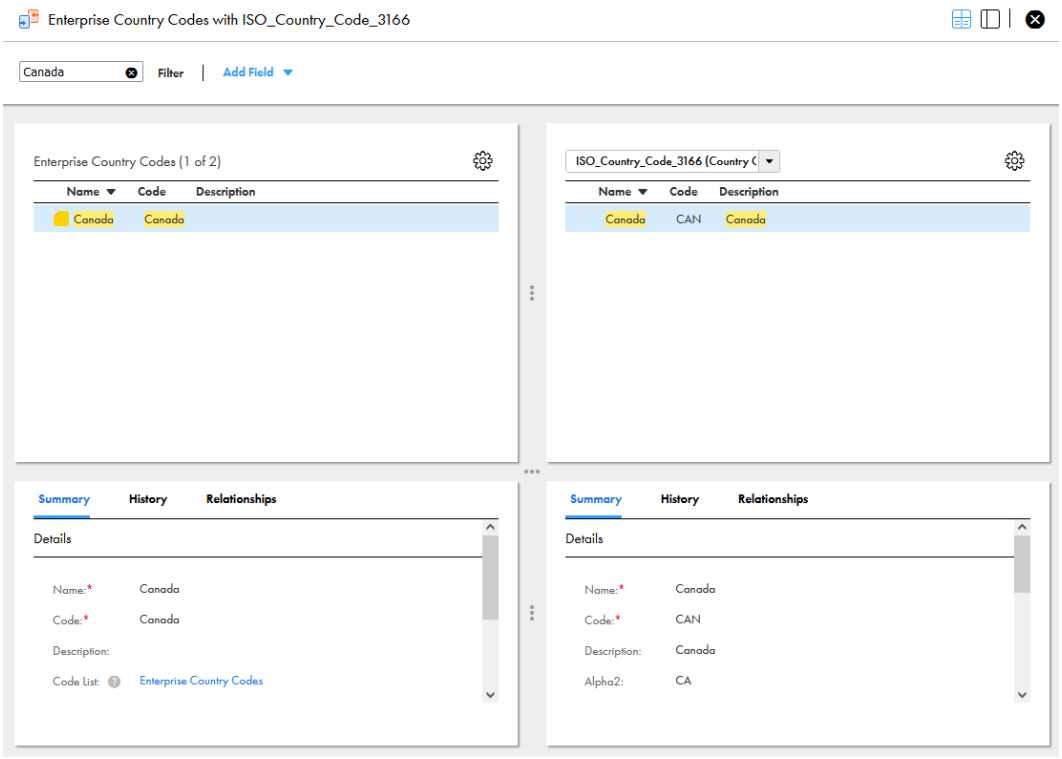
Compare lists

You can see a side-by-side view of code lists and hierarchies to identify similarities and differences between code values. You can choose to compare the same or different types of assets. For example, you might want to compare the code values in a code list with the code values in another code list, or the code values in a code list with the code values in a hierarchy.

To use the side-by-side view to display the information that you want to compare, perform the following tasks:

- To find specific code values, add filter criteria or enter search terms to find specific code values.
- To display the details of code values in each list, including summary, history, and relationship information, enable the detail view.
- To choose which attribute columns appear and rearrange the attribute columns, configure the column settings.

The following image shows a sample comparison between the Enterprise Country Codes and ISO Country Codes code lists and displays the details of the Canada code value in each code list:



History

The **History** tab displays a log of changes to an asset or code value. You can view historical information about a reference data set, a code list, a crosswalk, a hierarchy, or a code value.

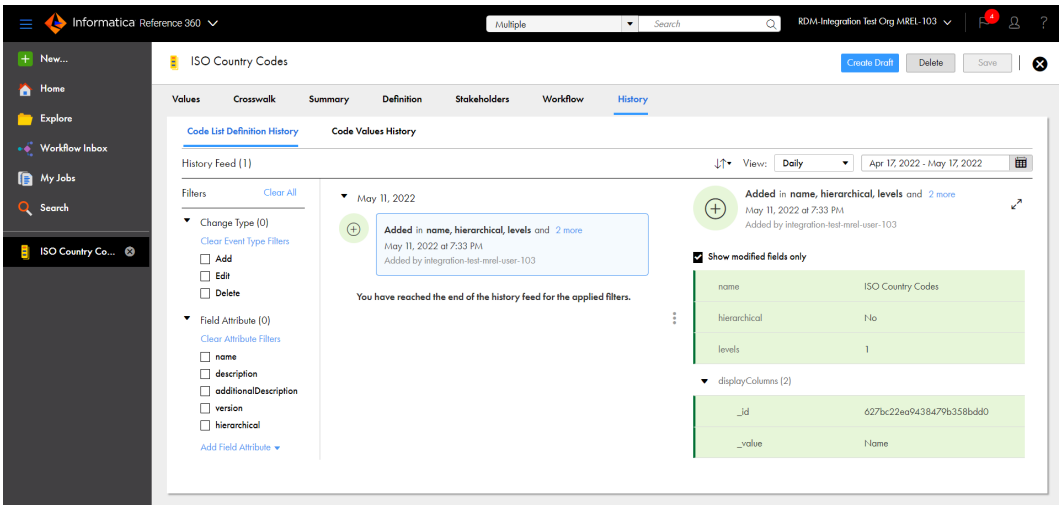
When you view the **History** tab, you can see the following information:

Asset	Historical information
Reference data set	Changes to properties, status, and definition.
Code list	Changes to properties, status, definition, and code values
Crosswalk	Changes to properties, status, definition, and value mappings.
Code value	Changes to the code value.

The **History** tab displays the following historical information:

- The field that was changed
- The value that was changed
- The new value
- The event that occurred
- The user who modified the value
- The date and time the value was modified
- The user who approved the changed value

The following image shows an example of the **History** tab for a code list:



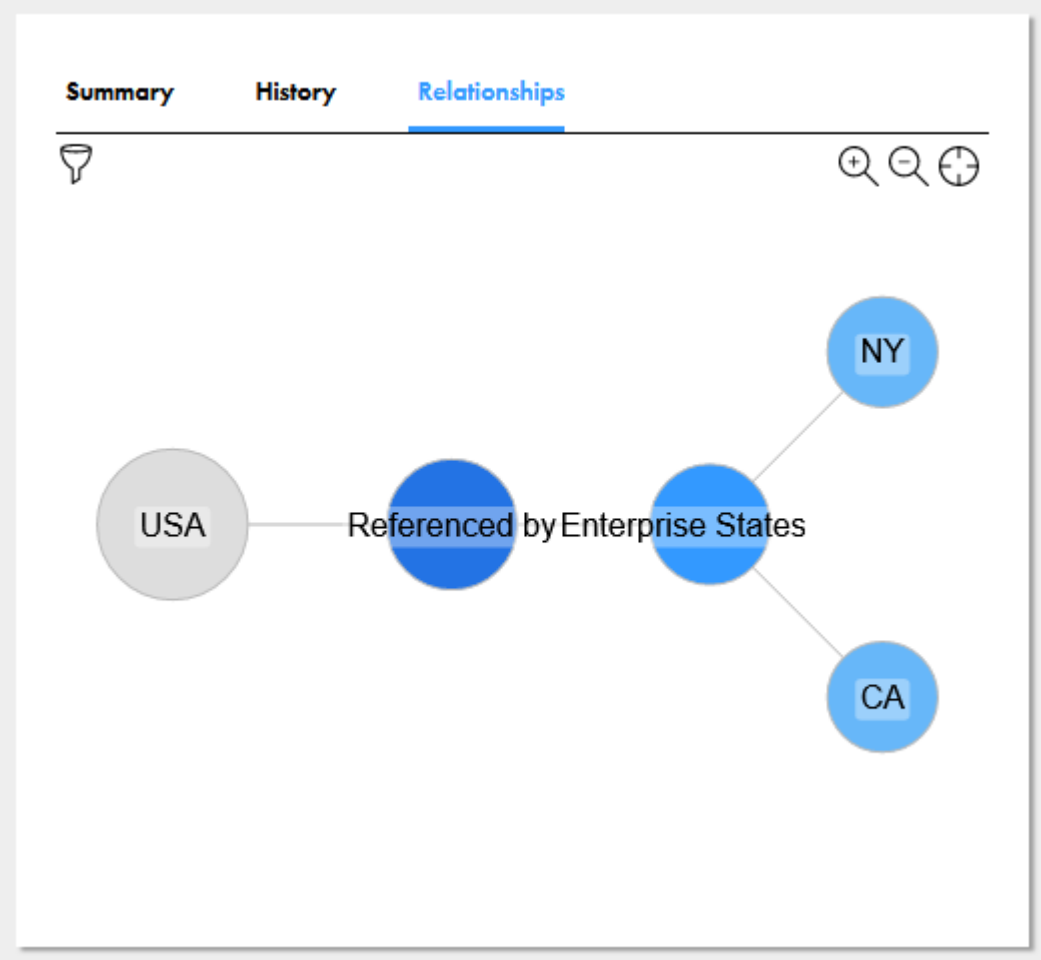
Relationships

You can view relationships between code values in code lists, crosswalks, and hierarchies. Use the **Relationships** tab to view the relationships for a code value in a graph.

The following table lists the relationships that the graph displays:

Asset Type	Relationships
Dependent code list	The code value that the selected code value is dependent on. The dependent code values of the selected code value.
Hierarchical code list	The parent code value of the selected code value. The child code values of the selected code value.
Crosswalk	The code value that the selected code value is mapped to. The code value that the selected code value is mapped from.
Hierarchy	The parent code value of the selected code value. The child code values of the selected code value.

The following image shows the relationships for the USA code value:



Rules

Create rules to ensure that code values meet your business standards. You can create rules for string, integer, and decimal attributes in code lists.

When users add or edit code values that do not meet your business standards, they receive inline validation errors.

For example, in the Country Codes code list, you might want the Code attribute to require a minimum of three characters. When users create code values in the code list, if they enter the USA value in the Codes attribute, then the value is valid. If they enter the US value, then the value is invalid and they receive inline validation errors.

For example, in the Health Insurers code list, you might have the Name, Code, and Third-Party Administrator attributes. You configure a concatenate rule for the Code attribute to populate the Code attribute with the values in the Name and Third-Party Administrator attributes. If the Name attribute value is Acme Inc. and the Third-Party Administrator attribute value is General Insurer, the value populated for the Code attribute is Acme Inc., General Insurer.

Warning: If you create or edit rules for a code list containing code values, some of the code values might become invalid. To find the invalid code values, edit each code value to trigger a validation check.

The following table lists the rules that you can configure for each attribute type:

Attribute Type	Rules
String	- Allowed Characters - Not Allowed Characters - Concatenate - Contains - Does Not Contain - Minimum Length - Maximum Length - Starts with - Ends with **Note:** For attributes with a concatenate rule, you can update the values that are used in the concatenated attribute value. However, if you configure a concatenate rule for the Code attribute, you cannot update the values in the concatenated attribute value.
Decimal	- Minimum Value - Maximum Value - Maximum Scale - Maximum Precision
Integer	- Minimum Value - Maximum Value
Date	- Start Date - End Date - Date Range

Note: Validation checks are not performed when you import code values.

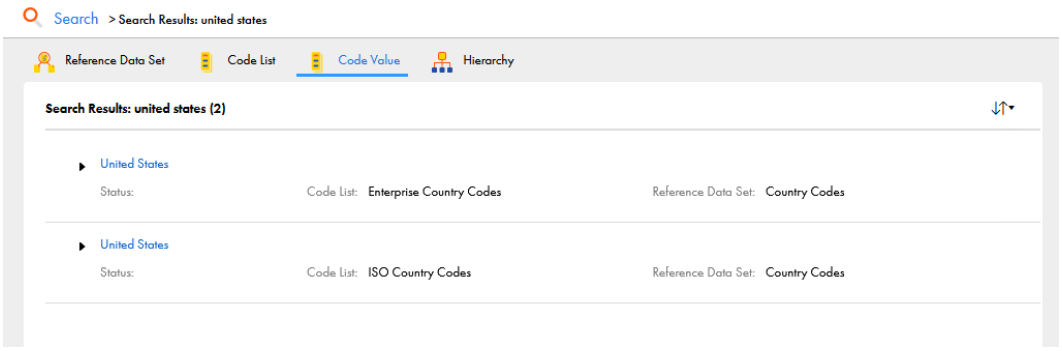
Search

You can search for reference data in your organization and filter the search results by reference data sets, code lists, code values, or hierarchies. The search results are grouped into tabs by asset type. The hierarchy details are not available for users with a Reference Data Management Basic Edition license.

Use the search box in the application header to perform a keyword search. Any reference data that match the keywords appears in the search results.

When code values appear in the search results, you can open the code lists and reference data sets that contain the code values from the search results.

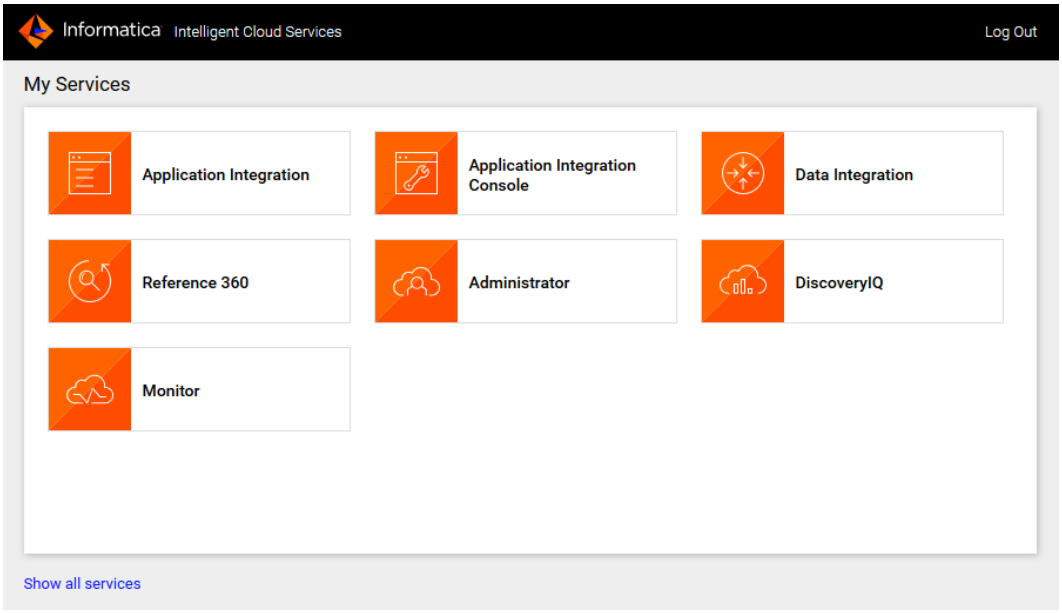
The following image shows an example of the **Search Results** page with the **Code Value** tab selected:



My Services page

When you log in to Informatica Intelligent Cloud Services, the **My Services** page displays the services that your organization is licensed to use and any common services that are available under the same license, such as Administrator. If your organization has trial licenses for additional services, the page also displays those services.

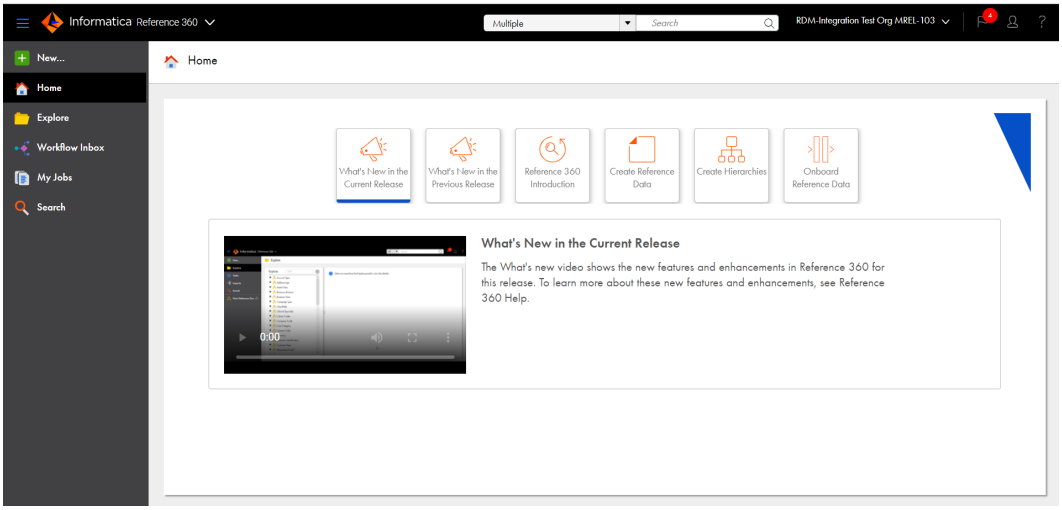
The following image shows an example **My Services** page:



To use Reference 360, click **Reference 360**.

Home page

After you log in to Reference 360, the **Home** page appears as shown in the following image:

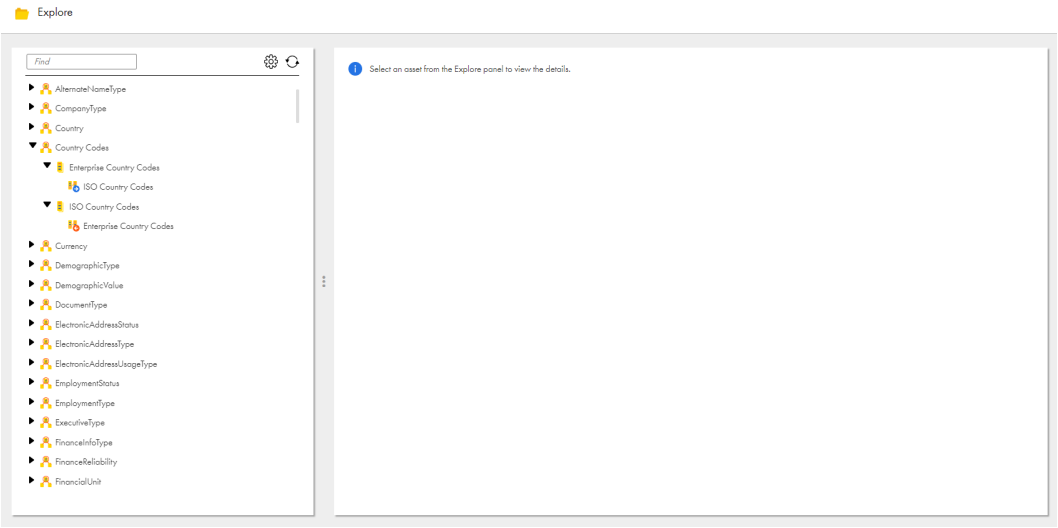


You can access the quick start videos on the **Home** page.

Explore page

Use the **Explore** page to find and work with your assets in Reference 360.

The following image shows an example **Explore** page:



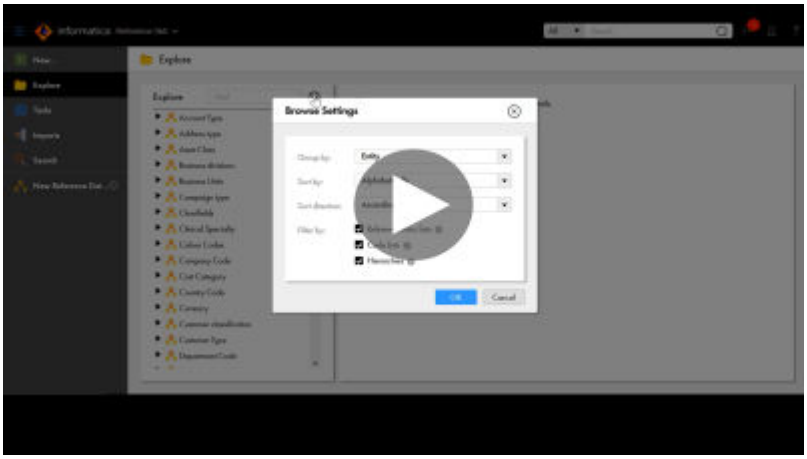
Finding assets on the Explore page

Use any of the following methods to find your assets on the **Explore** page:

- Group assets. View assets grouped by entity, status, domain, priority, and approver. To group assets, click **Browse Settings**, and then select the group by option.
- Sort assets. View assets alphabetically, by priority, or by status. You can also further sort by ascending or descending order. To sort assets, click **Browse Settings**, and then select the sort options.
- Filter the assets on the page. To view reference data sets, code lists, crosswalks, hierarchies, or all assets, click **Browse Settings**, and then select the filter by options.

  Note: The **Browse Settings** page does not display hierarchy assets for users with a Reference Data Management Basic Edition license.
- Search for assets. To search all assets in the organization, enter a name in the search box.

The following video shows you how to find assets on the Explore page:



Refreshing assets on the Explore page

To view the updated assets list in Reference 360, click the **Refresh** icon on the **Explore** panel.

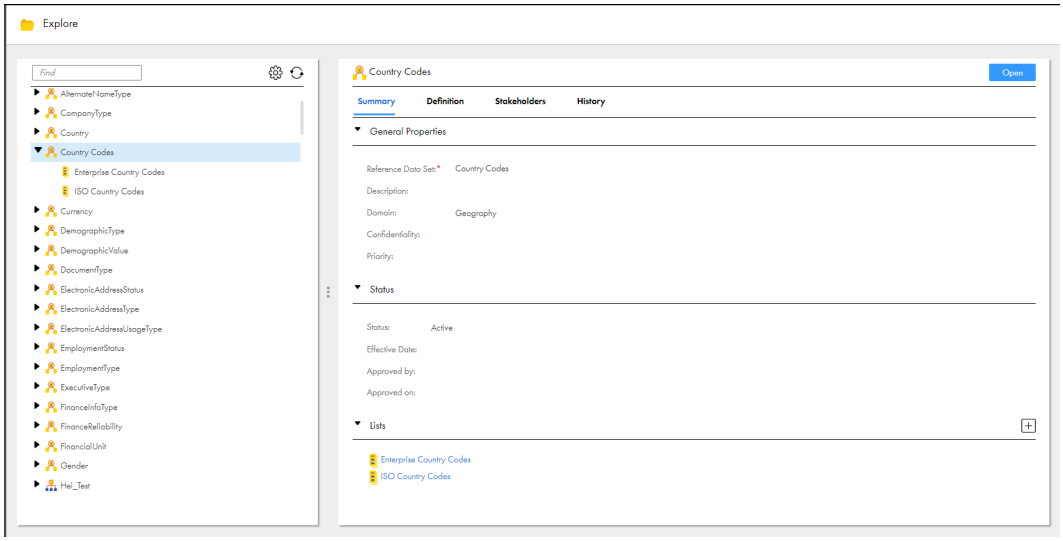
Note: The assets that other users created appear only after you refresh the **Explore** panel.

Working with assets on the Explore page

You can view and edit assets on the **Explore** page. The **Explore** panel displays all assets. The details panel displays the asset details, such as the summary, definition, stakeholders, and history. To edit an asset, select an asset in the **Explore** panel, and then click **Open** in the details panel.

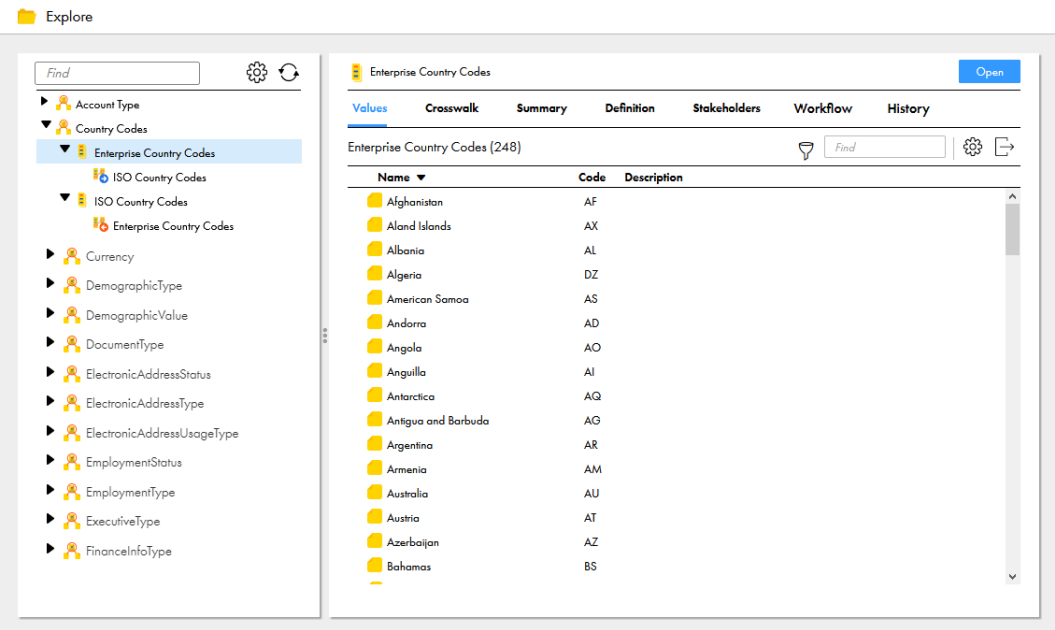
In the **Explore** panel, when you select a reference data set, you can view the associated code lists.

The following image shows a sample reference data set:



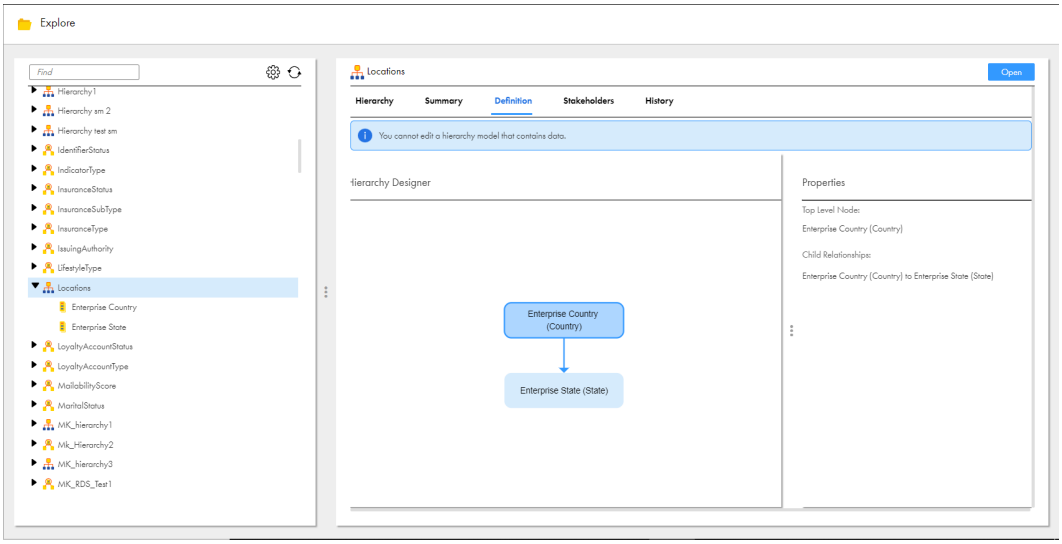
In the **Explore** panel, when you view a code list, you can see the associated outgoing and incoming crosswalks. In the details panel, you can view the code values in the code list and access all crosswalks associated to the code list.

The following image shows a sample code list:



In the **Explore** panel, when you select a hierarchy, you can see the code lists associated to the hierarchy. In the details panel, you can view the hierarchy model on the **Definition** tab and the code values in the hierarchy on the **Hierarchy** tab.

The following image shows a sample hierarchy:



Assets preserve the current view even when you navigate to another workspace or switch between assets in the **Explore** panel.

When you select a reference data set, code list, crosswalk, or hierarchy from the **Explore** panel, click the **History** tab, and then expand the **Explore** panel, the **History** tab displays the history feed similar to the history feed of code values. You can view the **History details** page by clicking the **Expand** icon. For more information, see "Viewing history of a code value" on page 76.

Note: The **Explore** panel retains its state when you navigate to the **Explore** page from a different workspace. The **Explore** panel resets to the default state when you perform the following actions:

- create or edit an asset to include the new asset or changes to the asset.

- delete an asset to exclude the asset.
- create a draft of a code list, which displays a lock icon against the draft code list.
- publish or discard the draft of a code list to save or discard the changes to the code list, which releases the lock icon that appears against the code list.

Custom branding

You can customize the logo and favicon that appear on your business application.

Use Administrator to configure custom branding for your parent organization and apply them to your sub-organizations. You can also configure custom branding for each sub-organization.

To change the branding, you must have the **Custom Logo & Color Themes** license. For more information about licenses, see Licenses in the Administrator help.

Use PNG, JPG, and GIF formats for the custom logo, and PNG format for the favicon.

For more information about configuring the custom branding, see Configuring custom branding for an organization in the Administrator help. For more information about the logo and favicon guidelines, see Logo and favicon guidelines in the Administrator help.

The branding settings do not apply to notifications, tooltips, custom components, and the following user interface screens:

- File Import
- My Jobs
- Reports
- Source Records

CHAPTER 2

Getting started with Reference 360

You can set up your organization in just a few steps.

Step 1. Create users or configure SAML

All Reference 360 users must have an Informatica Intelligent Cloud Services user account. Use the Administrator service to create Informatica Intelligent Cloud Services user accounts or configure SAML.

Step 2. Add system reference data values

System reference data contains a group of values that provide information about your reference data, such as the application, confidentiality, domain, priority, or status of an asset. You can add the values that you want to appear in the system reference data. For more information about adding values to the system reference data, see "Adding values to system reference data" on page 59 or "Add system reference data values" on page 236.

Note: To use workflows, you must add values to the Priority system reference data. For more information about workflows, see "Workflows" on page 27.

Users, groups, and roles

A user is an individual account in Informatica Intelligent Cloud Services. Reference 360 users must have a Informatica Intelligent Cloud Services user account. You create and manage users in the Administrator service.

A group is a collection of users who have something in common, such as working in the same team. Add users to groups to efficiently manage privileges for a collection of users.

A role is a collection of privileges that you assign to users and groups to allow access to Reference 360 and provide a collection of privileges in Reference 360.

Assign the following types of roles to provide privileges for Reference 360:

Reference 360 Roles

Reference 360 roles are pre-defined Informatica Intelligent Cloud Services roles that provide service-specific access to Reference 360 and global privileges in Reference 360. Use the Administrator service to assign Reference 360 roles.

For more information, see "Reference 360 roles" on page 44.

Stakeholder Roles

Stakeholder roles are pre-defined roles that provide asset-specific privileges in Reference 360. When you create assets in Reference 360, assign stakeholder roles to users and groups for each asset.

For more information, see "Stakeholder roles" on page 46.

For more information about Informatica Intelligent Cloud Services users, user groups, and roles, see the *Administrator* help.

Reference 360 roles

Reference 360 roles define a set of privileges that a user has while working in Reference 360. The privileges apply to all assets. To allow a user to access Reference 360, an administrator must assign at least one of the Reference 360 roles to the users or to their user group.

The following list describes the Reference 360 roles:

Reference 360 Planner

Planners create hierarchy assets, define hierarchy models, and import hierarchy relationships. Planners can delete hierarchies that are no longer needed. Planners can also assign the Planner stakeholder role to other users for a hierarchy.

Reference 360 Primary Owner

Primary Owners create and define reference data structures such as reference data sets and code lists. Primary Owners can delete code lists and propose changes to code values in code lists. The proposed changes must be approved by Business Stewards.

Reference 360 Business Steward

Business Stewards are subject matter experts for reference data. They create and manage code values in code lists and value mappings in crosswalks. Business Stewards are responsible for approving changes proposed by other users. Business Stewards can send their own changes for approval or directly publish their changes without approval.

Reference 360 Stakeholder

Stakeholders propose changes to code values. The proposed changes must be approved by Business Stewards.

Reference 360 Business Analyst

Business Analysts view and analyze assets. Business Analysts cannot propose changes to assets.

Reference 360 User

Users cannot access any assets. To work on assets, users require stakeholder roles for code lists and crosswalks.

The following table lists the privileges of the Reference 360 roles:

Function	Planner	Primary Owner	Business Steward	Stakeholder	Business Analyst	User
System reference data	Read	Create Read Update Delete	Create Read Update Delete	Read	Read	-
Reference data set	Read	Create Read Update Delete	Read	Read	Read	-
Reference data set structure definition	Read	Create Read	Read	Read	Read	-
Reference data set attributes	Read	Create Read Update Delete	Read	Read	Read	-
Code list	Read	Create Read Update Delete	Read	Read	Read	-
Code list structure definition	Read	Create Read	Read	Read	Read	-
Code list attributes	Read	Create Read Update Delete	Read	Read	Read	-
Code list draft	Read	Propose changes	Propose changes Approve changes Publish changes	Propose changes	-	-
Crosswalk	Read	Read	Create Read Update Delete	Read	Read	

Function	Planner	Primary Owner	Business Steward	Stakeholder	Business Analyst	User
Crosswalk value mappings	Read	Read	Create Read Update Delete	Read	Read	-
Hierarchy	Create Read Update Delete Assign stakeholders Import hierarchy relationships	Read	Read	Read	Read	-
Search and explore	Permitted	Permitted	Permitted	Permitted	Permitted	-
Export	Permitted	Permitted	Permitted	Permitted	Permitted	-
Import	-	Permitted	Permitted	Permitted	-	-
Direct import	-	-	Permitted	-	-	-

Note: After you create an asset, you might not be able to edit some definition settings. For more information, see "Reference data sets" on page 12 and "Code lists" on page 15.

For more information about Informatica Intelligent Cloud Services users, user groups, and roles, see the *Administrator* help.

Stakeholder roles

Stakeholder roles define a set of privileges for each asset in Reference 360. An asset might be a reference data set, code list, or crosswalk. Users with the Reference 360 Primary Owner role are responsible for assigning stakeholder roles to users and groups for each asset.

Note: When you create crosswalks, the stakeholders assigned to the associated reference data set and code lists are copied to the crosswalk.

The following list describes the stakeholder roles that you can assign to users and groups for an asset:

Planner

Planners define the hierarchy model and import hierarchy relationships to the hierarchy asset. Planners can also assign the Planner role to other users for the hierarchy.

Primary Owner

Primary Owners create and define the asset. Primary Owners can propose changes to the asset, but their changes must be approved by Business Stewards.

Business Steward

Business Stewards are subject matter experts for the asset. They create and manage data values in the asset. Business Stewards are responsible for approving changes proposed by other users. Business Stewards can send their own changes for approval or directly publish their changes without approval.

Stakeholder

Stakeholders propose changes to the asset. The proposed changes must be approved by Business Stewards.

Business Analyst

Business Analysts view and analyze the asset. Business Analysts cannot propose changes to the asset.

The following table lists the privileges that are assigned to users with a stakeholder role:

Function	Planner	Primary Owner	Business Steward	Stakeholder	Business Analyst
Reference data set	-	Read	Read	Read	Read
Reference data set structure definition	-	Read	Read	Read	Read
Reference data set attributes	-	Create Read Update Delete	Read	Read	Read
Code list	-	Read	Read	Read	Read
Code list structure definition	-	Read	Read	Read	Read
Code list attributes	-	Create Read Update Delete	Read	Read	Read
Code list draft	-	Propose changes	Propose changes Approve changes Publish changes	Propose changes	-
Crosswalk	-	Read	Read Update Delete	Read	Read
Crosswalk value mappings	-	Read	Create Read Update Delete	Read	Read

Function	Planner	Primary Owner	Business Steward	Stakeholder	Business Analyst
Hierarchy	Create Read Update Delete Assign stakeholders Import hierarchy relationships	Read	Read	Read	Read
Search and explore	-	Permitted	Permitted	Permitted	Permitted
Export	-	Permitted	Permitted	Permitted	Permitted
Import	-	Permitted	Permitted	Permitted	-
Direct import		-	-	-	-

Note: After you create an asset, you might not be able to edit some definition settings. For more information, see "Reference data sets" on page 12 and "Code lists" on page 15.

Note: The Planner stakeholder role is the only stakeholder role that you can assign to hierarchy assets.

Guidelines for assigning roles

To provide users with access to Reference 360, administrators must assign at least one Reference 360 role to users or groups to which the users belong. Then primary owners can assign stakeholder roles for assets to users. Role privileges are cumulative. Users who have multiple roles or who belong to groups receive the combined privileges associated with each role.

When administrators assign Reference 360 roles to users or groups, assign the role with the minimum number of privileges that are required for each user to work with Reference 360. Assign stakeholder roles only to those users or groups that work with an asset.

For example, you assign John the Reference 360 User role. He cannot view any assets in Reference 360. John is the subject matter expert for the Enterprise Language code list, so you assign John the Business Steward stakeholder role for the code list. The role allows him to create, manage, and approve code values in the Enterprise Language code list. He can also directly publish his changes to the code list. For all other assets, John's Reference 360 User role applies.

Guidelines for restricting access to assets

You can restrict a user's access to assets and only grant access to the assets that the user needs to view and work on. To restrict a user's access, assign the user the Reference 360 User role.

To grant privileges for only the assets that a user needs to view and work on, you assign the user a stakeholder role for the code list or crosswalk. The stakeholder role that you assign depends on the privileges that you want to grant the user.

When you grant privileges to code lists or crosswalks, the user gets access to the related assets.

Note: You must be assigned the Reference 360 Primary Owner role or the Primary Owner stakeholder role for the asset and the related assets to which you want to grant, change, or revoke access.

For example, assign Jane Smith the Reference 360 User role to restrict her access to assets. Then you assign Jane the Business Analyst stakeholder role for the Enterprise Language code list. The role allows her to view and analyze the Enterprise Language code list and the Languages reference data set.

The following table lists the assets for which you can assign stakeholders, and the assets and related assets to which users get access:

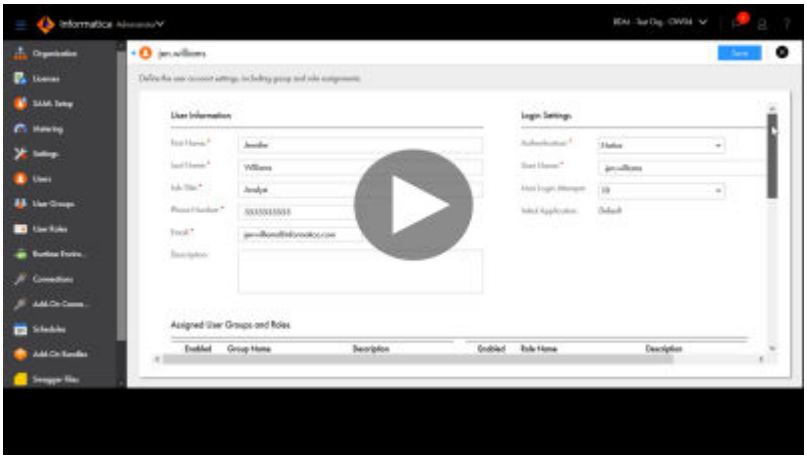
Stakeholder for	Access to
Code list	- The code list to which you granted access. - The reference data set to which the code list is associated.
Code list with a dependency	- The code list to which you granted access. - The reference data set to which the code list is associated. - The parent code list. - The reference data set to which the parent code list is associated.
Code lists with Reference Data attributes	- The code list to which you granted access. - The code lists referenced as Reference Data attributes. - The reference data sets to which the referenced code lists are associated.
Crosswalk	- The crosswalk to which you granted access. - The source and target code lists used in the crosswalk. - The reference data sets to which the code lists are associated.

Creating an Informatica Intelligent Cloud Services user

All Reference 360 users must have an Informatica Intelligent Cloud Services user account. You use the Administrator service to create user accounts and assign Reference 360 roles to users.

For more information about roles, see *User Administration* in the Administrator help.

The following video shows you how to create an Informatica Intelligent Cloud Services user:



1. In the service menu, click **Reference 360**.
2. In the **My Services** window, click **Administrator**.
3. Click **Users**.
4. Click **Add User**.
5. Enter the user information and login settings.

6. In the **Assigned User Groups and Roles** section, select the user groups and roles that you want to assign to the user.

   Assign Reference 360 roles to users to provide access privileges to Reference 360.
   For more information, see "Reference 360 roles" on page 44.

7. Click **Save**.

Creating a user group

Create a user group when multiple users in your organization need to perform the same tasks and need the same access rights for different types of assets. Group members can perform tasks and access assets based on the roles that you assign to the group. You use the Administrator service to create user groups and assign Reference 360 roles to groups.

1. In the service menu, click **Reference 360**.
2. In the **My Services** window, click **Administrator**.
3. Click **User Groups**.
4. Click **Add Group**.
5. Enter a group name and optional description.
6. In the **Assigned Roles** section, select the Reference 360 roles that you want to assign to the group.

   For more information, see "Reference 360 roles" on page 44.

7. To assign a user to the group, move the user from the **Available Users** list to the **Assigned Users** list.

   You can also assign a user to a group when you create or edit a user.

8. Click **Save**.

SAML single sign-on

You can enable single sign-on (SSO) capability so that users can access their organization without the need to enter login information. You can use SSO for user authentication or for both authentication and authorization in an organization. You configure SSO capability for an organization on the **SAML Setup** page.

Single sign-on to Informatica Intelligent Cloud Services is based on the Security Assertion Markup Language (SAML) 2.0 web browser single sign-on profile. The SAML web browser single sign-on profile consists of the following entities:

Identity provider

An entity that manages authentication information and provides authentication services through the use of security tokens.

Service provider

An entity that provides web services to principals, for example, an entity that hosts web applications. Informatica Intelligent Cloud Services is a service provider.

Principal

An end user who interacts through an HTTP user agent.

SAML 2.0 is an XML-based protocol that uses security tokens that contain assertions to pass information about a principal between an identity provider and a service provider. An assertion is a package of

information that supplies statements made by a SAML authority. You can find more information about SAML on the Oasis web site: https://www.oasis-open.org

The process that occurs when a user enters the Informatica Intelligent Cloud Services URL in a browser or launches Informatica Intelligent Cloud Services through a chicklet differs based on whether the organization uses SAML SSO for authentication only or for both authentication and authorization.

SAML single sign-on for authentication only

When a user signs on to Informatica Intelligent Cloud Services and the organization uses SAML SSO for user authentication only, the following process occurs:

1. Informatica Intelligent Cloud Services sends a SAML authentication request to the organization's identity provider.
2. The identity provider confirms the user's identity and sends a SAML authentication response to Informatica Intelligent Cloud Services. The authentication response includes a SAML token.
3. When Informatica Intelligent Cloud Services receives the SAML authentication response from the identity provider, it completes the following tasks:
   - If the user exists, Informatica Intelligent Cloud Services establishes the user session and logs the user in.
   - If the user does not exist and auto-provisioning of users is enabled, Informatica Intelligent Cloud Services gets the user attributes from the SAML token, creates the user, and assigns the user the default role and the default group, if it is configured. Informatica Intelligent Cloud Services establishes the user session and logs the user in.
   - If the user does not exist and auto-provisioning of users is disabled, Informatica Intelligent Cloud Services fails the login.
4. When a user logs out of Informatica Intelligent Cloud Services or the session times out, Informatica Intelligent Cloud Services sends a SAML logout request to the identity provider.
5. The identity provider terminates the user session on the identity provider side.

SAML single sign-on for authentication and authorization

When a user signs on to Informatica Intelligent Cloud Services and the organization uses SAML SSO for authentication and authorization, the following process occurs:

1. Informatica Intelligent Cloud Services sends a SAML authentication request to the organization's identity provider.
2. The identity provider confirms the user's identity and sends a SAML authentication response to Informatica Intelligent Cloud Services. The authentication response includes a SAML token.
3. When Informatica Intelligent Cloud Services receives the SAML authentication response from the identity provider, it completes the following tasks:
   - If the user exists, Informatica Intelligent Cloud Services gets the user roles, groups, and attributes from the SAML token. It finds the corresponding Informatica Intelligent Cloud Services user roles and groups, and updates the user roles, if necessary. Informatica Intelligent Cloud Services establishes the user session and logs the user in.
   - If the user does not exist and auto-provisioning of users is enabled, Informatica Intelligent Cloud Services gets the user roles, groups, and attributes from the SAML token and creates the user. Informatica Intelligent Cloud Services establishes the user session and logs the user in. If the token contains no SAML role or group information, Informatica Intelligent Cloud Services fails the login.
   - If the user does not exist and auto-provisioning of users is disabled, Informatica Intelligent Cloud Services fails the login.

4. When a user logs out of Informatica Intelligent Cloud Services or the session times out, Informatica Intelligent Cloud Services sends a SAML logout request to the identity provider.
5. The identity provider terminates the user session on the identity provider side.

SAML single sign-on requirements

To set up SAML single sign-on for an Informatica Intelligent Cloud Services organization, the system must use an appropriate identity provider. You must also have the appropriate license.

To set up SAML single sign-on for an organization, ensure that the following requirements are met:

- The system must use a SAML 2.0-based identity provider.

  Common identity providers include Microsoft Active Directory Federation Services (AD FS), Okta, SSOCircle, OpenLDAP, and Shibboleth. The identity provider must be configured to use either the DSA-SHA256 or RSA-SHA256 algorithm to generate the signature.
- The Informatica Intelligent Cloud Services organization must have the SAML based Single Sign-On license.
- You must have access to the organization as an organization administrator to set up single sign-on.

Single sign-on restrictions

There are some restrictions for SAML single sign-on access to Informatica Intelligent Cloud Services.

The following restrictions apply to SAML single sign-on access:

- If your license with the identity provider expires, you cannot access Informatica Intelligent Cloud Services through single sign-on.
- If the identity provider is down or Informatica Intelligent Cloud Services servers cannot reach it, users cannot log in to Informatica Intelligent Cloud Services through single sign-on.
- If the identity provider certificate used for SAML single sign-on to Informatica Intelligent Cloud Services expires, users cannot access Informatica Intelligent Cloud Services through single sign-on.
- If your organization uses trusted IP address ranges, users cannot log in to Informatica Intelligent Cloud Services from an IP address that is not within the trusted IP address ranges.

User management with SAML single sign-on

The following rules apply to users and user accounts when you enable SAML single-sign on for Informatica Intelligent Cloud Services:

- Informatica Intelligent Cloud Services stores user information that passes from the identity provider such as first name and email address in the Informatica Intelligent Cloud Services repository.
- You can create a regular user account with credentials in Informatica Intelligent Cloud Services after you enable an organization for single sign-on, and the user credentials are saved in the Informatica Intelligent Cloud Services repository. However, the user must log in to Informatica Intelligent Cloud Services directly instead of using single sign-on.
- If you delete a user from Informatica Intelligent Cloud Services, the user is deleted from the Informatica Intelligent Cloud Services repository. The user is not deleted from the identity provider.

SAML single sign-on configuration for Informatica Intelligent Cloud Services

Informatica Intelligent Cloud Services and your identity provider exchange configuration information when you set up single sign-on.

Informatica Intelligent Cloud Services requires identity provider metadata to send authentication and authorization requests to the identity provider. The identity provider requires service provider metadata from Informatica Intelligent Cloud Services to send responses to Informatica Intelligent Cloud Services.

SAML and Informatica Intelligent Cloud Services attributes need to be mapped so that Informatica Intelligent Cloud Services can consume the data passed in authentication responses. After you configure single sign-on settings in Informatica Intelligent Cloud Services, pass the Informatica Intelligent Cloud Services service provider metadata to your identity provider.

To configure single sign-on for Informatica Intelligent Cloud Services, complete the following tasks:

1. Configure the SAML identity provider and service provider settings, and map SAML attributes to Informatica Intelligent Cloud Services attributes in Informatica Intelligent Cloud Services.
2. Download the Informatica Intelligent Cloud Services service provider metadata from Informatica Intelligent Cloud Services, and deliver the metadata and the Informatica Intelligent Cloud Services single sign-on URL for your organization to your SAML identity provider administrator.

Configuring provider settings and mapping attributes

Configure SAML single sign-on settings and map SAML attributes on the **SAML Setup** page.

1. Log in to Informatica Intelligent Cloud Services as an organization administrator.
2. In Administrator, select **SAML Setup**.
3. On the **SAML Setup** page, configure the following properties:
   - SSO configuration properties
   - Identity provider configuration properties
   - Service provider settings
   - SAML attribute mapping properties
   - SAML role and group mapping properties (if you use SAML SSO for authentication and authorization)
4. Click **Save**.

   Informatica Intelligent Cloud Services generates the service provider metadata file. Informatica Intelligent Cloud Services also generates a unique token for your organization and saves the token to the Informatica Intelligent Cloud Services repository. The single sign-on URL for your organization includes the token. For example:

   ```
   https://dm-us.informaticacloud.com/ma/sso/<organization token>
   ```

After you save your changes on the **SAML Setup** page, download the service provider metadata, and send it to your identity provider along with the Informatica Intelligent Cloud Services single sign-on URL.

Identity provider configuration properties

Define identity provider configuration properties on the **SAML Setup** page.

The following table describes the identity provider configuration properties:

Property	Description
Issuer	The entity ID of the identity provider, which is the unique identifier of the identity provider. The Issuer value in all messages from the identity provider to Informatica Intelligent Cloud Services must match this value. For example: `<saml:Issuer>http://idp.example.com</saml:Issuer>`
Single Sign-On Service URL	The identity provider's HTTP-POST SAML binding URL for the SingleSignOnService, which is the SingleSignOnService element's location attribute. Informatica Intelligent Cloud Services sends login requests to this URL.
Single Logout Service URL	The identity provider's HTTP-POST SAML binding URL for the SingleLogoutService, which is the SingleLogoutService element's location attribute. Informatica Intelligent Cloud Services sends logout requests to this URL.
Signing Certificate	Base64-encoded PEM format identity provider certificate that Informatica Intelligent Cloud Services uses to validate signed SAML messages from the identity provider. **Note:** The identity provider signing algorithm must be either DSA-SHA1 or RSA-SHA1.
Use signing certificate for encryption	Uses the public key in your signing certificate to encrypt logout requests sent to your identity provider when a user logs out from Informatica Intelligent Cloud Services.
Encryption Certificate	Base64-encoded PEM format identity provider certificate that Informatica Intelligent Cloud Services uses to encrypt SAML messages sent to the identity provider. Applicable if you do not enable use of the signing certificate for encryption.
Name Identifier Format	The format of the name identifier in the authentication request that the identity provider returns to Informatica Intelligent Cloud Services. Informatica Intelligent Cloud Services uses the name identifier value as the Informatica Intelligent Cloud Services user name. The name identifier cannot be a transient value that can be different for each login. For a particular user, each single sign-on login to Informatica Intelligent Cloud Services must contain the same name identifier value. To specify that the name identifier is an email address, the Name Identifier Format is as follows: `urn:oasis:names:tc:SAML:1.1:nameidformat:emailAddress`
Logout Service URL (SOAP Binding)	The identity provider's SAML SOAP binding URL for the single logout service. Informatica Intelligent Cloud Services sends logout requests to this URL.
Logout Page URL	The landing page to which a user is redirected after the user logs out of Informatica Intelligent Cloud Services. Informatica Intelligent Cloud Services redirects the logged out user to the landing page in the following ways: - If you specify a logout page URL, Informatica Intelligent Cloud Services redirects the user to this URL after logout. - If you do not specify a logout page URL, Informatica Intelligent Cloud Services redirects the user to a default logout page.

Service provider settings

Define the Informatica Intelligent Cloud Services service provider settings on the **SAML Setup** page.

The following table describes service provider settings:

Property	Description
Informatica Cloud Platform SSO	Displays the single sign-on URL for your organization. This URL is automatically generated by Informatica Intelligent Cloud Services.
Clock Skew	Specifies the maximum permitted time, in seconds, between the time stamps in the SAML response from the identity provider and the Informatica Intelligent Cloud Services clock. Default is 180 seconds (3 minutes).
Name Identifier value represents user's email address	If enabled, Informatica Intelligent Cloud Services uses the name identifier as the email address. Default is enabled.
Sign authentication requests	If enabled, Informatica Intelligent Cloud Services signs authentication requests to the identity provider. Default is enabled.
Sign logout requests sent using SOAP binding	If enabled, Informatica Intelligent Cloud Services signs logout requests sent to the identity provider. Default is enabled.
Encrypt name identifier in logout requests	If enabled, Informatica Intelligent Cloud Services encrypts the name identifier in logout requests. **Note:** Verify that the identity provider supports decryption of name identifiers before you enable this option. Default is disabled.

SAML attribute mapping properties

User login attributes such as name, email address, and user role are included in the authentication response from the identity provider to Informatica Intelligent Cloud Services. If the identity provider passes user and group information using SCIM 2.0, the authentication response includes additional SCIM attributes such as Display Name, Employee Number, and Organization.

Map the Informatica Intelligent Cloud Services user fields to corresponding SAML attributes on the **SAML Setup** page.

Note: The attribute format differs based on your identity provider. Refer to the provider documentation for more information.

The following table describes the SAML attribute mapping properties:

Property	Description
Use friendly SAML attribute names	If selected, uses the human-readable form of the SAML attribute name which might be useful in cases in which the attribute name is complex or opaque, such as an OID or a UUID.
First Name	SAML attribute used to pass the user first name.

Property	Description
Last Name	SAML attribute used to pass the user last name.
Job Title	SAML attribute used to pass the user job title.
Email Addresses	SAML attribute used to pass the user email addresses. This property must be mapped.
Emails Delimiter	Delimiter to separate the email addresses if multiple email addresses are passed.
Phone Number	SAML attribute used to pass the user phone number.
Time Zone	SAML attribute used to pass the user time zone.
User Roles	SAML attribute used to pass the assigned user roles. This field is enabled when the **Map SAML Groups and Roles** option is enabled.
Roles Delimiter	Delimiter to separate the roles if multiple roles are passed. This field is enabled when the **Map SAML Groups and Roles** option is enabled.
User Groups	SAML attribute used to pass the assigned user groups. This field is enabled when the **Map SAML Groups and Roles** option is enabled.
Groups Delimiter	Delimiter to separate the groups if multiple groups are passed. This field is enabled when the **Map SAML Groups and Roles** option is enabled.

The following table describes the additional attributes. These attributes are visible when the **Enable IdP to push users/groups using SCIM 2.0** option is enabled:

Property	Description
Display Name	SCIM attribute used to pass the user displayName.
Employee Number	SCIM attribute used to pass the enterprise user employeeNumber.
Organization	SCIM attribute used to pass the enterprise user organization.
Department	SCIM attribute used to pass the enterprise user department.
Street Address	SCIM attribute used to pass the user streetAddress.
Locality	SCIM attribute used to pass the user locality.
Region	SCIM attribute used to pass the user region.
Post Code	SCIM attribute used to pass the user postalCode.
Country	SCIM attribute used to pass the user country.
Locale	SCIM attribute used to pass the user locale.
Preferred Language	SCIM attribute used to pass the user preferredLanguage.

Property	Description
ID	SCIM attribute used to pass the user id.
External ID	SCIM attribute used to pass the user externalId. For Azure Active Directory, this is the objectID. For Okta, it is the id.

SAML role and group mapping properties

When you use SAML for authentication only, define a default role and optional default user group for new users. When you use SAML for authentication and authorization, map SAML role and group names to Informatica Intelligent Cloud Services role names. You can map multiple SAML roles and groups to a single Informatica Intelligent Cloud Services role.

Define the SAML role and group mapping properties on the **SAML Setup** page.

The following table describes SAML role mapping properties:

Property	Description
Informatica Intelligent Cloud Services role	The SAML role equivalent for the Informatica Intelligent Cloud Services role. If you need to enter more than one role, use a comma to separate the roles. The role mapping fields are enabled when the **Map SAML Groups and Roles** option is enabled.
Default Role	Default user role for single sign-on users. When auto-provisioning is enabled, new users are assigned this role the first time they sign on to Informatica Intelligent Cloud Services. This field is visible when the **Map SAML Groups and Roles** option is disabled.
Default User Group	Optional, default user group for single sign-on users. When auto-provisioning is enabled, new users are assigned to this user group the first time they sign on to Informatica Intelligent Cloud Services. This field is visible when the **Map SAML Groups and Roles** option is disabled.

The following table describes SAML group mapping properties:

Property	Description
Informatica Intelligent Cloud Services role	The SAML group equivalent for the Informatica Intelligent Cloud Services role. If you need to enter more than one group, use a comma to separate the groups. You can enter up to 4000 characters. The role mapping fields are enabled when the **Map SAML Groups and Roles** option is enabled.
Default Role	Default user role for single sign-on users. When auto-provisioning is enabled, new users are assigned this role the first time they sign on to Informatica Intelligent Cloud Services. This field is visible when the **Map SAML Groups and Roles** option is disabled.
Default User Group	Optional, default user group for single sign-on users. When auto-provisioning is enabled, new users are assigned to this user group the first time they sign on to Informatica Intelligent Cloud Services. This field is visible when the **Map SAML Groups and Roles** option is disabled.

Downloading the service provider metadata

The identity provider requires the SAML service provider metadata and Informatica Intelligent Cloud Services URL to complete the SAML single sign-on setup process. After Informatica Intelligent Cloud Services generates the service provider metadata file, deliver the file and the Informatica Intelligent Cloud Services URL to the identity provider.

1. On the **SAML Setup** page, click **Download Service Provider Metadata**.

   The service provider metadata file is downloaded to your machine.
2. In the **Information** dialog box, note the URL for single sign-on access to your Informatica Intelligent Cloud Services organization.
3. Click **OK** to close the **Information** dialog box.
4. Send the metadata file and the Informatica Intelligent Cloud Services single sign-on URL to your identity provider administrator.

CHAPTER 3

Manage system reference data

System reference data contains a group of values that provide information about your reference data.

You can configure system reference data, such as application, confidentiality, domain, priority, and status. You can then use the system reference data when you define other reference data assets.

To manage the system reference data, users must be assigned one of the following roles:

- Reference 360 Administrator
- Reference 360 Primary Owner
- Reference 360 Business Steward
- MDM Designer

Related Topics:

- "System reference data" on page 11

Adding values to system reference data

You can add values to the system reference data. You can then use these values when you define reference data assets.

1. In the **Explore** panel, select **System Reference Data** and click **Open**.

   The system reference data opens in a new tab.
2. Click the system reference data for which you want to add a value.
3. Click the **Add** icon.

   A blank row appears.
4. To add a value, click the new row and perform the following actions:
   a. In the **Name** field, enter a name.
   b. In the **Key** field, enter a unique identifier for the value.

      You can't change the key for the values after you save the values.
5. Click the **Save** icon.

Editing values of system reference data

You can edit your system reference data values.

1. In the **Explore** panel, select **System Reference Data** and click **Open**.

   The system reference data opens in a new tab.
2. Click the system reference data for which you want to edit a value.
3. Click the **Edit** icon.
4. Click the value that you want to edit.
5. Edit the name as required.

   Note: You can't modify the key for the values.
6. Click the **Save** icon.

Deleting values of system reference data

You can delete the system reference data values that you no longer need.

1. In the **Explore** panel, select **System Reference Data**, and click **Open**.

   The system reference data opens in a new tab.
2. Click the system reference data for which you want to delete a value.
3. Hover over the value that you want to delete, and then click the **Delete** icon.

   A confirmation dialog box appears.
4. Click **Delete**.

   The value is deleted for all the assets.

CHAPTER 4

Manage reference data sets

All reference data in Reference 360 is categorized under reference data sets. You create a reference data set to contain a category of reference data, such as country codes or currency codes. In a reference data set, you create code lists. Code lists contain code values from an application.

For example, you create a Country Codes reference data set to categorize the country codes reference data in your organization. In the reference data set, you create a Sales Country Codes code list and a Marketing Country Codes code list. In the Sales Country Codes code list, you create or import country code values used in your organization's sales application. In the Marketing Country Codes code list, you create or import country code values used in your organization's marketing application.

When you create a reference data set, you define the definition and attributes. Later, when you create code lists, the code lists inherit the structure definition and attributes of the reference data set.

If you no longer need a reference data set, you can delete the reference data set.

For more information, see "Reference data sets" on page 12.

Creating reference data sets

Create reference data sets to categorize the reference data in your organization. Reference data sets contain code lists, and code lists contain code values.

To create a reference data set, perform the following actions:

1. Create a reference data set.
2. Define a reference data set.
3. Assign stakeholders.
4. View the history of a reference data set.

Related Topics:

- "Reference data sets" on page 12

Step 1. Create a reference data set

Create a reference data set, and configure its general properties and status.

1. Click **New** and select **Reference Data Set**.

   The **New Reference Data Set** page appears and displays the **Summary** tab.

2. In the **Reference Data Set** field, enter a name.
3. Optionally, complete the general properties fields.

   Note: The domain, confidentiality, and priority are inherited by code lists in the reference data set.

Field	Description
Description	A description of the reference data set.
Domain	An area or grouping to describe the code values.
Confidentiality	The confidentiality level of the reference data set and its code lists and crosswalks. The confidentiality levels are confidential, internal, restricted, and public.
Priority	The priority of the asset. The priority levels are critical, high, medium, and low.
Status	The state of the asset in the life cycle.
Effective Date	The date from when the status is effective.
Approved by	The user who approved the asset.
Approved on	The date of approval.

Note: If you do not see options in some lists, configure your system reference data values. For more information, see "Adding values to system reference data" on page 59.

Step 2. Define a reference data set

Define the structure, attributes, and display settings of a reference data set. When you create code lists, the code lists inherit the structure definition and attributes from the reference data set.

Important: After you create the reference data set, some actions are restricted to prevent issues with the definition of the crosswalk, such as modifying the structure definition or creating additional required attributes. For more information, see "Reference data sets" on page 12.

1. Click **Definition**.

   The **Definition** tab opens.
2. Optionally, configure the structure definition of the reference data set.

Option	Description
Hierarchical	A hierarchical reference data set supports hierarchical data structures and passes on its hierarchical support to its code lists. Code lists that support hierarchies allow you to arrange their code values into levels. For more information, see "Hierarchical reference data sets" on page 13.
Dependent	A dependent reference data set depends on the code values in another reference data set and passes on its dependency to its code lists. For more information, see "Dependent reference data sets" on page 14.

3. Optionally, select the attributes that you want to display by default.
   a. Click **Pencil**.
   b. Select **Required**.
4. Optionally, configure a custom attribute for the reference data set.
   a. Click **Add**.

      An attribute row appears in the **Attributes** section. In the **Attribute Details** panel, you can configure the details of the attribute.
   b. In the **Attribute Name** field, enter a name.
   c. From the **Type** list, select the data type of the attribute.

      For more information, see "Attributes" on page 22.
   d. To make the attribute required, select **Required**.

   Note: You can set a new attribute or an existing attribute as required for the existing reference data sets. You cannot set the Name and Code attributes as optional as they are required by default.
5. When you set a new attribute or modify the existing attribute as required, a warning message appears.
6. In the **Display Settings** section, select the attribute that you want displayed in Reference 360 to represent code values in the reference data set.

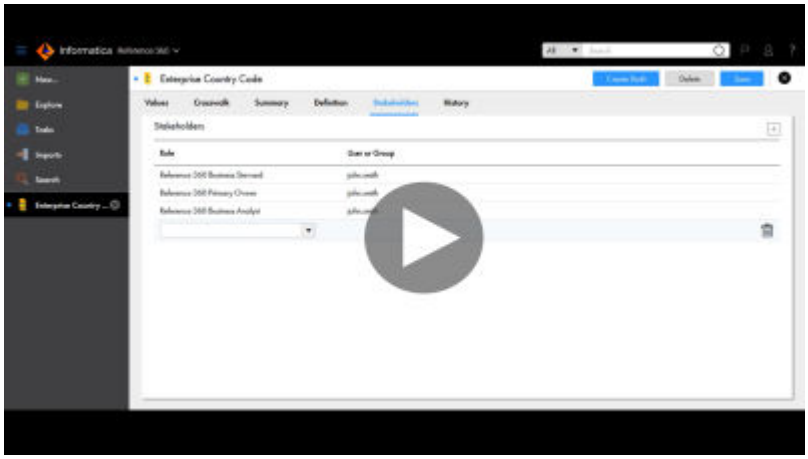
Step 3. Assign stakeholders

Assign stakeholder roles to users who play a role in using, creating, or maintaining the asset. Stakeholder roles determine a user's privileges for the asset.

When a user manages an asset as a stakeholder, they have the combined privileges provided by their stakeholder role and their Reference 360 role. For more information about stakeholder roles, see "Stakeholder roles" on page 46. For more information about assigning roles, see "Guidelines for assigning roles" on page 48.

Note: When you create crosswalks, the stakeholders assigned to the associated reference data set and code lists are copied to the crosswalk.

The following video shows you how to assign stakeholders:



1. Click **Stakeholders**.

   The **Stakeholder** tab opens.

2. Click **Add**.

   A list appears in an empty row.

3. In the **Role** list, select a stakeholder role.

   The **New Stakeholder** dialog box appears.

4. Select a user name and click **Add**.

   The user name of the stakeholder appears in the row.

5. Click **Save**.

Step 4. View the history of a reference data set

You can view the changes made to a reference data set in a chronological order. Use the historical data to track the changes made to the reference data set at any point in time.

1. In the **Explore** panel, select a reference data set and click **Open**.

   The reference data set opens in a new tab.

2. To view the change history of the reference data set, click the **History** tab.

   The change history of the reference data set appears.
   The following table lists the different user interface elements that you can find on the **History** tab:

User Interface Element	Description
Filters panel	Lists the predefined filters. You can select the values based on which you want to filter the history feed.
Add Field Attribute	Displays additional fields that you can select to filter the history feed.
Sort	Displays the following values based on which you can sort the history feed: - New to Old. Sorts the changes in reverse chronological order. - Old to New. Sorts the changes in chronological order.
View	Groups the history feed based on the selected frequency. Use one of the following frequencies: - Daily. Displays the changes for each day. - Monthly. Displays the changes for the entire month. - Yearly. Displays the changes for the entire year.
Calendar icon	Allows you to configure a time period. You can view the history feed for the specified time period. Use the following predefined time periods: - Today. Displays the changes occurred on the current date. - Last 7 days. Displays the changes occurred in the last seven days. - Last 30 days. Displays the changes occurred in the last 30 days.

3. To view all the fields in the history feed, clear **Show modified fields only**.

   By default, the **Show modified fields only** check box is selected, and the modified fields appear in the history feed.

Deleting reference data sets

Users with the Reference 360 Primary Owner role can delete reference data sets that are no longer needed.

You cannot delete reference data sets that contain code lists or are referenced by other assets. For more information, see “Considerations for deleting reference data sets” on page 65.

1. Open the reference data set that you want to delete.
2. Click **Delete**.
3. Click **OK**.

Considerations for deleting reference data sets

You can delete reference data sets that you no longer need. You must be assigned the Reference 360 Primary Owner role to delete reference data sets.

You cannot delete the following types of reference data sets:

- Reference data sets that contain code lists
- Reference data sets that are specified as a dependent of another asset
- Reference data sets that are specified as a Reference Data attribute for another asset

Note: Users with the Primary Owner stakeholder role cannot delete reference data sets. For more information, see “Users, groups, and roles” on page 43 .

CHAPTER 5

Manage code lists

Code lists contain a set of code values that you create or import from a source application. You create code lists in reference data sets. A code list inherits its structure definition and attributes from the reference data set.

Later, you can create crosswalks to map code values in a pair of code lists. Crosswalks provide a way to translate between the different code values each application uses for a business term.

You can create validation rules for attributes in code lists to ensure that code values meet your business standards. For example, in the Country Codes code list, you might want the Code attribute for code values to require a minimum of three characters.

If you no longer need a code list, you can delete the code list. When you delete a code list, the code list and its code values are removed from Reference 360. You cannot delete a code list that is used in a picklist field of a business entity in a business application, such as Customer 360.

For more information about code lists, see "Code lists" on page 15.

Creating code lists

You create code lists in reference data sets. A code list inherits its structure definition and attributes from the reference data set. You can define additional attributes or display columns, and assign stakeholders to a code list that are different from the reference data set.

Later, you create or import a set of code values from a source system into the code list.

For more information, see "Code lists" on page 15.

To create a code list, perform the following actions:

1. Create a code list.
2. Define a code list.
3. Assign stakeholders.
4. View the history of a code list.

Related Topics:

- "Code lists" on page 15

Step 1. Create a code list

Create a code list in a reference data set, and configure the general properties, external URLs, notification recipients, and status of the code list.

Before you begin, you must create a reference data set.

1. Click **New** and select **Code List**.

   The **New Code List** page appears and displays the **Summary** tab.
2. Select a reference data set to associate with this code list.
3. In the **Name** field, enter a name.
4. Optionally, complete the general properties fields.

   Note: Code lists inherit the domain, confidentiality, and priority from the reference data set.

Field	Description
Version	The version information of the code list.
Description	A description of the code list. The description cannot exceed 4000 characters.
Additional Description	Additional description of the code list. You can add additional details of a code list in the Additional Description field.
Application	The source application of the code values.
Domain	An area or grouping to describe the code values.
Confidentiality	The confidentiality level of the code list inherited from the reference data set. The confidentiality levels are confidential, internal, restricted, and public.
Priority	The priority of the asset. The priority levels are critical, high, medium, and low.
Status	The state of the asset in the life cycle.
Effective Date	The date from when the status is effective.
Approved by	The user who approved the asset.
Approved on	The date of approval.

   If you do not see options in some lists, configure your system reference data values. For more information, see "Adding values to system reference data" on page 59.

   Note: The **Summary** tab of the code lists displays only the Reference Data Set, Name, and Description fields for users with a Reference Data Management Basic Edition license.
5. In the **External URLs** section, click **Add External URL**.

   A blank row appears.
6. To add an external URL, enter a URL name and a link of an external page for an externally mandated code list.

The added external URL appears in the row.

7. To remove an external URL, hover over a row, and click **Delete External URL**.
8. In the **Notification Recipients** section, click **Add Notification Recipient** to add external stakeholders to be notified about changes to a code list.

   A blank row appears.
9. To add a notification recipient, enter a name, contact type, such as email or SMS text message, and the respective contact detail.

   The added notification recipient appears in the row.
10. To remove a notification recipient, hover over a row, and click **Delete Notification Recipient**.

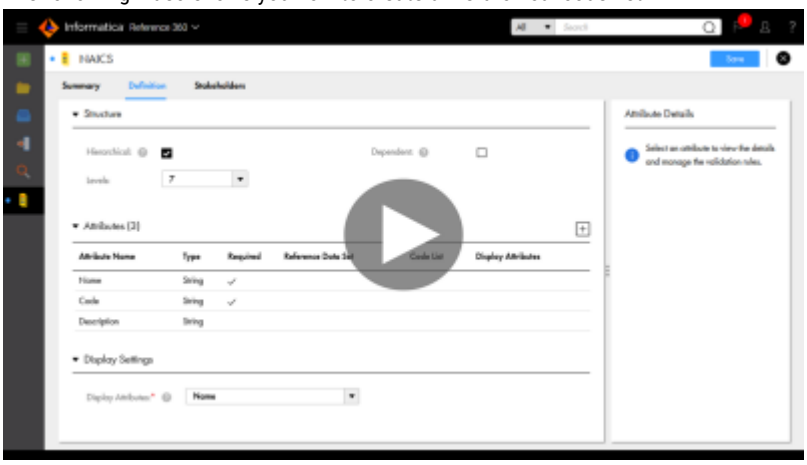
Step 2. Define a code list

Define the structure, attributes, and display settings of the code list. Code lists inherit their structure and attributes from the reference data set.

Important: After you create the code list, some actions are restricted to prevent issues with the definition of the code list, such as modifying the structure definition or creating additional required attributes. For more information, see "Code lists" on page 15.

1. Click **Definition**.

   The **Definition** tab opens.
2. If the code list did not inherit a structure from the reference data set, configure the structure definition.

Option	Description
Hierarchical	A hierarchical code list allows you to arrange code values into levels to create hierarchies. The following video shows you how to create a hierarchical code list:  For more information, see "Hierarchical code lists" on page 17.
Dependent	A dependent code list depends on code values in a code list that belongs to a different reference data set. For more information, see "Dependent code lists" on page 18.

3. Optionally, configure a custom attribute for the code list.
   a. Click **Add**.

      An attribute row appears in the **Attributes** section. In the **New Attribute** panel, you can configure the details of the attribute.

   b. In the **Attribute Name** field, enter a name.

   c. From the **Type** list, select the data type of the attribute.

      For more information, see "Attributes" on page 22.

      Note: You can define scale for the Decimal data type attributes. The supported scale value ranges from 1 to 34. Default is 4.

   d. To make the attribute required, select **Required**.

   e. To add a validation rule for a string, integer, or decimal attribute type, click **Add Rule** and configure a validation rule.

      The following table lists the validation rules that you can configure for each attribute type:

Attribute Type	Rules
String	- Allowed Characters - Not Allowed Characters - Concatenate - Contains - Does Not Contain - Minimum Length - Maximum Length - Starts with - Ends with **Note:** For attributes with a concatenate rule, you can update the values that are used in the concatenated attribute value. However, if you configure a concatenate rule for the Code attribute, you cannot update the values in the concatenated attribute value.
Decimal	- Minimum Value - Maximum Value - Maximum Scale - Maximum Precision
Integer	- Minimum Value - Maximum Value
Date	- Start Date - End Date - Date Range

      You can set a new attribute or an existing attribute as required for the existing code list. You cannot set the Name and Code attributes as optional as they are required by default.

   Note: Users with a Reference Data Management Basic Edition license cannot edit the existing attributes, add new attributes, or add rules to the attributes of the code list.

4. When you set a new attribute or modify the existing attribute as required, a warning message appears.
5. Optionally, in the **Display Settings** section, select the attribute that you want displayed in Reference 360 to represent code values in the code list.

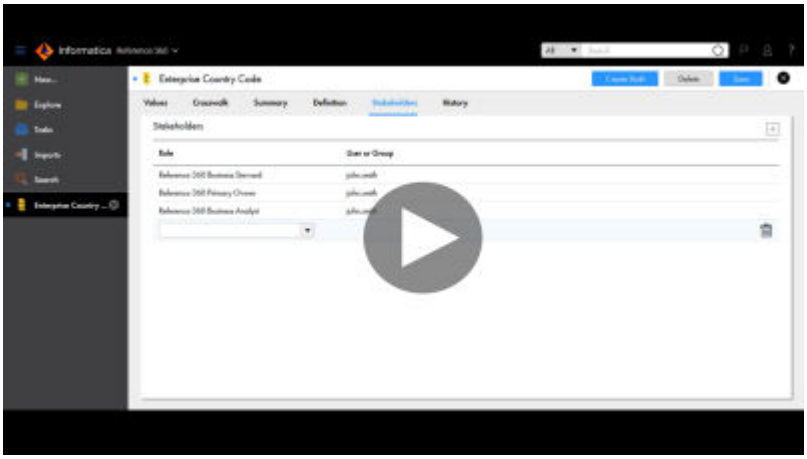
Step 3. Assign stakeholders

Assign stakeholder roles to users who play a role in using, creating, or maintaining the asset. Stakeholder roles determine a user's privileges for the asset.

When a user manages an asset as a stakeholder, they have the combined privileges provided by their stakeholder role and their Reference 360 role. For more information about stakeholder roles, see "Stakeholder roles" on page 46. For more information about assigning roles, see "Guidelines for assigning roles" on page 48.

Note: When you create crosswalks, the stakeholders assigned to the associated reference data set and code lists are copied to the crosswalk.

The following video shows you how to assign stakeholders:



1. Click **Stakeholders**.

   The **Stakeholder** tab opens.
2. Click **Add**.

   A list appears in an empty row.
3. In the **Role** list, select a stakeholder role.

   The **New Stakeholder** dialog box appears.
4. Select a user name and click **Add**.

   The user name of the stakeholder appears in the row.
5. Click **Save**.

Step 4. View the history of a code list

You can view the changes made to a code list in a chronological order. Use the historical data to track the changes made to the code list at any point in time.

1. In the **Explore** panel, select a code list and click **Open**.

   The code list opens in a new tab.
2. Click the **History** tab.

   By default, the **Code List Definition History** tab opens. The **Code List Definition History** tab displays the change history of the code list properties.

The following table lists the different user interface elements that you can find on the **Code List Definition History** tab of the **History** tab:

User Interface Element	Description
Filters panel	Lists the predefined filters. You can select the values based on which you want to filter the history feed.
Add Field Attribute	Displays additional fields that you can select to filter the history feed.
Sort	Displays the following values based on which you can sort the history feed: - New to Old. Sorts the changes in reverse chronological order. - Old to New. Sorts the changes in chronological order.
View	Groups the history feed based on the selected frequency. Use one of the following frequencies: - Daily. Displays the changes for each day. - Monthly. Displays the changes for the entire month. - Yearly. Displays the changes for the entire year.
Calendar icon	Allows you to configure a time period. You can view the history feed for the specified time period. Use the following predefined time periods: - Today. Displays the changes occurred on the current date. - Last 7 days. Displays the changes occurred in the last seven days. - Last 30 days. Displays the changes occurred in the last 30 days.

3. To view all the fields in the history feed, clear **Show modified fields only**.

   By default, the **Show modified fields only** check box is selected, and the modified fields appear in the history feed.

4. To view the change history of code values of the code list in a table, click the **Code Values History** tab.

Deleting code lists

Users with the Reference 360 Primary Owner role can delete code lists that are no longer needed.

You cannot delete code lists that are locked or are referenced by other assets. For more information, see "Considerations for deleting code lists" on page 72.

Note: Users with a Reference Data Management Basic Edition license cannot delete default code lists.

1. Open the code list that you want to delete.
2. Click **Delete**.
3. Click **OK**.

Considerations for deleting code lists

You can delete code lists that you no longer need. You must be assigned the Reference 360 Primary Owner role to delete code lists. When you delete a code list, the code list and its code values are removed from Reference 360.

You cannot delete the following types of code lists:

- Code lists that are locked
- Code lists that are specified as a dependent of another asset
- Code lists that are specified as a Reference Data attribute for another asset
- Code lists that are part of a value mapping in a crosswalk

Note: Users with the Primary Owner stakeholder role cannot delete code lists. For more information, see "Users, groups, and roles" on page 43.

CHAPTER 6

Manage code values

A code value is a unique value, such as a business term, code, or lookup value. Code values are organized into code lists. Each code list contains code values from a single source application or industry standard list. You can add code values or import code values into a code list.

You can add or edit code values in a code list by locking the code list or without locking the code list. When you lock the code list, you restrict other users to make concurrent edits to the code list. When you don't lock the code list and edit a code value, other users can edit other code values in the same code list. The edited code value is locked for other users until you publish or discard the draft code list or an approver approves the updated code list.

Later, you can create crosswalks to map code values in a pair of code lists. Crosswalks provide a way to translate between the different code values each application uses for a business term.

For more information, see "Code values" on page 19.

To add code values, perform one of the following actions:

"Adding code values" on page 73

Chapter 8, "Import data" on page 85

Related Topics:

- "Code values" on page 19

Adding code values

Add code values in a code list by locking the code list or without locking the code list.

1. In the **Explore** panel, select a code list and click **Open**.

   The code list opens on a new tab.
2. To restrict other users from editing the code list, click **Lock**.

   A draft version of the code list is created, and the code list is locked for other users.
3. Click **Add Code Value at the Root Level**.

   The **New Code Value** dialog box appears.
4. In the **Name** field, enter a name.
5. In the **Code** field, enter a code value.

When you add a code value, if the reference data type attributes of a code list contain more than 100 values, type a few characters and select one of the matching values. If the reference data type attributes contain less than 100 values, select the value from the list.

Note: When you search for a reference data attribute value, type a few characters and then press **Enter** or click the **Search** icon to display relevant matching values.

6. Optionally, complete the remaining code value fields.

Field	Description
Description	A description of the code value.
Status	The state of the code value in the life cycle. **Note:** If you don't see options in the list, configure your system reference data values. For more information, see "Adding values to system reference data" on page 59.
Effective Date	The date the status is effective from.

7. Click **Save and New** to add additional code values, or click **Save**.

   The new code values appear in the code list. If you skipped step 2, a draft version of the code list is created now.

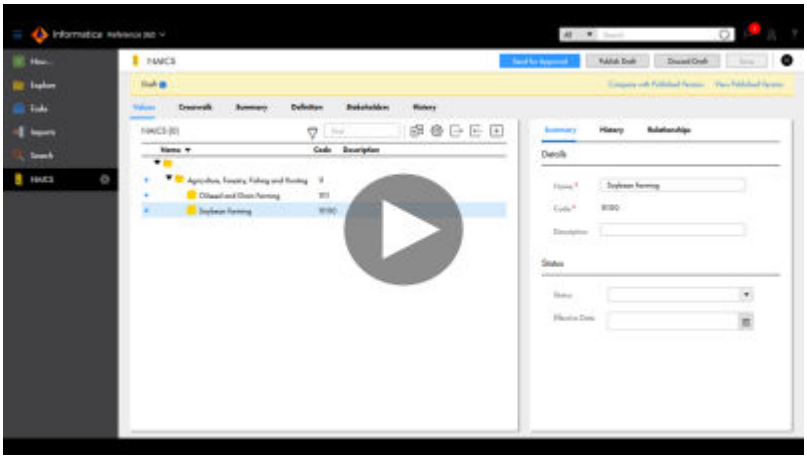
   Note: Users with a Reference Data Management Basic Edition license can't send the proposed changes for approval.

If you don't have privileges to publish your proposed changes, send your changes for approval. If you are a Business Steward, you can publish your proposed changes without approval or send your changes for approval. For more information, see "Users, groups, and roles" on page 43.

Creating child code values

Create child code values in a hierarchical code list. For example, you might want to add the Electric Power Distribution code value as a child code value of the Utilities code value.

The following video shows you how to create child code values in a hierarchical code list:



1. In the **Explore** panel, select a code list and click **Open**.

   The code list opens on a new tab.
2. Click **Lock**.

   A draft version of the code list is created.
3. Hover over the code value that you want to create a child code value under, and then click **New**.

   The **New Code Value** dialog box appears.
4. To copy all the values from the parent code value, perform the following actions:
   a. Click **Copy Values from Parent Code Value**.
   b. Edit the values that are different from the parent code value.
5. To enter values for the child code value, perform the following actions:
   a. In the **Name** field, enter a name.
   b. In the **Code** field, enter a code value.
   c. Optionally, complete the remaining code value fields.
6. To move code values, drag a code value to a new location. Click **OK**.
7. Create additional child code values.
8. Click **Save**.

   The new code values appear in the code list.

   Note: You can't add a child code value to multiple parent code values in hierarchical code lists.

Editing code values

Edit code values in a code list by locking the code list or collaborating with other users without locking the code list.

1. In the **Explore** panel, select a code list and click **Open**.

   The code list opens on a new tab.
2. To restrict other users from editing the code list, click **Lock**.

   A draft version of the code list is created, and the code list is locked for other users.
3. On the **Values** tab, click the code value that you want to edit.

   The code value details panel displays the details of the code value in different tabs.
4. On the Summary tab, click the **Edit** icon.
5. Edit the details as required.
6. Click the **Save** icon.

   When you don't lock the code list, the edited code value appears locked for other users and can't be edited by other users.

   Note: Users with a Reference Data Management Basic Edition license can't send the proposed changes for approval.

If you don't have privileges to publish your proposed changes, send your changes for approval. If you are a Business Steward, you can publish your proposed changes without approval or send your changes for approval. For more information, see "Users, groups, and roles" on page 43.

Viewing history of a code value

You can view the changes made to a code value of a code list in a chronological order. Use the historical data to track the changes made to the code value at any point in time.

1. In the **Explore** panel, select a code list and click **Open**.

   The code list opens in a new tab.
2. On the **Values** tab, select a code value.

   The code value details panel displays the details of the code value in different tabs.
3. To view the change history of the code value, click the **History** tab.

   The change history of the code value appears.
4. On the **History** tab, click the **Expand** icon.

   The **History Details** page appears.

The following table lists the different user interface elements that you can find on the **History Details** page:

User Interface Element	Description
Filters panel	Lists the predefined filters. You can select the values based on which you want to filter the history feed.
Add Field Attribute	Displays additional fields that you can select to filter the history feed.
Sort	Displays the following values based on which you can sort the history feed: - New to Old. Sorts the changes in reverse chronological order. - Old to New. Sorts the changes in chronological order.
View	Groups the history feed based on the selected frequency. Use one of the following frequencies: - Daily. Displays the changes for each day. - Monthly. Displays the changes for the entire month. - Yearly. Displays the changes for the entire year.
Calendar icon	Allows you to configure a time period. You can view the history feed for the specified time period. Use the following predefined time periods: - Today. Displays the changes occurred on the current date. - Last 7 days. Displays the changes occurred in the last seven days. - Last 30 days. Displays the changes occurred in the last 30 days.

5. To view the change details, click each of the following tabs:
   - Business Entity Fields. Displays the changes to the fields of code value.
     To view all the fields in the history feed, clear **Show modified fields only**.

     By default, the **Show modified fields only** check box is selected, and the modified fields appear in the history feed.
   - Task Details. Displays the task details when the record is associated to an approval workflow.
     - **Change List ID**. A system-generated unique identifier to track the changes made to the code values of the code list.
     - **Requested By**. The user who submitted the changes for approval.
     - **Requested On**. The date when the changes are sent for approval.
     - **Approved By**. The user who approved the changes.
     - **Approved On**. The date of approval.
6. Click **Close**.

Exporting code values

Export code values in a code list to a CSV file. You can configure the format of the data and the attributes that you want to include in the exported file.

1. Open a code list.

2. Click **Export Values**.

   The **Export** dialog box appears with a warning message indicating that you check the formatting of the exported file.
3. On the **General** tab, enter a file name.
4. To export the parent code value for each code value in the hierarchy, select **Include hierarchy**.

   The **Include hierarchy** check box is only available for hierarchical code lists.
5. Optionally, configure how you want the data to be formatted in the exported file.

Field	Description
Delimiter	Character used to separate values.
Date Format	Format used for dates.
Decimal Mark	Decimal separator used for numbers.
Thousands Separator	Grouping separator used for numbers.

6. Optionally, click **Columns**, and then select the attribute columns that you want in the exported file. For example, you might want to only include the Name and Code attribute columns for code values.
7. Click **Export**, and then save or open the exported file.

Deleting code values

You can propose to delete code values that you no longer need.

1. In the **Explore** panel, select a code list and click **Open**.

   The code list opens on a new tab.
2. Click **Lock**.

   A draft version of the code list is created.
3. Hover over the code values that you want to delete, and then click **Delete**.
4. Optionally, delete additional code values.

When you are done deleting code values, send your changes for approval.

Note: Users assigned both the Reference 360 Business Steward role and the Business Steward stakeholder role for an asset can send their proposed changes for approval or directly publish their changes without approval. For more information about roles, see "Users, groups, and roles" on page 43.

Considerations for deleting code values

You can delete code values that you no longer need. When you delete a code value, the code value is permanently deleted from Reference 360.

You cannot delete the following types of code values:

- Code values that are defined as a parent code value in a hierarchical code list
- Code values that are used as a Reference Data attribute for another code list

- Code values that are used as a dependency in another code list
- Code values that are part of a value mapping in a crosswalk
- Code values that are part of a hierarchy asset

CHAPTER 7

Create crosswalks

A crosswalk is a visual representation of a one-way relationship between code values in a pair of code lists. A reference data set can contain many code lists, and each code list contains a variation of the same type of code values. Crosswalks provide a way to translate between the different variations each code list uses.

For more information, see "Crosswalks" on page 19.

To create a crosswalk, perform the following actions:

1. Create a crosswalk.
2. Create value mappings.
3. Assign stakeholders.
4. View the history of a crosswalk.

Related Topics:

- "Crosswalks" on page 19

Step 1. Create a crosswalk

Create a crosswalk to map code values in a source code list to code values in a target code list from same or different reference data sets. Value mappings provide a way to translate code values in one code list to code values in another code list.

1. Click **New** and select **Crosswalk**.

   The **New Crosswalk** window appears.
2. In the **Source Reference Data Set** field, select the reference data set that contains the code list you want to map.
3. In the **From Code List** field, select the source code list.

   The crosswalk is associated with the source code list.
4. In the **Target Reference Data Set** field, select the reference data set that contains the code list you want to map to the selected source code list.
5. In the **To Code List** field, select the target code list

   The target code list contains code values that you want to map.
6. Click **OK**.

A new crosswalk is created. The crosswalk name displays the source code list and target code list. If the target reference data set is different from the source reference data set, the crosswalk name displays the name of the target reference data set adjacent to the target code list.
In the **Explore** panel, you can view the crosswalk name appended with the target reference data set name when the source and target reference data sets are different.

Note: Users with a Reference Data Management Basic Edition license can create crosswalks only between the default code list and another code list of the same reference data set.

7. Click the **Summary** tab.
8. Optionally, enter a description.

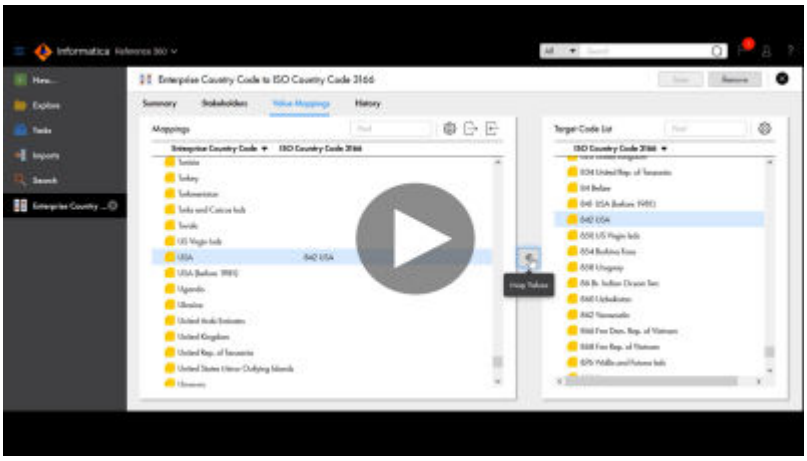
   Note: You cannot configure the general properties. A crosswalk inherits its general properties from the source and target reference data sets.
9. Optionally, complete the status fields.

Field	Description
Status	The state of the asset in the life cycle. **Note:** If you do not see options in the list, configure your system reference data values. For more information, see "Adding values to system reference data" on page 59.
Effective Date	The date from when the status is effective.
Approved by	The user who approved the asset.
Approved on	The date of approval.

Step 2. Create value mappings

Map code values in one code list to code values in another code list to indicate that the code values represent the same business term.

The following video shows you how to create value mappings:



1. Click **Value Mapping**.

   The **Value Mapping** tab opens and displays the **Mappings** panel and **Target Code List** panel.
2. To create a value mapping, perform one of the following actions:
   - In the **Mappings** panel, select a source value that you want to map. In the **Target Code List** panel, select the equivalent code value, and click **Map Values**.
   - Drag a code value from the **Target Code List** panel onto the equivalent code value in the **Mappings** panel.

   The mapping appears in the **Mappings** panel.
3. To create additional value mappings, repeat 2.
4. To remove a value mapping, perform the following actions:
   a. Hover over a value mapping, and click the **Remove** icon.

      The **Remove Value Mappings** dialog box displays the list of available mappings for the specific value mapping.

   b. Select the mappings that you want to remove, and click **Remove**.

      To select all the mappings, click **Mappings**.

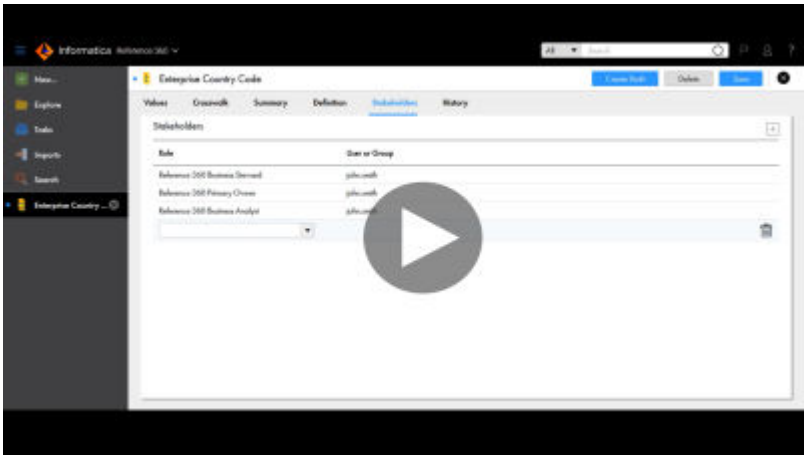
Step 3. Assign stakeholders

Assign stakeholder roles to users who play a role in using, creating, or maintaining the asset. Stakeholder roles determine a user's privileges for the asset.

When a user manages an asset as a stakeholder, they have the combined privileges provided by their stakeholder role and their Reference 360 role. For more information about stakeholder roles, see "Stakeholder roles" on page 46. For more information about assigning roles, see "Guidelines for assigning roles" on page 48.

Note: When you create crosswalks, the stakeholders assigned to the associated reference data set and code lists are copied to the crosswalk.

The following video shows you how to assign stakeholders:



1. Click **Stakeholders**.

   The **Stakeholder** tab opens.
2. Click **Add**.

   A list appears in an empty row.
3. In the **Role** list, select a stakeholder role.

   The **New Stakeholder** dialog box appears.
4. Select a user name and click **Add**.

   The user name of the stakeholder appears in the row.
5. Click **Save**.

Step 4. View history of a crosswalk

You can view the changes made to a crosswalk in a chronological order. Use the historical data to track the changes made to the crosswalk at any point in time.

1. In the **Explore** panel, select a crosswalk and click **Open**.

   The crosswalk opens in a new tab.
2. Click the **History** tab.

   By default, the **Crosswalk Definition History** tab opens. The **Crosswalk Definition History** tab displays the change history of the crosswalk properties.

The following table lists the different user interface elements that you can find on the **Crosswalk Definition History** tab of the **History** tab:

User Interface Element	Description
Filters panel	Lists the predefined filters. You can select the values based on which you want to filter the history feed.
Add Field Attribute	Displays additional fields that you can select to filter the history feed.
Sort	Displays the following values based on which you can sort the history feed: - New to Old. Sorts the changes in reverse chronological order. - Old to New. Sorts the changes in chronological order.
View	Groups the history feed based on the selected frequency. Use one of the following frequencies: - Daily. Displays the changes for each day. - Monthly. Displays the changes for the entire month. - Yearly. Displays the changes for the entire year.
Calendar icon	Allows you to configure a time period. You can view the history feed for the specified time period. Use the following predefined time periods: - Today. Displays the changes occurred on the current date. - Last 7 days. Displays the changes occurred in the last seven days. - Last 30 days. Displays the changes occurred in the last 30 days.

3. To view all the fields in the history feed, clear **Show modified fields only**.

   By default, the **Show modified fields only** check box is selected, and the modified fields appear in the history feed.

4. To view the change history of value mappings of the crosswalk in a table, click the **Value Mapping History** tab.

CHAPTER 8

Import data

You can import code lists and crosswalks from a CSV file into Reference 360. You can import a file of up to 20 MB at a time.

The import process parses the import data and automatically maps the columns of the source file with the target fields. The automatically mapped fields display confidence indicators to indicate the accuracy of the mapping. For the fields with low confidence scores or incorrect field mappings, you can manually map the source columns with the required target fields.

You can preview and verify the field mappings before you import the file.

Import process

Your user role must have permission to import data.

To import data, perform the following tasks:

1. Upload a CSV file to import.
2. Verify the automatically mapped fields. Manually map the fields that have incorrect mapping to ensure that the columns of the source file match the target fields.
3. Preview and verify the field mapping and import the file.

Step 1. Upload a file

To import data into Reference 360, upload a CSV file with code values or value mappings.

Review your CSV file and note from which line data in the CSV file starts. In the **Import Data Starting From Line** field, enter the line number from where you want to import data.

Ensure that the CSV file does not contain the following special characters: / \ * ? % : | " < > and any columns with the headers named as 'Key'.

1. From the **Explore** page, search and open the code list or crosswalk.
2. Click the **Import Values** icon.

   The **Upload File** page of the **Import Values** wizard appears.
3. To select a file to upload, drag a CSV source file, or click **Browse** and select the source file.

   The first 10 rows of records appear in the preview section.

   Note: You can't add a child code value to multiple parent code values in hierarchical code lists.

4. In the **File Import Settings** section, verify the system suggested settings based on your data, and update any incorrect settings.

   The following table describes the available import settings:

Field	Description
Delimiter	A character that represents the break between data values in the import file. Select a predefined delimiter or select **Other** to define a custom delimiter.
Text Qualifier	Symbols used in the file to enclose a string.
Encoding Type	Unicode encoding type.
Import Data Starting From Line	Line number in the file from where you want to import data. Use this field to exclude headers. Default is 2.
Contains column headers	Indicates whether the source file includes column headers.
Column Header Row	Header row number in the file.
Regional settings	Indicates whether you want to configure date and time formats.
Date Pattern	Format of the date fields in the file.
Date Time Pattern	Format of the date and time fields in the file. The supported time zone is in UTC (Coordinated Universal Time).
Decimal Separator	A character used as a decimal separator. Default is period (.).
Thousand Separator	A character used as a thousand separator. Default is comma (,).

5. Click **Next**.

   The **Map Fields** page appears.

Step 2. Map fields

After you select the file to import, the import process automatically maps the possible columns to the suitable target fields. You can either use the system suggested mapping or modify the mapping as required.

1. On the **Map Fields** page of the **Import Values** wizard, verify the system suggested mapping of fields.
2. To manually map any field, perform one of the following tasks:
   - Drag a source column to a target field.
   - Select a source column and a corresponding target field, and click **Map Selected**.

   The import process maps the field accordingly.
3. Verify the mapping of all fields and ensure that all the required fields are mapped.
4. Click **Next**.

   The **Preview and Import File** page appears.

Step 3. Preview and import data

After you map the fields, use the preview section to review and verify the mapping, and then import the data.

1. On the **Preview and Import File** page of the **Import Values** wizard, verify the mapping.

   If the preview does not return desired results, you can go back to the **Map Fields** page to map the fields again.

2. To import the file, click **Import**.

   The import process starts. You get a notification after the import process is complete.

You can view the status of the import on the **My Jobs** page. If the import job fails, click the **Download** icon in the **Status** column of the failed job to view the error report. For more information, see "My jobs page" on page 112.

Field mapping

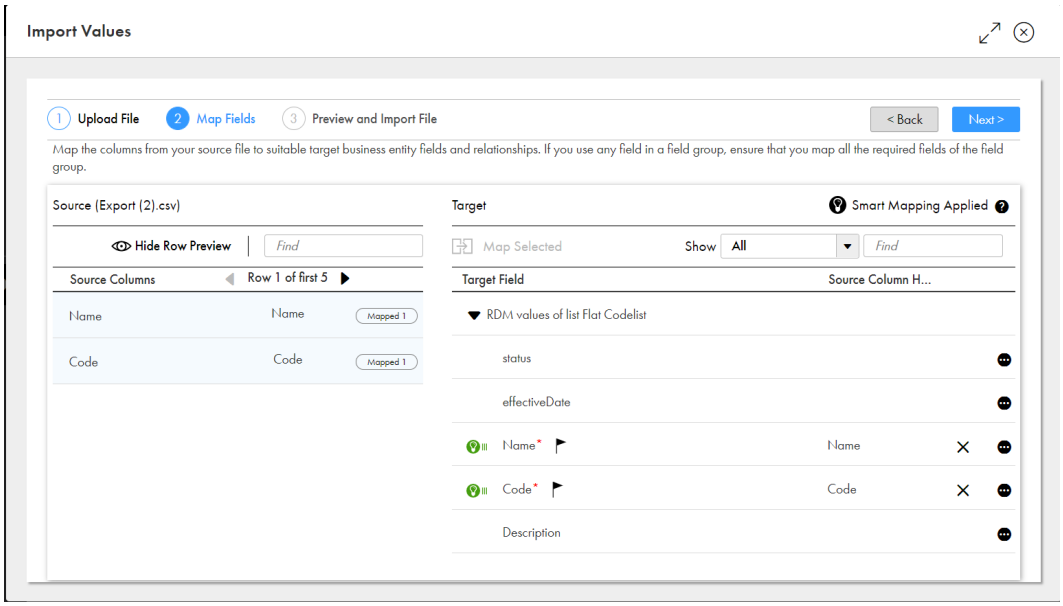
After you upload the source file, the import process automatically maps the columns of your source file to suitable target fields. For each mapped field, a confidence indicator indicates the confidence level of the field mapping.

If you don't agree with the system suggested field mapping, manually map the source columns with the appropriate target fields. When you map manually, a **User Mapped** icon appears against the mapped field.

The following table describes the confidence indicators:

Confidence Indicator	Description
	Indicates a high level of confidence in the field mapping.
	Indicates a medium level of confidence in the field mapping. If you don't agree with the mapping, delete the mapping and modify the field mapping as required.
	Indicates a low level of confidence in the field mapping. If you don't agree with the mapping, delete the mapping and modify the field mapping as required.

The following image shows some mapped fields with confidence indicators:



CHAPTER 9

Manage hierarchies

Create hierarchies to show how code values are related to other code values. First, define the hierarchy model, and then add code values to the hierarchy to define relationships between code values.

For example, you might create a location hierarchy. First, you define the hierarchy model. You define the Region code list as the top-level code list. Then you create a parent-child relationship from the Region code list to the Enterprise Country Codes code list. Based on this hierarchy model, in the hierarchy, you create a hierarchy relationship from the North America code value to the United States code value. You create a hierarchy relationship from the North America code value to the Canada code value.

You can extend a hierarchy model that contains data by adding nodes and relationships to the nodes of the hierarchy model. You can add multiple child nodes and a recursive relationship to any node. After you extend and save the hierarchy model, you cannot delete the nodes or relationships.

You can use the details panel to view the details of code values in hierarchies. The panel displays the summary, history, and relationships for a code value.

You can also export the code values in a hierarchy to a CSV file.

If you no longer need a hierarchy, you can delete the hierarchy.

Note: You must be assigned the Planner role to work with hierarchies. For more information, see "Users, groups, and roles" on page 43 .

For more information, see "Hierarchies" on page 21.

Creating hierarchy models

Create a hierarchy model to define hierarchy relationships between code lists. Later, you can add code values from the code lists to define hierarchy relationships.

To create a hierarchy model, perform the following actions:

1. Create a hierarchy asset.
2. Define the hierarchy model.
3. Assign stakeholders.

Step 1. Create a hierarchy

Create a hierarchy and configure the general properties and status.

Note: You must be assigned the Planner role to work with hierarchies. For more information, see "Users, groups, and roles" on page 43 .

1. Click **New** and select **Hierarchy**.

   The **New Hierarchy** page appears and displays the **Summary** tab.
2. Enter a name for the hierarchy.
3. Optionally, complete the general properties fields.

   Note: The domain, confidentiality, and priority are inherited by code lists in the reference data set.

Field	Description
Version	The version information of the hierarchy.
Description	A description of the hierarchy.
Application	The source application of the hierarchy.
Domain	An area or grouping to describe the hierarchy.
Confidentiality	The confidentiality level of the hierarchy.
Priority	The priority of the asset. The priority levels are critical, high, medium, and low.
Status	The state of the asset in the life cycle.
Effective Date	The date from when the status is effective.
Approved by	The user who approved the asset.
Approved on	The date of approval.

   Note: If you do not see options in some lists, configure your system reference data values. For more information, see "Adding values to system reference data" on page 59.
4. Click **Save**.

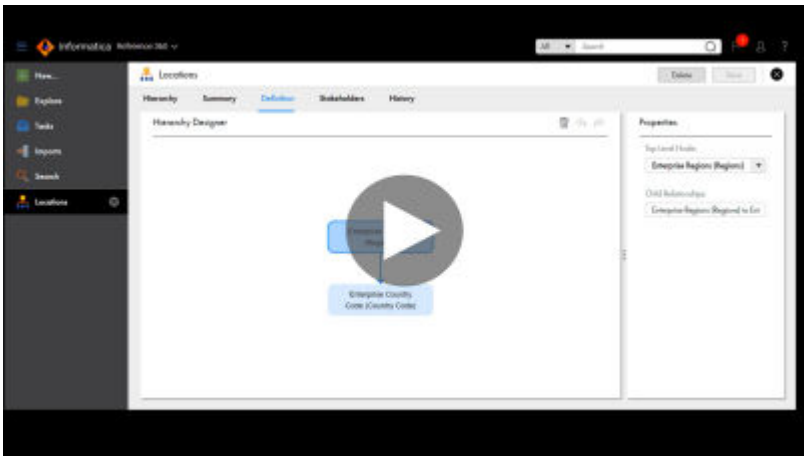
   The **Definition** tab appears and displays an undefined node in the Hierarchy Designer.

Step 2. Define the hierarchy model

Define the top level node and then add child nodes to the hierarchy model. You can add flat and hierarchical code lists as nodes to different levels of the hierarchy model. For example, you might define the Enterprise

Regions code list as the top level node. Then you create a relationship from the Enterprise Regions code list to the Enterprise Country Codes code list.

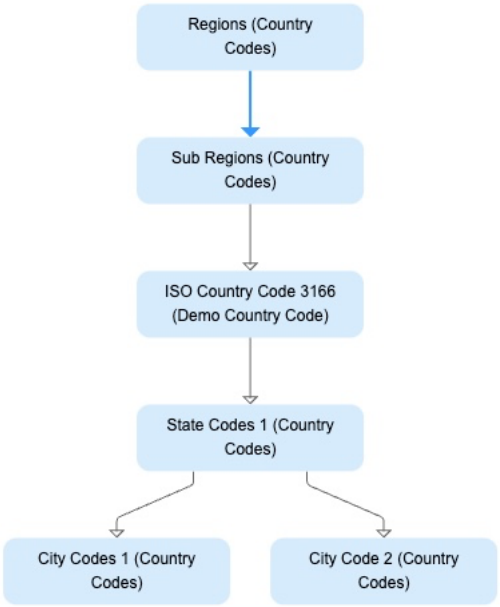
The following video shows you how to define the hierarchy model:



1. In the **Properties** panel, select a code list as the top level node for the hierarchy model.

   The undefined node name is replaced with the code list you selected.
2. In the Hierarchy Designer, hover over the node and click **Add Child**.

   A child node appears in the Hierarchy Designer.
3. In the **Node** field, select a code list to create a relationship.
   The following image shows a relationship from the top level node to the child node:



   The undefined node name is replaced with the code list you selected.
4. Optionally, add additional child nodes.
5. Click **Save**.

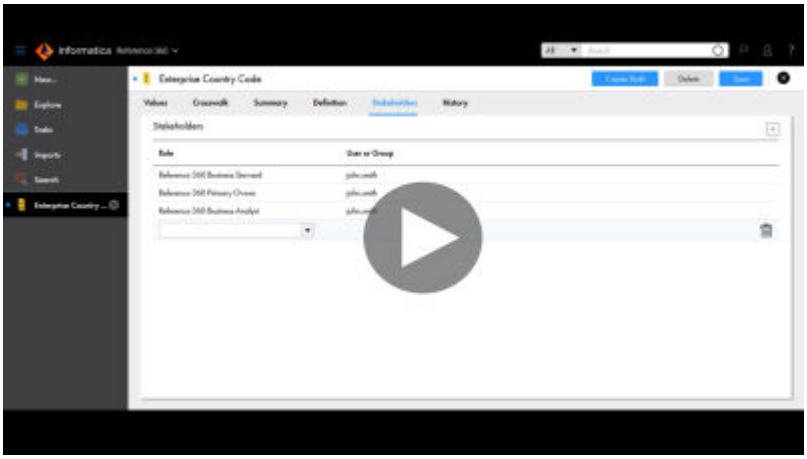
Step 3. Assign stakeholders

Assign stakeholder roles to users who play a role in using, creating, or maintaining the asset. Stakeholder roles determine a user's privileges for the asset.

When a user manages an asset as a stakeholder, they have the combined privileges provided by their stakeholder role and their Reference 360 role. For more information about stakeholder roles, see "Stakeholder roles" on page 46. For more information about assigning roles, see "Guidelines for assigning roles" on page 48.

Note: When you create crosswalks, the stakeholders assigned to the associated reference data set and code lists are copied to the crosswalk.

The following video shows you how to assign stakeholders:



1. Click **Stakeholders**.

   The **Stakeholder** tab opens.
2. Click **Add**.

   A list appears in an empty row.
3. In the **Role** list, select a stakeholder role.

   The **New Stakeholder** dialog box appears.
4. Select a user name and click **Add**.

   The user name of the stakeholder appears in the row.
5. Click **Save**.

Step 4. View the history of a hierarchy

You can view the changes made to a hierarchy in a chronological order. Use the historical data to track the changes made to the hierarchy at any point in time.

1. In the **Explore** panel, select a hierarchy and click **Open**.

   The hierarchy opens in a new tab.
2. To view the change history of the hierarchy, click the **History** tab.

   The change history of the hierarchy appears.

The following table lists the different user interface elements that you can find on the **History** tab:

User Interface Element	Description
Filters panel	Lists the predefined filters. You can select the values based on which you want to filter the history feed.
Add Field Attribute	Displays additional fields that you can select to filter the history feed.
Sort	Displays the following values based on which you can sort the history feed: - New to Old. Sorts the changes in reverse chronological order. - Old to New. Sorts the changes in chronological order.
View	Groups the history feed based on the selected frequency. Use one of the following frequencies: - Daily. Displays the changes for each day. - Monthly. Displays the changes for the entire month. - Yearly. Displays the changes for the entire year.
Calendar icon	Allows you to configure a time period. You can view the history feed for the specified time period. Use the following predefined time periods: - Today. Displays the changes occurred on the current date. - Last 7 days. Displays the changes occurred in the last seven days. - Last 30 days. Displays the changes occurred in the last 30 days.

3. To view all the fields in the history feed, clear **Show modified fields only**.

   By default, the **Show modified fields only** check box is selected, and the modified fields appear in the history feed.

Creating hierarchies

You can add, move, and remove code values in the hierarchy to define hierarchy relationships between code values.

Before you create a hierarchy, define the hierarchy model. The hierarchy relationships that you can create and the code values that you can use depends on the hierarchy model.

1. In the **Explore** panel, select a hierarchy and click **Open**.

   The hierarchy opens in a new tab.

2. Click **Hierarchy**.

   The hierarchy page appears.

3. To add code values as top-level nodes, perform the following actions:

   a. Click **Add Hierarchy Value at the Top Level**.

      The **Add Code Value** dialog box appears.

   b. To view all available code values, in the **Search** field, enter `*`.

c. Select the code values that you want to add, and click **Add**.

4. To add code values as child nodes, perform the following actions:
   a. Hover over the top-level node for which you want to add a child node, and click **Add**.

      The **Add Code Value** dialog box appears.

   b. To view all available code values, in the **Search** field, enter `*`.
   c. Select the code values that you want to add, and click **Add**.
5. To move a node, select a code value and drag the code value to a new location.
6. To remove a node, select a code value and click **Remove**.

The following image shows an example hierarchy:

Note: When you sort a hierarchy that contains multiple child nodes in a parent node, the code values are sorted based on the category of child nodes. For example, if a parent node has two categories of child nodes such as CRM country codes and ISO country codes, sorting is applied on each child node category.

Exporting hierarchies

Export code values in a hierarchy to a CSV file to analyze the hierarchies. You can choose how to format the data, the attributes to include, and the code value levels to include in the exported file.

1. In the **Explore** panel, select a hierarchy and click **Open**.

   The hierarchy appears in a new tab and displays the **Definition** tab.
2. Click **Hierarchy**.

   The **Hierarchy** page appears and displays the code values in the hierarchy.
3. Click **Export**.

   The **Export** dialog box appears with a warning message indicating that you check the formatting of the exported file.
4. On the **General** tab, enter a file name.

5. Optionally, configure how you want the data to be formatted in the exported file.

Field	Description
Delimiter	Character used to separate values.
Date Format	Format used for dates.
Decimal Mark	Decimal separator used for numbers.
Thousands Separator	Grouping separator used for numbers.

6. Optionally, on the **Columns** tab, select the code value levels and attributes that you want to export and the column order of the data.
7. Click **Export**, and then save or open the exported file.

CHAPTER 10

Manage attributes

An attribute is a component of a code value. When you create reference data sets and code lists, you define the attributes of the code values. By default, all code values consist of the Name and Code attributes.

You can configure custom attributes. For example, in an Organizational Chart reference data set, you might want code values to include a Title attribute and Location attribute, in addition to the Name and Code attributes

You can create rules for attributes to ensure that code values meet your business standards. For example, in the Country Codes code list, you might want the Code attribute for code values to require a minimum of three characters.

If you no longer need an attribute, you can delete the attribute.

For more information about attributes, see "Attributes" on page 22. For more information about rules, see "Rules" on page 36.

Note: You cannot export the rules for attributes.

Deleting attributes

Users with the Reference 360 Primary Owner role or the Primary Owner stakeholder role can delete attributes that are no longer needed. When you delete an attribute, the attribute is no longer a component of code values in the asset.

There are some attributes that you cannot delete. For more information, see "Considerations for deleting attributes" on page 97.

1. Open the reference data set or code list that contains attributes that you want to delete.
2. Click **Definition**.

   The **Definition** tab opens.
3. Hover over the attribute that you want to delete and then click **Delete**.
4. Click **OK**.
5. Click **Save**.

   Note: If the asset fails to save, the reasons why the save failed appear in a dialog box.

Considerations for deleting attributes

You can delete attributes of code values that you no longer need as components of code values. When you delete attributes, the attributes are no longer components of code values. You must be a Primary Owner to delete attributes.

You can delete attributes in reference data sets or empty code lists. You can also delete attributes that are required.

You cannot delete the following types of attributes:

- Default attributes such as Name and Code
- Attributes that are used as display attributes for the asset
- Attributes that are used as display attributes in dependent reference data sets or dependent code lists
- Attributes that are used as Reference Data attributes
- Attributes in a locked code list

Basic rule associations

You can use a basic rule association to validate or transform the attribute values based on the selected predefined function. A validation function validates whether data matches the specified condition. For example, you can set a maximum length condition for a field value. A transformation function transforms data. For example, the uppercase function converts lowercase values to uppercase.

You can group the predefined functions into the following categories:

- Validation. Validates whether data matches the specified condition. When you use a validation function, you must configure a validation message or use the default validation message.
- Transformation. Transforms data.

The following table lists the rules that you can configure for each attribute type:

Attribute Type	Rules and Descriptions
String	- **Maximum Length** - Validates whether the length of a string is less than or equal to the specified length. - **Minimum Length** - Validates whether the length of a string is greater than or equal to the specified length. - **Starts With** - Validates whether a string starts with the specified string. - **Ends With** - Validates whether a string ends with the specified string. - **No Spaces** - Validates whether a string does not contain spaces. - **Contains** - Validates whether a value contains the specified string in the exact order. - **Does Not Contain** - Validates whether a value does not contain the specified string in the exact order. - **Allowed Characters** - Validates whether a value contains the specified allowed characters. - **Not Allowed Characters** - Validates whether a value contains the specified characters that are not allowed. - **Regular Match** - Validates whether a string matches with the specified regular expression pattern. - **External Validation** - Validates the field data based on an external validation rule that is associated to a Cloud Application Integration process. The Cloud Application Integration process uses external APIs to validate the data. - **Concatenate** - Concatenates two or more strings. - **Concatenate With Spaces** - Concatenates two or more strings with spaces in between. - **Uppercase** - Converts lowercase string characters to uppercase. - **Lowercase** - Converts uppercase string characters to lowercase. - **Title Case** - Capitalizes the first character in each word of a string and converts all other characters to lowercase. - **Left Trim** - Removes spaces from the beginning of a string. - **Right Trim** - Removes spaces at the end of a string. - **Replace** - Replaces a character set in a string with another character set.
Decimal	- **Minimum Value** - Validates whether a numerical value is greater than or equal to the specified value. - **Maximum Value** - Validates whether a numerical value is less than or equal to the specified value. - **Maximum Decimal Scale** - Validates whether the number of decimal places is less than or equal to the specified value. - **Maximum Precision** - Validates whether the total number of digits in a decimal number is less than or equal to the specified value.
Integer	- **Minimum Value** - Validates whether a numerical value is greater than or equal to the specified value. - **Maximum Value** - Validates whether a numerical value is less than or equal to the specified value. - **Maximum Precision** - Validates whether the total number of digits in a decimal number is less than or equal to the specified value.
Date	- **Date Range** - Validates whether a date is in between two specified dates. - **Greater Than Date** - Validates whether a date is greater than the specified date. - **Less Than Date** - Validates whether a date is less than the specified date. - **No Future Date** - Validates whether a date is not a future date.

Guidelines for basic rule associations

Consider the following guidelines when you define basic rule associations:

- When you configure a basic rule association for a field that uses a reference data field, the rule uses the code field of the referenced code value.

- When you import code values in to a code list, date fields that you might use in a concatenate rule for the Code attribute must be in the DD-MM-YYYY format.

Adding rules for attributes

Add rules for attributes in a code list. When users add or edit code values in the code list that does not meet the business standards, they receive inline validation errors.

1. Open a code list.
2. Click **Definition**.

   The **Definition** tab opens.
3. In the **Attributes** section, select the attribute to add a rule.

   The **Attribute Details** panel displays the details of the attribute on the **Details** tab.
4. On the **Basic Rules** tab, click the **Add** icon.

   A form appears.
5. In the **General** section, enter a unique name for the rule association.
6. Optionally, enter a description.
7. On the **Condition** tab, perform the following tasks:
   a. Select a function.
   b. For the input field, select an option from the list.
   c. If you select **Select a Field**, select an attribute from the list.
   d. If you select **Enter a Value**, type a value for validation.
8. If you select a validation function, on the **Validation** tab, enter an error message or use the default validation message to display when the validation fails.
9. To verify the created rule, perform the following tasks:
   a. Click **Test**.

      The **Test** dialog box appears.
   b. Enter values in the **Input** column, and click **Run**.

      The validation result appears in the **Output** column.
   c. Click **Reset** to reset the values in the **Input** and **Output** columns.
   d. Click **Close**.
10. Click **Save**.

Changing the execution order of basic rule associations

When you have multiple basic rule associations for an attribute, set their order of execution to obtain clean and correct data.

1. On the **Rules** tab, click **Change Order**.

   The **Change Order** dialog box appears.
2. Use the up and down arrows to change the order of the rule associations.
3. Click **OK**.

   The list of rule associations with the updated order of execution appears.

Advanced rule associations

An advanced rule association is an association between attribute values and a Cloud Data Quality rule specification. You can use the predefined rule specifications or create custom rule specifications in Cloud Data Quality. For more information about rule specifications in Cloud Data Quality, see *Rule specification assets* in Cloud Data Quality.

For example, you work for an organization as a Governance Lead. To retire the old general ledger accounts created before 2012, you configure advanced rule associations based on the account creation date. These rule associations update the status of accounts created before 2012 to Retired.

To learn how to create rule specifications in Cloud Data Quality and validate reference data using the rule specifications in Reference 360, see the following video:

https://www.youtube.com/watch?v=k-vGsF2KdcQ

Guidelines for advanced rule associations

Consider the following guidelines when you define advanced rule associations:

- When a rule specification is modified in Cloud Data Quality, select the rule specification again or reset the fields of the advanced rule association in Reference 360. Otherwise, the changes to the rule specification aren't effective when the rule is executed.
- If you set an attribute as input and output fields for the same advanced rule association, you can't add values for the fields that use the rule specification, on the **New Code Value** page. To add code values to a code list, you can use reference data import jobs.
- In an advanced rule association that concatenates a reference data field with another field, the reference data field uses the code field of a referenced code value.
- When you re-enable the advanced rule association for the Code attribute and update a field other than the input fields, you can't save the code value if the input fields were updated previously.
  To save the code value, perform one of the following actions:
  - Disable the advanced rule association.
  - Revert the changes to the input fields and re-enable the advanced rule association.
- By default, Cloud Data Quality uses the Date/Time format to execute rules. When you configure advanced rule associations to validate date fields in Reference 360, the rules consider the date based on the time zone. For example, when you configure an advanced rule association to check whether the date is earlier than the current date. The rule considers the time zone and determines the date.
- To ensure the success of the comparison operation for the code value field in Reference 360, use integer or string values in the rule statement output of the advanced rule association. If you use a float value in the rule statement output, specify additional decimal values in the **Rule is Valid when Status Is** field.
- An advanced rule association for a transformation function must have rule statement output values for all the conditions in Cloud Data Quality to generate values in the output field.
- After you configure an advanced rule association to code list attributes, specify values in the rule applied field to run the rule and display validation error messages for code values.

Add advanced rule associations

Use Cloud Data Quality rule specifications to add advanced rule associations for code list attributes. You can associate a rule specification as a transformation rule or a validation rule.

1. Open a code list.

2. Click **Definition**.

   The **Definition** tab opens.

3. In the **Attributes** section, select the attribute to add a rule.

   The **Attribute Details** panel displays the details of the attribute on the **Details** tab.

4. On the **Advanced Rules** tab, click the **Add** icon.

   A form appears.

5. Select a rule specification.

   a. Click the asset picker.

      The **Select an Asset** page appears.

   b. Select a rule specification, and click **Select**.

6. Enter a unique name for the rule association.

7. Optionally, enter a description.

8. On the **Input and Output Fields** tab, perform the following tasks:

   a. In the **Input Fields** section, select a business entity field or enter an input value for each rule specification input field.

   b. In the **Output Fields** section, select a business entity field for each rule specification output field if you want to associate a transformation rule.

      If you associate a transformation rule by specifying a value in the **Output Fields** section, the **Validation** tab is disabled.

9. To associate a validation rule, specify values on the **Validation** tab without providing values in the **Output Fields** section.

   a. In the **Rule Status** section, select a rule specification field that contains the validation status.

   b. Enter a validation status that indicates successful validation.

      Note: Ensure that the validation status matches the rule specification output value for successful validation.

   c. In the **Error Status** section, configure the error message by using one of the following ways:

      - Select a rule specification field that contains an error message.
      - Enter a custom error message.

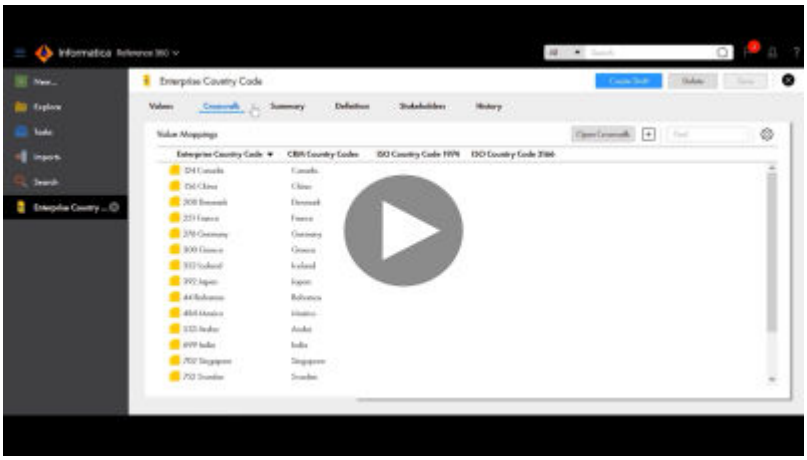
   d. Click **Save**.

Configuring display attributes

Configure the attributes that you want to represent code values in other assets. For example, you might want the Code and Name attributes to represent code values in the Country Codes code lists. You might have a

code value with the Name value as `Canada` and the Code value as `124`. When the code value appears in other assets, the code value appears as `124 Canada`.

The following video shows you how to configure display attributes:



1. Open a code list.
2. Click **Definition**.

   The **Definition** page appears.
3. To configure one attribute as the display attribute, from the **Display Attributes** list, select an attribute.
4. To configure multiple attributes as the display attributes, perform the following actions:
   a. From the Display Attributes list, select **Advanced...**.

      The **Advanced Display Attributes** dialog box appears.
   b. From the **Column** list, select an attribute.
   c. To add an additional attribute, click **Add**, and select an attribute.
   d. Repeat Step C for additional attributes.
   e. Click **OK**.

   Tip: Add the attributes in the order you want the attribute values to appear. For example, you might want the value in the Code attribute to appear before the Name attribute. Select the Code attribute first and then add the Name attribute.

   The display attributes appear in the **Display Attributes** list.
5. Click **Save**.

CHAPTER 11

Manage workflows

When you create a draft and propose changes to a code list, your edits are associated to the draft version. When you are done editing, you send your proposed changes for approval. This initiates an approval workflow and creates an approval task.

You can configure the workflow triggered when users edit code lists. The workflow configuration includes the approval tasks, approvers for each approval task, and whether comments are required.

For more information, see "Workflows" on page 27.

Related Topics:

- "Workflows" on page 27

Configuring workflows

You can configure the workflow configuration of a code list. You can add approval steps, define approvers for each approval task, and configure whether comments are required for task actions.

By default, one-step approval workflows are configured for all code lists. To create multi-step approval workflows, you add additional approval tasks to the workflow configuration of a code list.

1. In the **Explore** panel, select a code list and click **Open**.

   The code list opens on a new tab.
2. Click **Workflow**.

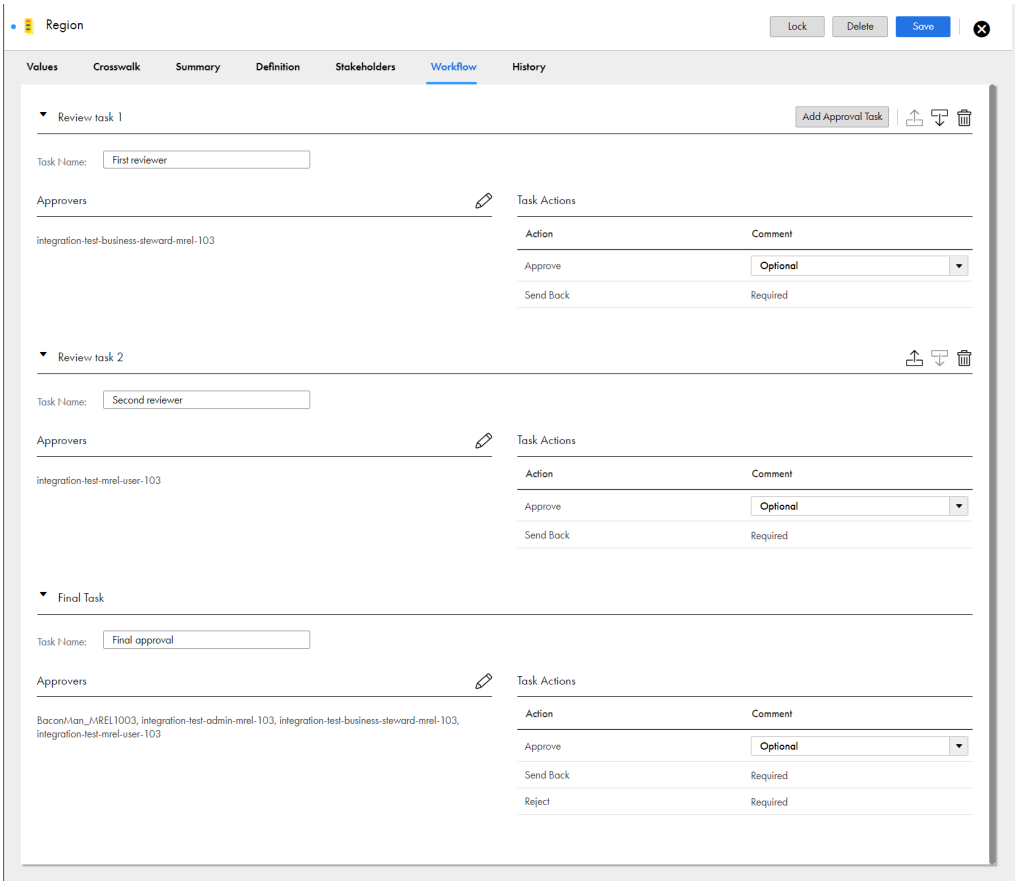
   The **Workflow** page appears.
3. To edit an existing approval task, perform any of the following actions:
   - To edit the approval task name, edit the name in the **Task Name** field.
   - To update the approvers, in the **Approvers** section, click **Edit**. Select or clear approvers and click **OK**.
   - To update whether users are required to leave a comment when they approve the approval task, in the **Task Actions** section, change your selection.
4. To add an approval task to the workflow, perform the following actions:
   a. Click **Add Approval Task**.

      The approval task appears at the top of the list.
   b. Enter an approval task name.
   c. In the **Approvers** section, click **Add**. Select approvers and click **OK**.

d. Optional. If you don't want to require users to leave a comment when they approve the approval task, select **Optional**.

5. To add additional approval tasks to the workflow, repeat step 4.
6. To rearrange the order of approval tasks, click **Move Up** or **Move Down** for the appropriate approval task.

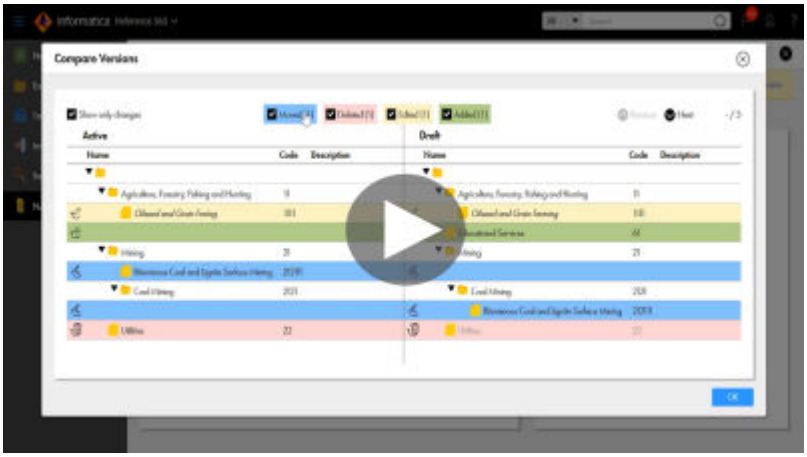
   The order that the approval tasks appear in the list determines the order that the approval tasks are linked together. The approval task at the top is the first approval task in the workflow.
7. Click **Save**.
   The following image shows a sample workflow configuration:



Proposing changes to code lists

Propose changes to a code list that is no longer accurate. You can create, edit, move, delete, or import code values in a code list.

The following video shows you how to propose changes to code lists:



1. In the **Explore** panel, select a code list and click **Open**.

   The code list opens on a new tab.
2. Click **Lock**.

   A draft version of the code list is created.
3. Perform any of the following actions:
   a. To create a code value, click **Add**, and then define a new code value.
   b. To edit a code value, select a code value. In the **Summary** tab, edit the values.
   c. To move a code value, select a code value and drag the code value to a new location.
   d. To delete a code value, hover over a code value, and then click **Delete**.
   e. To import code values, click **Import Values**.
4. Click **Save**.

   Your changes are saved to the draft version of the code list.

When you are finished proposing changes to the code list, send your changes for approval.

Note: Users assigned both the Reference 360 Business Steward role and the Business Steward stakeholder role for an asset can send their proposed changes for approval or directly publish their changes without approval. For more information about roles, see "Users, groups, and roles" on page 43.

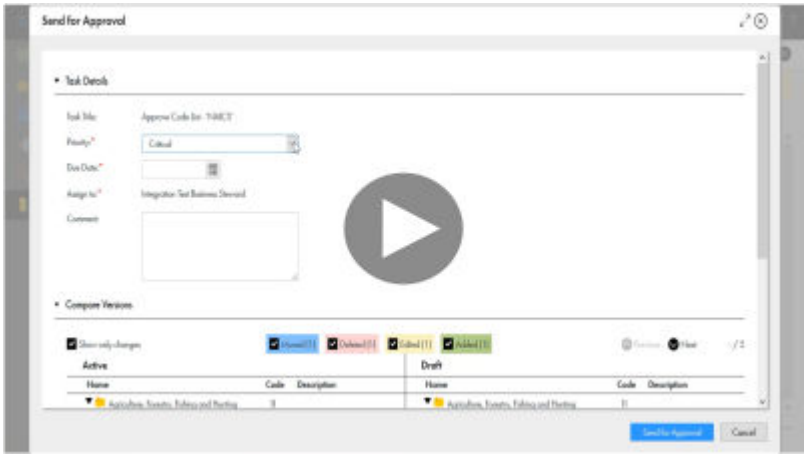
Related Topics:

- "Adding code values" on page 73

Sending changes for approval

When you are finished proposing changes to a code list, you send your changes for approval. When you send your changes for approval, you create a task and initiate an approval workflow.

The following video shows you how to send changes for approval:



1. Click **Workflow Inbox**.

   The **Workflow Inbox** page appears.
2. Select the task that contains your proposed changes.
3. Click **Send for Approval**.

   The **Send for Approval** window appears.
4. Set the priority and due date.
5. Optionally, add a comment.
6. In the **Change Comparison** section, review your proposed changes.
7. Click **Send for Approval**.

You receive an email notification confirming that you successfully sent the approval request. Users assigned the Business Steward stakeholder role for the code list or the Reference 360 Business Steward role are notified of the task. The task appears in their inbox and the **Workflow Inbox** tab.

Reviewing tasks

Users assigned the Business Steward stakeholder role for the code list or the Reference 360 Business Steward role are responsible for reviewing and approving changes proposed by other users.

1. In the **Workflow Inbox** tab or from your notification inbox, select a task.

   The **Task Details** panel displays the details of the task.
2. Review comments in the task.
3. To work on a task, click **Claim**.

   After you claim a task, other users can't work on it.
4. To confirm whether the proposed changes are correct, review the **Compare Versions** section.

5. To resolve the task, select one of the following task actions:

Option	Description
Approve	If you agree with the changes, click **Approve**. The draft and proposed changes become part of the active version.
Reject	If you disagree with the changes and think that the asset does not need to be changed, click **Reject**. Add a comment to explain your decision. The draft and proposed changes are discarded.

6. To send back the task or return the task to the pool of unassigned tasks, select one of the following task actions:

Option	Description
Send Back	If you disagree with some of the changes or think that some changes are missing, click **Send Back**. Add a comment to explain your decision.
Release	To return the task to the pool of unassigned tasks, click **Release**.

7. Click **OK**.

You receive an email notification confirming your action. Other assignees receive an email notification notifying them that another user performed an action on the approval request. The requester receives a notification in their inbox notifying them that an approver performed an action on the approval request.

Publishing proposed changes

When you are finished making your changes, publish the draft that contains your proposed changes.

You must be assigned the Business Steward role to publish your proposed changes without approval. For more information, see "Users, groups, and roles" on page 43.

1. In the **Workflow Inbox** tab, select the task associated with the code list.

   The **Task Details** panel displays the details of the task.

2. Click **Publish Draft**.

Your changes become part of the active version of the asset. Your changes are logged in the **History** tab of the asset.

CHAPTER 12

Manage jobs

Users with specific roles can create a job and define job schedules to run the jobs. Reference 360 supports file import, reference data import, and reference data export jobs.

To define a job, users must be assigned one of the following roles in addition to a Reference 360 role:

- IICS Admin
- IICS Designer
- MDM Designer

To run a job, users must be assigned the Job Executor role.

To define job schedules, users must be assigned one of the following roles in addition to a Reference 360 role:

- IICS Admin
- Service Consumer

For more information about user roles, see *User administration* in the *Administrator* help.

Defining reference data import jobs

Reference data import is a process that imports code lists and crosswalks to Reference 360. When you get started with Reference 360 for the first time, to onboard your initial data, you must import it.

Reference data import process

The process available to import reference data is Extract, Validate, and Load Data.

1. Extract data from a source file and load to a staging area.
2. Cleanse and transform the staged data.
3. Load the cleansed data to a target asset.

Prerequisites for importing reference data

Before you import reference data, ensure that you complete the following prerequisites:

1. In Administrator, create a source connection to read data from source and a Business 360 connection to write to the Reference 360 assets. For more information about creating connections, see Configuring a connection in the Administrator help.

2. In Administrator, specify a Secure Agent that is up and running as the runtime environment to run a mapping task. For more information about Secure Agent, see Runtime environments in the Administrator help.
3. In Cloud Data Integration, create a mapping with the source connection as the source transformation and the Business 360 connection as the target transformation. Configure the target by selecting the code list from Reference 360 and define field mappings for the source and target fields. Ensure that you create an in-out parameter `jobInstanceId` in the Parameters panel. For more information about creating a mapping, see Mapping configuration in the Data Integration help.

   Note: The name of the in-out parameter is case sensitive.
4. In Cloud Data Integration, create a mapping task by selecting the runtime environment and adding the mapping that you create. For more information about configuring a mapping task, see Mapping task configuration in the Data Integration help.
5. In Cloud Data Integration, create the required taskflows. Publish the taskflows to use in your reference data import job. For more information about creating taskflows, see Taskflows in the Data Integration help.

Import reference data

Create a reference data import job to import data to Reference 360 data store.

1. Click **New** > **Jobs** > **Reference Data Import** > **Create**.

   The **Reference Data Import** dialog box appears.
2. Specify the following job properties:

Property	Description
Display Name	Name of the job.
Internal ID	A unique job identifier, which is generated based on the display name that you enter. You cannot change the internal ID after you create the job.
Description	Optional. A brief description of the job.
Location	Project or folder within which you want to save the job.

3. Click **OK**.

   The reference data import job page displays the process, description, and source system.
4. To add a taskflow, perform the following tasks:
   a. Click **Add Taskflow**.

      The **Select an Asset** page appears.
   b. Select a taskflow, and click **Select**.

      The taskflow appears in the **Taskflows** section and the associated assets display in the **Assets** section.

   Note: Ensure that the taskflow is published before you execute the reference data import job.
5. Click **Save**.
6. To execute the job, click **Run**.

   You can monitor the status of job in the **My Jobs** page.

Defining reference data export jobs

You can export code lists from Reference 360 data store to an external data source.

Prerequisites

Before you export data, ensure that you complete the following prerequisites:

1. In Administrator, create a Business 360 connection to read data from Reference 360 and a target connection to write the data to an external data source. For more information about creating connections, see Configuring a connection in the Administrator help.

   Note: You cannot use same connections as source and target for reference data export jobs.
2. In Administrator, specify a Secure Agent that is up and running as the runtime environment to run a mapping task. For more information about Secure Agent, see Runtime environments in the Administrator help.
3. In Cloud Data Integration, create a mapping with the Business 360 connection as the source transformation and the target connection as the target transformation. Specify the source object by selecting code lists from Reference 360 and define field mappings by selecting **Map all descendants** for the root element. Select **root** as the output group. Ensure that you create an in-out parameter `jobInstanceId` in the Parameters panel. For more information about creating a mapping, see Mapping configuration in the Data Integration help.

   Note: The name of the in-out parameter is case sensitive.
4. In Cloud Data Integration, create a mapping task by selecting the runtime environment and adding the mapping that you create. For more information about configuring a mapping task, see Mapping task configuration in the Data Integration help.
5. In Cloud Data Integration, create the required taskflow. Publish the taskflow to use in the reference data export job. For more information about creating taskflows, see Taskflows in the Data Integration help.

Export types

You can export the code lists to an external data source. When you export the data, you can select to export all data or only incremental data after exporting all data for the first time.

You can use any one of the following export types:

Standard export

Exports all the data in the first run of the job and then exports the data that are added or updated after the first run incrementally based on the job schedule.

For example, you add 30 code values to a code list. During the first run of the job, the job exports 30 code values. After the initial run of the job, you update 15 code values to the code list. During the second run of the job, the job exports the last updated 15 code values.

Custom export

Exports all the data added or modified after the specified date in the first run of the job, and then exports the data that are added or updated after the first run incrementally.

For example, you add 20 code values to a code list after January 18, 2022, 5:00 p.m. During the first run, if you specify January 18, 2022, the job exports the 20 code values added after the specified date. After the initial run of the job, you update 10 code values. During the second run of the job, the job exports the last updated 10 code values.

Export all

Exports all the data in each run of the job.

Export reference data

Before you create a reference data export job, ensure that a taskflow is created and published in Cloud Data Integration. You can create the reference data export job to export data from Reference 360 data store to an external data source.

1. Click **New** > **Jobs** > **Reference Data Export** > **Create**.

   The **Reference Data Export** dialog box appears.

2. Specify the following job properties:

Property	Description
Display Name	Name of the job.
Internal ID	A unique job identifier, which is generated based on the display name that you enter. You cannot change the internal ID after you create the job.
Description	Optional. A brief description of the job.
Location	Project or folder within which you want to save the job.

3. Click **OK**.

   The reference data export job page displays the process, description, and export type.

4. Select one of the export types.

   For more information about export types, see "Export types" on page 110.

5. To add a taskflow, perform the following tasks:

   a. Click **Add Taskflow**.

      The **Select an Asset** page appears.

   b. Select a taskflow, and click **Select**.

      The taskflow appears in the **Taskflows** section and the associated assets display in the **Assets** section.

   Note: Ensure that the taskflow is published before you run the reference data export job.

6. Click **Save**.

7. To run the job, click **Run**.

   You can monitor the status of job on the **My Jobs** page.

Monitoring jobs

You can monitor jobs on the **My Jobs** page. This page lists all jobs that are started in Reference 360. You can schedule jobs to ensure that the job executes at a specified time.

To open the **My Jobs** page, click **My Jobs** in the left navigation bar.

My jobs page

The **Imports** tab on the navigation bar is enhanced and renamed to **My Jobs**. You can monitor the jobs instances and job schedules on the **My Jobs** page.

The **My Jobs** page contains the following tabs:

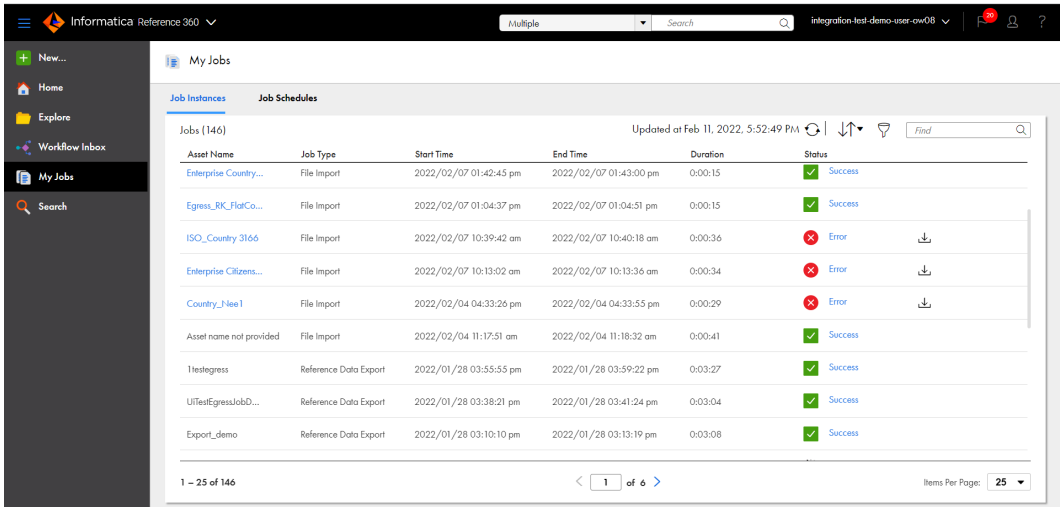
- Job Instances. Lists the job instances that are currently running, failed, stopping, stopped, and successfully completed.
- Job Schedules. Lists the job schedules.

To filter the jobs that appear on the **My Jobs** page, click the **Filter** icon. You can use filters to find specific jobs. To specify a filter, click **Add Field**, and select the property such as job type and status, to filter results.

When you search for a job on the **My Jobs** page, the search results display jobs based on instance ID, started by, job name, and status fields. The instance ID and started by fields do not appear on the **Job Instances** tab of the **My Jobs** page

You can search for reference data import jobs and reference data export jobs using asset names. You can search for a job schedule only using schedule names.

The following image shows the list of job instances on the **My Jobs** page:



By default, the following properties display for each job instance:

Property	Description
Asset Name	Name of the asset. - For file import jobs and Reference 360 draft jobs, displays the asset name to which data is imported. - For reference data import jobs and reference data export jobs, displays the job name. **Note:** Clicking the asset name of a file import job opens the asset.
Job Type	The type of job that was executed.
Start Time	Date and time that the job started.
End Time	Date and time that the job completed or stopped.

Property	Description
Duration	Time taken for the job to run.
Status	Status of the job instance.

To view the job details, click the status of the job. To download error reports for the failed jobs, click the **Download** icon in the **Status** column of the failed jobs. You can view error messages and a summary of rejected and invalid records in the error reports.

Viewing specific job details

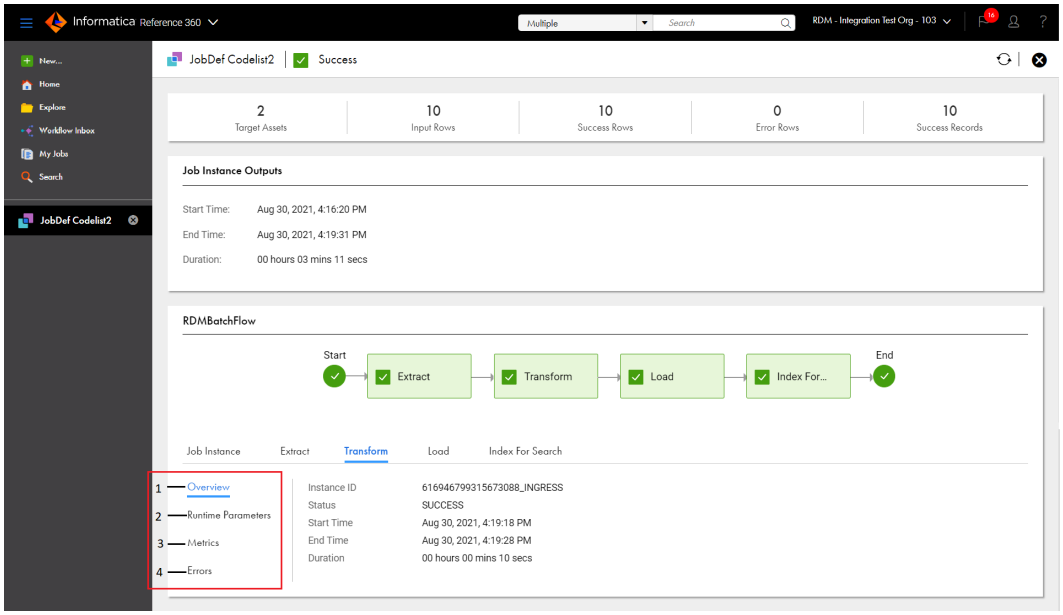
On the **My Jobs** page, click the status of any job to view detailed information about the job. If errors occur during a job execution, you can view the error details in the process steps of the job.

You can view key metrics about the job, such as total records processed, records processed successfully, and failed records. The metrics vary for each job.

You can also view the overall data flow process for each job instance. Select a specific process step to view the step level details. The details displayed for the job vary based on the job type.

Note: The number of processed records for reference data import jobs shows the summary of all types of imported records, such as business entity records and relationship records, to identify which record type failed. For example, when you import 5 rows of flat code lists, the **Input Rows** field displays the number of processed records as 10 which includes the business entity records and the relationship records.

The following image shows the details of a reference data import job:



1. Overview. Lists the details of the job instance for each step, such as instance ID, status, start time, end time, and duration.
2. Runtime Parameters. Lists the runtime parameters of the job instance and each step.
3. Metrics. Lists the key metrics related to each step, such as total records processed.
4. Errors. Lists the rejected and invalid records that failed to adhere to data quality rules.

Creating a job schedule

Schedule a job to ensure that the job executes at a specified time. You can also set a frequency to repeat the job at regular intervals. Ensure that you create a job before creating the job schedule.

1. On the **My Jobs** page, click **Job Schedules** > **Add Job Schedule**.

   The **New Job Schedule** page appears.
2. Enter the name of the job schedule.
3. To select a job, perform the following tasks:
   a. Click the asset picker.

      The **Select a Job Definition** page appears.
   b. Select a project, and then select a job definition.
   c. Click **Select**.
4. Select an existing IICS schedule, or add a new schedule. To add a new IICS schedule, perform the following tasks:
   a. Click **New Schedule**.

      The **New Schedule** dialog box appears.
   b. Enter the name and description of the schedule.
   c. Select the start date and time of the schedule.
   d. Select the time zone for the schedule.
   e. To run the job again in a schedule, select a frequency.
   f. Click **Save**.
5. Click **Save** to save the job schedule.

System-generated jobs

Reference 360 runs system-generated jobs to complete operations that take longer than expected.

Reference 360 draft job

The Reference 360 draft job is a system-generated job that runs when you import large number of code values to a draft code list and publish or discard the draft code list. The code list is locked when the job is triggered.

When operations, such as publishing or discarding a draft code list with large code value changes take longer than expected, the Reference 360 draft job starts to complete these operations. You can view the job details on the **My Jobs** page.

To stop a Reference 360 draft job that is in progress, perform one of the following tasks on the **My Jobs** page:

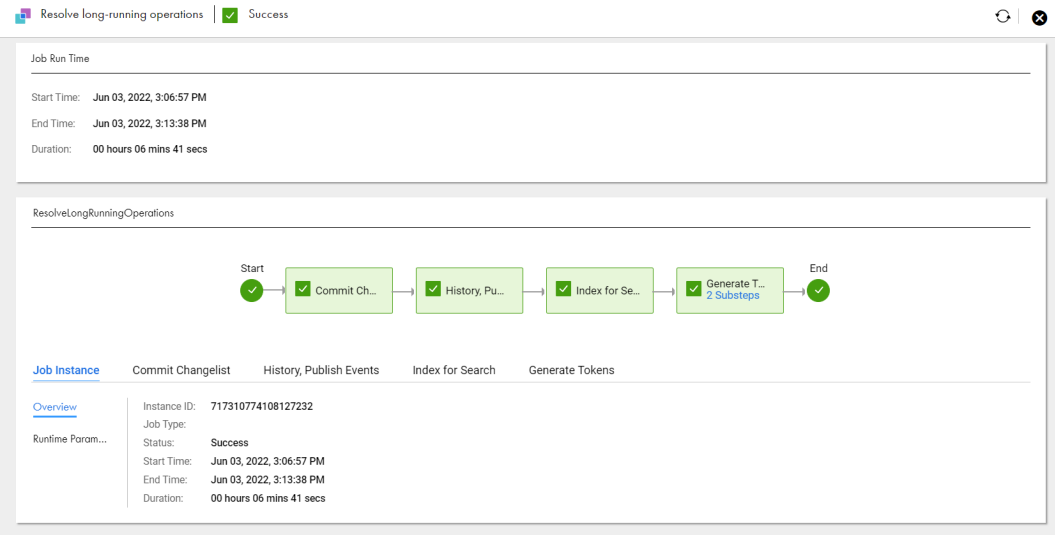
- On the **Actions** menu of the job, click **Stop**.
- Click the status of a job that is running, and then click **Stop** on the job details page.

You can restart a job. A new job is triggered when you restart the job that is stopped.

Viewing Reference 360 draft jobs

After the Reference 360 draft job is run, you can monitor and view the job details.

The following image shows a sample Reference 360 draft job type:



You can view the following steps in a Reference 360 draft job:

- Commit Changelist. Commits large code value changes to a draft code list.

  Note: The status of the code list changes to unlocked after the commit changelist step is complete.
- History, Publish Events. Generates the history feed for the imported code values.
- Index for Search. Indexes the code values for search.
- Generate Tokens. Generates tokens for the code values.

Click each step within a data flow to view more information. The parameters vary based on the step.

Note: You can view the history details of the imported code values only when the history, publish events step is complete.

Stopping reference data import and file import jobs

You can stop reference data import and file import jobs that are in progress. For example, if a job takes a longer time to complete, you might want to stop it. You can't resume a job that is stopped.

To stop a job, perform one of the following tasks on the **My Jobs** page:

- On the **Actions** menu of the job, click **Stop**.
- Click the status of a job that is running, and then click **Stop** on the job details page.

If you stop the reference data import and file import jobs during or after the Load step, the data is partially imported. To find and process this data, contact your administrator.

You can also download error reports for the stopped file import jobs.

CHAPTER 13

Reference 360 REST API

Use the Reference 360 REST APIs to interact with your reference data. For example, you can export or import reference data sets, import code values and value mappings, or retrieve a list of assets and their details.

When you use Reference 360 REST APIs, note the following rules:

- Use the following base URL:

```
<serverUrl>/rdm-service/external/v1/<API name>
```

- Use the following request header format:

```
<METHOD> <serverUrl>/<URI> HTTP/<HTTP version>
Content-Type: application/json
Accept: application/json
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXXX
```

For example, you might use the following request to export the model for reference data sets:

```
POST https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/model/export
HTTP/1.1
Content-Type: application/json
Accept: application/json
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXXX

{
   "referenceDataSetIds":[
      "56dbde1afe4d8257b6d7735e",
      "a83fa4bda81df711caca4e71"
   ]
}
```

You can access the REST API documentation in the OpenAPI format by using the following URL format:

```
https://{POD region}.informaticacloud.com/rdm-service/api-docs
```

For example, you can access the REST API documentation using the following URL:

```
https://dm-us.informaticacloud.com/rdm-service/api-docs
```

Session IDs

Each Reference 360 REST API request must be authenticated. To authenticate your requests, you must get a session ID, and then add the session ID to the header of every request. The Informatica Intelligent Cloud Services (IICS) Identity Service issues the session ID.

Note: If your session ID expires, log in again to get a new session ID.

To get a session ID, submit the following POST request with your credentials:

```
POST https://dm-us.informaticacloud.com/identity-service/api/v1/Login

{
  "username": "myUser",
  "password": "myPassword",
}
```

The response returns a `sessionId`. For example, you might receive the following response:

```
{
  ...
  "sessionId": "XXXXXXXXXXXXXXXXXXXXXX",
  "sessionExpireTime": "2000-01-01T00:00:00.000Z",
  ...
}
```

To authenticate your requests, add `IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXXX` in the request header.

The following example shows how `IDS-SESSION-ID` is used in the request header:

```
POST https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/model/export
HTTP/1.1
Content-Type: application/json
Accept: application/json
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXXX

{
  "referenceDataSetIds":[
    "56dbde1afe4d8257b6d7735e",
    "a83fa4bda81df711caca4e71"
  ]
}
```

Asset IDs

Each asset has an unique identifier. Some Reference 360 REST APIs require you to specify the ID of the asset. You can identify the ID of an asset in Reference 360 or by using REST APIs.

Reference 360

In Reference 360, the ID of an asset appears in the URL when you open a reference data set, code list, crosswalk, or hierarchy.

When you open an asset, you see the following URL format:

```
https://use4-mdm.dm-us.informaticacloud.com/rdm-ui/#/<asset type>/<asset ID>/edit
```

For example, the asset ID for the following code list is `5d38987dbc49de0001113db3`:

```
https://use4-mdm.dm-us.informaticacloud.com/rdm-ui/#/codelist/5d38987dbc49de0001113db3/
edit
```

REST APIs

The following table describes the REST APIs that you can use to identify the ID of an asset:

REST API	Description
List reference data sets	Retrieves a list of reference data sets with their ID. With the ID of a reference data set, you can use the List code lists REST API to retrieve a list of code lists.
List code lists	Retrieves a list of code lists with their IDs for the specified reference data set. With the ID of a code list, you can use the List crosswalks REST API to retrieve a list of crosswalks.
List crosswalks	Retrieves a list of crosswalks associated with the code list and the ID of each crosswalk.
List hierarchies	Retrieves a list of hierarchies with their ID.

Resources

The API resources in this section apply specifically to the Reference 360 service.

The following table describes the resources that you can use:

Resource	Description
configuration	Configures the Reference 360 organization with settings, such as the approval mode configuration and approval implementation mode.
model	Exports or imports the models of reference data sets, including their code lists and crosswalks.
import	Imports code values, value mappings, and hierarchy relationships. Checks the status of an import job, and retrieves the details of a failed import job.
export	Exports the data in code lists, crosswalks, and hierarchies to a CSV file or the JSON format.
rds	Lists reference data sets, the details of a reference data set, and the code lists in a reference data set.
codelists	Lists the details of a code list, the details of a code value, and the crosswalks associated with a code list. Lists the modified code values in a code list, the modified code value relationships in a hierarchical code list, and the modified code lists. Deletes code values that you no longer need. You can also unlock code lists locked by other users.
crosswalks	Lists the details of a crosswalk and the value mappings for a code value.
hierarchies	Lists hierarchies, hierarchy details, and hierarchy model relationships.
enums	Lists system reference data values, adds system reference data values, and updates system reference data values.

configuration

Use this resource to configure the Reference 360 organization with settings, such as the approval mode for Business Stewards, approval implementation mode, and the rule validation mode in Reference 360.

Note: To use the configuration resource, you must be assigned the Informatica Intelligent Cloud Services Reference 360 Administrator role.

Approval mode configuration

The approval mode determines whether approval is required for changes to code values and whether direct import is enabled.

The following table describes the supported approval modes for users assigned the Business Steward role:

Approval Mode	Description
DIRECT_PUBLISH_AND_APPROVAL	- Can choose to send their changes for approval or publish their drafts. - Can directly import code values.
DIRECT_PUBLISH	- Must publish their drafts. - Can directly import code values.
APPROVAL	- Must send their changes for approval. - Cannot directly import code values.

Approval implementation mode

The approval implementation mode determines whether the Workflow Inbox and multi-step approval workflows are enabled.

Note: Before you update the approval implementation mode, resolve all tasks and close all draft code lists. Existing tasks and draft code lists will be discarded. After you update the approval implementation mode to `PLATFORM`, you cannot revert back to the `RDM` approval implementation mode.

The following table describes the supported approval implementation modes:

Approval Implementation Mode	Description
RDM	Enables the Tasks interface and uses a pre-defined one-step approval workflow. The default approval implementation mode is `RDM`.
PLATFORM	Enables the Workflow Inbox interface and multi-step approval workflows.

Rule validation mode

The rule validation mode determines the type of rules that you can configure for attributes in a code list.

The following table describes the rule validation modes:

Rule Validation Mode	Description
RDM	Allows configuring the legacy rules in Reference 360.
PLATFORM	Allows configuring the basic rule associations based on the enhanced data validation framework.

Note: After you change the rule validation mode to `PLATFORM`, you cannot revert back to the `RDM` rule validation mode.

Get approval mode configuration

Retrieves the approval mode configuration.

GET request

To get the approval mode configuration, submit a GET request with the following URI:

```
/rdm-service/external/v1/configuration/approval
```

GET response

The response contains the approval mode.

The following table describes the field:

Field	Description
mode	Approval modes. The following modes are available: - DIRECT_PUBLISH_AND_APPROVAL - DIRECT_PUBLISH - APPROVAL

GET example

To retrieve the approval mode, you might use the following request:

```
GET  https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/configuration/
approval HTTP/1.1
Accept: application/json
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXXX
```

The following sample response shows the approval mode:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 31

{
        "mode" : "DIRECT_PUBLISH"
}
```

Update approval mode configuration

Updates the approval mode configuration.

PUT request

To update the approval mode configuration, submit a PUT request with the following URI and specify the approval mode:

```
/rdm-service/external/v1/configuration/approval?mode=<approval mode>
```

Use the `mode` parameter in the URI and request body to specify the approval mode configuration.

Parameter	Description
mode	Approval modes. Values are `DIRECT_PUBLISH_AND_APPROVAL`, `DIRECT_PUBLISH`, or `APPROVAL`.

PUT response

A 204 no content response is returned.

PUT example

To update the approval mode configuration, you might use the following request:

```
PUT https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/configuration/
approval?mode=DIRECT_PUBLISH_AND_APPROVAL HTTP/1.1
Host: localhost:8080
Content-Type: application/x-www-form-urlencoded

mode=DIRECT_PUBLISH_AND_APPROVAL
```

The following sample response shows the no content response:

```
HTTP/1.1 204 No Content
```

Get approval implementation mode

Retrieves the approval implementation mode.

GET request

To get the approval implementation mode, submit a GET request with the following URI:

```
/rdm-service/external/v1/configuration/approvalImplementation
```

GET response

The response contains the approval implementation mode.

The following table describes the field:

Field	Description
mode	Approval implementation mode. The following modes are available: - `RDM`. Enables the Tasks interface and uses a pre-defined one-step approval workflow. - `PLATFORM`. Enables the Workflow Inbox interface and multi-step approval workflows.

GET example

To retrieve the approval implementation mode, you might use the following request:

```
GET  https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/configuration/
approvalImplementation HTTP/1.1
```

```
Accept: application/json
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXX
```

The following sample response shows the approval implementation mode:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 20

{
   "mode":"RDM"
}
```

Update approval implementation mode

Updates the approval implementation mode.

Note: Before you update the approval implementation mode, resolve all tasks and close all draft code lists. Existing tasks and draft code lists will be discarded. After you update the approval implementation mode to `PLATFORM`, you cannot revert back to the `RDM` approval implementation mode.

PUT request

To update the approval implementation mode, submit a PUT request with the following URI and specify the approval implementation mode:

```
/rdm-service/external/v1/configuration/approvalImplementation?mode=<mode>
```

Use the `mode` parameter in the URI and request body to specify the approval mode configuration.

Field	Description
mode	Approval implementation mode. The following modes are available: - `RDM`. Enables the Tasks interface and uses a pre-defined one-step approval workflow. - `PLATFORM`. Enables the Workflow Inbox interface and multi-step approval workflows.

PUT response

A 204 no content response is returned.

PUT example

To update the approval implementation mode, you might use the following request:

```
PUT https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/configuration/
approvalImplementation?mode=PLATFORM HTTP/1.1
Content-Type: application/x-www-form-urlencoded
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXX

mode=PLATFORM
```

The following sample response shows the no content response:

```
HTTP/1.1 204 No Content
```

Get current rule validation mode

Retrieves the current rule validation mode.

GET request

To get the current mode of rule validation, submit a GET request with the following URI:

```
/rdm-service/external/v1/configuration/ruleValidationMode
```

GET response

The response contains the current rule validation mode.

The following table describes the field:

Field	Description
ruleValidationMode	Rule validation mode. Value must be `RDM` or `PLATFORM`.

GET example

To retrieve the current rule validation mode, you might use the following request:

```
GET  https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/configuration/
ruleValidationMode HTTP/1.1
Accept: application/json
```

The following sample response shows the current rule validation mode:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 39

{
  "ruleValidationMode" : "PLATFORM"
}
```

Migrate to enhanced data validation framework

Migrates legacy data validation rules to enhanced data validation framework.

Note: After you migrate all your legacy data validation rules to the enhanced data validation framework, you cannot revert back to the legacy data validation framework.

PUT request

To migrate to the enhanced data validation framework, submit a PUT request with the following URI and specify the rule validation mode:

```
/rdm-service/external/v1/configuration/ruleValidationMode?mode=PLATFORM
```

Use the `mode` parameter in the URI to specify the rule validation mode.

Field	Description
mode	Rule validation mode. Value must be `PLATFORM`.

PUT response

The response contains a token of a rule validation migration job.

The following table describes the field in the response:

Field	Type	Description
token	String	Token of a started rule validation migration job.

PUT example

To migrate to the enhanced data validation framework, you might use the following request:

```
PUT https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/configuration/
ruleValidationMode?mode=PLATFORM HTTP/1.1
Accept: application/json
Content-Type: application/x-www-form-urlencoded
```

The following sample response shows a token of a rule validation migration job:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 42

{
  "token" : "61abc4332e3b3459e56ad7da"
}
```

Get rule validation migration job details

Retrieves the progress, status, and result of a rule validation migration job.

GET request

To get the rule validation migration job details, submit a GET request with the following URI:

```
/rdm-service/external/v1/configuration/ruleValidationMode/migrationStatus/{token}
```

GET response

The response contains the details of rule validation migration job, such as the job status, the number of records processed for migration, and the error details.

The following table describes the fields in the response:

Field	Type	Description
token	String	Token of a rule validation migration job.
createdByUserId	String	ID of the user who initiated the migration job.
createdDate	String	Date when the migration job was initiated.
status	String	Current status of the migration job. Values are `RUNNING`, `SUCCESS`, `FAILED`, or `CANCELLED`.
initialNumberOfCodeLists	Number	Number of code lists available for migration.
processedNumberOfCodeLists	Number	Number of code lists processed.
errors	Array	Errors occurred during migration.
codeListId	String	ID of the code list in which error occurred.
errorCode	String	Error code for the error type.
errorSummary	String	Error summary.

GET example

To retrieve the rule validation migration job details, you might use the following request:

```
GET  https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/configuration/
ruleValidationMode/migrationStatus/4a86d1f6687a9bc83d1cdd80 HTTP/1.1
Accept: application/json
```

The following sample response shows the progress, status and result of a migration job:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 395

{
   "token":"4a86d1f6687a9bc83d1cdd80",
   "createdByUserId":"integration-test-admin",
   "createdDate":"2021-11-09T17:18:48.610+00:00",
   "status":"RUNNING",
   "initialNumberOfCodeLists":30,
   "processedNumberOfCodeLists":20,
   "errors":[
      {
         "codeListId":"5c1fdd059ca47cb706f2ecd2",
         "errorCode":"RDM.0040000",
         "errorSummary":"Internal RDM application error"
      }
   ]
}
```

model version 1

Support for export and import model version 1 APIs is deprecated in December 2022. You can use export and import model version 2 APIs to export and import assets from source to target organizations.

model version 2

Use this resource to export and import reference data sets and hierarchies.

The model resource exports and imports the reference data sets and hierarchies, including the following entities:

- Basic and advanced rule associations of code list attributes
- Crosswalks
- User groups in the stakeholder configuration of assets
- Workflow configurations that contain workflow tasks with user group assignments

The model resource doesn't export and import the data in code lists or crosswalks, such as code values and value mappings.

To export and import data, use the export and import resources.

Export model

Exports the reference data sets and hierarchies.

Note: When you create a crosswalk between code lists of different reference data sets, if you don't specify the IDs of the source and target reference data sets, the crosswalk export fails without a warning message.

POST request

To export the reference data sets and hierarchies, submit a POST request with the following URI:

```
/rdm-service/external/v2/model/export
```

Use the following attributes in the request body to specify the reference data sets and hierarchies to export:

Field	Type	Description
referenceDataSetIds	Array	Comma-separated list of IDs of reference data sets to export. If you don't specify IDs, all reference data sets are exported. **Note:** You can find the ID of assets in Reference 360 or retrieve the IDs by using REST APIs. For more information, see "Asset IDs" on page 117.
hierarchyIds	Array	Comma-separated list of IDs of hierarchies to export. If you don't specify IDs, all hierarchies are exported. **Note:** You can find the ID of assets in Reference 360 or retrieve the IDs by using REST APIs. For more information, see "Asset IDs" on page 117.

POST response

The response contains the exported assets.

The following table describes the attributes in the response:

Field	Type	Description
version	String	Version of the export file.
referenceDataSets	-	Includes details about the reference data set.
id	String	ID of the reference data sets. For more information, see "Asset IDs" on page 117.
name	String	Name of the asset.
description	String	Description of the asset.
hierarchical	Boolean	Indicates whether code lists in the reference data set inherit the hierarchical structure definition.
levels	Number	Number of hierarchical levels supported in the code lists associated with the reference data set. If hierarchical levels are not provided, value is `1`. If hierarchical levels are unlimited, value is `-1`.
defaultList	String	ID of the default code list.
codeValueFields	-	Includes the attribute definition for code values in the reference data set.

Field	Type	Description
name	String	Name of the field.
origin	String	Origin of the definition of the field.
datatype	String	Datatype of the field. Values are `String`, `Integer`, `Decimal`, `Boolean`, `Date`, or `Reference Data`.
mandatory	Boolean	Indicates whether the attribute is required.
dependencyDef	-	Includes the definition of the asset specified as the dependency.
termId	String	ID of the asset specified as the dependency.
assetStakeholders	-	Stakeholders associated to the asset.
codeLists	-	Includes details about the code list.
id	String	ID of the code lists. For more information, see "Asset IDs" on page 117.
termId	String	ID of the reference data set to which the code list is associated.
name	String	Name of the asset.
description	String	Description of the asset.
hierarchical	Boolean	Indicates whether code lists in the reference data set inherit the hierarchical structure definition.
levels	Number	Number of hierarchical levels supported in the code lists associated with the reference data set. If hierarchical levels are not provided, value is `1`. If hierarchical levels are unlimited, value is `-1`.
codeValueFields	-	Includes the attribute definition for code values in the reference data set.
name	String	Name of the field.
origin	String	Origin of the definition of the field.
datatype	String	Datatype of the field. Values are `String`, `Integer`, `Decimal`, `Boolean`, `Date`, or `Reference Data`.
mandatory	Boolean	Indicates whether the attribute is required.
dependencyDef	-	Includes the definition of the asset specified as the dependency.
termId	String	ID of the asset specified as the dependency.
displayColumns	Array	Display columns used as labels for code values associated with the dependent asset.
dqValidationInfo	-	Includes details about the basic and advanced rule associations of code list attributes.

Field	Type	Description
assetStakeholders	-	Stakeholders associated to the asset.
assetWorkflowConfiguration	-	Workflow configuration of the asset.
crosswalks	-	Details about the crosswalk.
id	String	ID of the crosswalks. For more information, see "Asset IDs" on page 117.
description	String	Description of the asset.
status	String	Status of the asset.
sourceCodelistId	String	ID of the source code list to which the crosswalk is associated.
targetCodelistId	String	ID of the target code list.
assetStakeholders	-	Stakeholders associated to the asset.
enums	-	Includes details about enum groups and entries.
key	String	ID of the system reference data value.
label	String	Label for the system reference data value.
hierarchies	-	Includes details about the hierarchies.
codeListRelations	-	Code list relations defined by the hierarchies.

POST example

To export reference data sets and hierarchies, you might use the following request:

```
POST https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/model/export
HTTP/1.1
Content-Type: application/json
Accept: application/json
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXXX

{
   "referenceDataSetIds":[
      "aaa97c7034473568b09d65f7",
      "fddde0de9f7740721d3ac264"
   ],
   "hierarchyIds":[
      "hierarchy1"
   ]
}
```

The following sample response shows the exported model of reference data sets and hierarchies:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 6076

{
   "version":"3.6",
   "referenceDataSets":[
      {
         "id":"fddde0de9f7740721d3ac264",
         "name":"Country",
```

```
   "description":"desc",
   "hierarchical":false,
   "levels":1,
   "defaultList":"be82a1de2bc0fbd095249478",
   "codeValueFields":[
      {
         "name":"Name",
         "labels":[
            {
               "language":"en",
               "value":"Name"
            }
         ],
         "origin":"TERM",
         "datatype":"String",
         "mandatory":true
      },
      {
         "name":"Code",
         "labels":[
            {
               "language":"en",
               "value":"Code"
            }
         ],
         "origin":"TERM",
         "datatype":"String",
         "mandatory":true
      }
   ],
   "assetStakeholders":{
      "stakeholders":[
         {

         }
      ]
   }
},
{
   "id":"aaa97c7034473568b09d65f7",
   "name":"rds2",
   "description":"desc",
   "hierarchical":false,
   "levels":1,
   "defaultList":"51cf12512e6fa0602c4fe438",
   "codeValueFields":[
      {
         "name":"Name",
         "labels":[
            {
               "language":"en",
               "value":"Name"
            }
         ],
         "origin":"TERM",
         "datatype":"String",
         "mandatory":true
      },
      {
         "name":"Code",
         "labels":[
            {
               "language":"en",
               "value":"Code"
            }
         ],
         "origin":"TERM",
         "datatype":"String",
         "mandatory":true
      },
      {
```

```
                "name":"fieldDef",
                "labels":[
                   {
                      "language":"en",
                      "value":"fieldDef"
                   }
                ],
                "origin":"TERM",
                "datatype":"Reference",
                "mandatory":false,
                "relatedTermId":"fddde0de9f7740721d3ac264",
                "displayColumns":[
                   "Name"
                ]
             }
          ],
          "dependencyDef":{
             "termId":"fddde0de9f7740721d3ac264",
             "displayColumns":[
                "Name"
             ]
          },
          "assetStakeholders":{
             "stakeholders":[
                {

                }
             ]
          }
       }
    ],
    "codeLists":[
       {
          "id":"db73317aab6d239460fdac9f",
          "termId":"fddde0de9f7740721d3ac264",
          "name":"rds1_cl1_name",
          "description":"desc",
          "hierarchical":false,
          "codeValueFields":[
             {
                "name":"Name",
                "labels":[
                   {
                      "language":"en",
                      "value":"Name"
                   }
                ],
                "origin":"TERM",
                "datatype":"String",
                "mandatory":true
             },
             {
                "name":"Code",
                "labels":[
                   {
                      "language":"en",
                      "value":"Code"
                   }
                ],
                "origin":"TERM",
                "datatype":"String",
                "mandatory":true
             }
          ],
          "assetWorkflowConfiguration":{
             "assetWorkFlowTasks":[
                {
                   "taskName":"TestTaskSendback",
                   "taskAction":{
                      "approve":"REQUIRED",
                      "reject":"REQUIRED",
```

```
                     "sendBack":"REQUIRED"
                  },
                  "approvers":{
                     "userGroups":[
                        {
                           "name":"testgroup2"
                        }
                     ]
                  }
               },
               {
                  "taskName":"TestTaskReject",
                  "taskAction":{
                     "approve":"OPTIONAL",
                     "sendBack":"REQUIRED"
                  },
                  "approvers":{
                     "userGroups":[
                        {
                           "name":"testgroup1"
                        }
                     ]
                  }
               },
               {
                  "taskName":"TestTakApprove",
                  "taskAction":{
                     "approve":"REQUIRED",
                     "sendBack":"REQUIRED"
                  },
                  "approvers":{
                     "userGroups":[
                        {
                           "name":"testgroup"
                        }
                     ]
                  }
               }
            ]
         },
         "rdsName":"Country",
         "dqRulesFrsLocation":{
            "5sciIEK0QBKftMdqHqTCeT":{
               "name":"RuleSpecName",
               "project":"Default",
               "folder":"ExampleFolder"
            }
         },
         "dqValidationInfo":{
            "validationConfiguration":{
               "dqLightWeightRuleAssociations":[
                  {
                     "eClass":"http://informatica.com/mdm/v1/DQI#//
DQLightWeightRuleAssociation",
                     "associatedFieldName":{
                        "$ref":"//@field[name='Name']"
                     },
                     "dqLightWeightRules":[
                        {
                           "eClass":"http://informatica.com/mdm/v1/DQI#//
DQLightWeightRuleConfiguration",
                           "description":"Minimum Length",
                           "validationMessage":"Enter a minimum of 3 characters.",
                           "downgradeTrustScorePercentage":100,
                           "errorSeverity":"INFO",
                           "dqRuleName":"Minimum Length",
                           "dqLightWeightRuleId":"MIN_LENGTH",
                           "dqLightWeightRuleParameters":"[{\"min\":3}]",
                           "isEnabled":true
                        }
                     ]
```

```
                  }
               ],
               "dqRuleAssociations":[
                  {
                     "ruleId":"5sciIEK0QBKftMdqHqTCeT",
                     "statusField":"Validation",
                     "statusCode":"Valid number",
                     "statusMessageField":"Validation",
                     "validationMessage":"",
                     "errorSeverity":"ERROR",
                     "associatedFieldName":{
                        "$ref":"//@field[name='Name']"
                     },
                     "id":"Validation_AlternateIdentifier",
                     "dqRuleName":"Validation_AlternateIdentifier",
                     "description":"Validate if the alternate identifier has a valid
format",
                     "isEnabled":true,
                     "ruleInputFields":[
                        {
                           "accessPath":{
                              "eclass":"http://informatica.com/mdm/v2/Core#//AccessPath",
                              "pathElements":[
                                 {
                                    "$ref":"//@field[name='Name']"
                                 }
                              ]
                           },
                           "ruleFieldName":"value",
                           "downgradeTrustScorePercentage":100
                        }
                     ]
                  }
               ],
               "eClass":"http://informatica.com/mdm/v1/DQI#//DQRuleValidationConfig"
            },
            "enabled":false
         },
         "assetStakeholders":{
            "stakeholders":[
               {

               }
            ]
         }
      },
      {
         "id":"fb4c079c4d8376e03f72a73a",
         "termId":"aaa97c7034473568b09d65f7",
         "name":"SapCountry",
         "description":"desc",
         "hierarchical":false,
         "codeValueFields":[
            {
               "name":"Name",
               "labels":[
                  {
                     "language":"en",
                     "value":"Name"
                  }
               ],
               "origin":"TERM",
               "datatype":"String",
               "mandatory":true
            },
            {
               "name":"Code",
               "labels":[
                  {
                     "language":"en",
                     "value":"Code"
```

```
            }
         ],
         "origin":"TERM",
         "datatype":"String",
         "mandatory":true
      }
   ],
   "assetWorkflowConfiguration":{
      "assetWorkFlowTasks":[
         {
            "taskName":"TestTaskSendback",
            "taskAction":{
               "approve":"REQUIRED",
               "reject":"REQUIRED",
               "sendBack":"REQUIRED"
            },
            "approvers":{
               "userGroups":[
                  {
                     "name":"testgroup2"
                  }
               ]
            }
         },
         {
            "taskName":"TestTaskReject",
            "taskAction":{
               "approve":"OPTIONAL",
               "sendBack":"REQUIRED"
            },
            "approvers":{
               "userGroups":[
                  {
                     "name":"testgroup1"
                  }
               ]
            }
         },
         {
            "taskName":"TestTakApprove",
            "taskAction":{
               "approve":"REQUIRED",
               "sendBack":"REQUIRED"
            },
            "approvers":{
               "userGroups":[
                  {
                     "name":"testgroup"
                  }
               ]
            }
         }
      ]
   },
   "rdsName":"rds2",
   "dqValidationInfo":{
      "enabled":false
   },
   "assetStakeholders":{
      "stakeholders":[
         {

         }
      ]
   }
},
{
   "id":"71e94e4b43e146edb370461a",
   "termId":"aaa97c7034473568b09d65f7",
   "name":"IsoCountry",
   "description":"desc",
```

```
"hierarchical":false,
"codeValueFields":[
   {
      "name":"Name",
      "labels":[
         {
            "language":"en",
            "value":"Name"
         }
      ],
      "origin":"TERM",
      "datatype":"String",
      "mandatory":true
   },
   {
      "name":"Code",
      "labels":[
         {
            "language":"en",
            "value":"Code"
         }
      ],
      "origin":"TERM",
      "datatype":"String",
      "mandatory":true
   }
],
"assetWorkflowConfiguration":{
   "assetWorkFlowTasks":[
      {
         "taskName":"TestTaskSendback",
         "taskAction":{
            "approve":"REQUIRED",
            "reject":"REQUIRED",
            "sendBack":"REQUIRED"
         },
         "approvers":{
            "userGroups":[
               {
                  "name":"testgroup2"
               }
            ]
         }
      },
      {
         "taskName":"TestTaskReject",
         "taskAction":{
            "approve":"OPTIONAL",
            "sendBack":"REQUIRED"
         },
         "approvers":{
            "userGroups":[
               {
                  "name":"testgroup1"
               }
            ]
         }
      },
      {
         "taskName":"TestTakApprove",
         "taskAction":{
            "approve":"REQUIRED",
            "sendBack":"REQUIRED"
         },
         "approvers":{
            "userGroups":[
               {
                  "name":"testgroup"
               }
            ]
         }
```

```
            }
         ]
      },
      "rdsName":"rds2",
      "dqValidationInfo":{
         "enabled":false
      },
      "assetStakeholders":{
         "stakeholders":[
            {

            }
         ]
      }
   }
],
"crosswalks":[
   {
      "id":"c984c9006ec3b492c2395e1a",
      "description":"description",
      "status":"status",
      "sourceCodelistId":"71e94e4b43e146edb370461a",
      "targetCodelistId":"fb4c079c4d8376e03f72a73a",
      "assetStakeholders":{
         "stakeholders":[
            {

            }
         ]
      }
   }
],
"enums":{
   "application":[
      {
         "key":"CRM",
         "label":"CRM"
      }
   ]
},
"hierarchies":[
   {
      "id":"hierarchy1",
      "name":"hierarchy1",
      "description":"hierarchy1 desc",
      "codeListRelations":{
         "relations":[
            {
               "parent":{
                  "codeListId":"71e94e4b43e146edb370461a",
                  "codeListName":"IsoCountry",
                  "termId":"aaa97c7034473568b09d65f7",
                  "termName":"rds2"
               },
               "child":{
                  "codeListId":"db73317aab6d239460fdac9f",
                  "codeListName":"rds1_cl1_name",
                  "termId":"fddde0de9f7740721d3ac264",
                  "termName":"Country"
               }
            }
         ]
      },
      "stakeholderAssignments":{
         "stakeholders":[
            {
               "subject":{
                  "name":"rdmUserGroup",
                  "type":"USERGROUP"
               }
            }
```

```
                ]
            }
        }
    ]
}
```

Import model

Imports the new assets from a previously exported model and skips the existing assets.

POST request

To import the new assets from a previously exported model, submit a POST request with the following URI:

```
/rdm-service/external/v2/model/import?isDeltaImport=true
```

The request contains the attributes that the Export model REST API returns. For more information, see "Export model" on page 126.

The following table describes the attributes in the request:

Field	Type	Description
version	String	Version of the export file.
referenceDataSets	-	Includes details about the reference data set.
id	String	ID of the reference data sets. For more information, see "Asset IDs" on page 117.
name	String	Name of the asset.
description	String	Description of the asset.
hierarchical	Boolean	Indicates whether code lists in the reference data set inherit the hierarchical structure definition.
levels	Number	Optional. Number of hierarchical levels supported in the code lists associated with the reference data set. If hierarchical levels are not provided, value is `1`. If hierarchical levels are unlimited, value is `-1`.
defaultList	String	ID of the default code list.
codeValueFields	-	Includes the attribute definition for code values in the reference data set.
name	String	Name of the field.
origin	String	Origin of the definition of the field.
datatype	String	Datatype of the field. Values are `String`, `Integer`, `Decimal`, `Boolean`, `Date`, or `Reference Data`.
mandatory	Boolean	Indicates whether the attribute is required.
dependencyDef	-	Includes the definition of the asset specified as the dependency.

Field	Type	Description
termId	String	ID of the asset specified as the dependency.
assetStakeholders	-	Stakeholders associated to the asset.
codeLists	-	Includes details about the code list.
id	String	ID of the code lists. For more information, see "Asset IDs" on page 117.
termId	String	ID of the reference data set to which the code list is associated.
name	String	Name of the asset.
description	String	Description of the asset.
hierarchical	Boolean	Indicates whether code lists in the reference data set inherit the hierarchical structure definition.
levels	Number	Number of hierarchical levels supported in the code lists associated with the reference data set. If hierarchical levels are not provided, value is `1`. If hierarchical levels are unlimited, value is `-1`.
codeValueFields	-	Includes the attribute definition for code values in the reference data set.
name	String	Name of the field.
origin	String	Origin of the definition of the field.
datatype	String	Datatype of the field. Values are `String`, `Integer`, `Decimal`, `Boolean`, `Date`, or `Reference Data`.
mandatory	Boolean	Indicates whether the attribute is required.
dependencyDef	-	Includes the definition of the asset specified as the dependency.
termId	String	ID of the asset specified as the dependency.
displayColumns	Array	Display columns used as labels for code values associated with the dependent asset.
dqValidationInfo	-	Includes details about the basic and advanced rule associations of code list attributes.
assetStakeholders	-	Stakeholders associated to the asset.
assetWorkflowConfiguration	-	Workflow configuration of the asset.
crosswalks	-	Details about the crosswalk.
id	String	ID of the crosswalks. For more information, see "Asset IDs" on page 117.
description	String	Description of the asset.

Field	Type	Description
status	String	Status of the asset.
sourceCodelistId	String	ID of the source code list to which the crosswalk is associated.
targetCodelistId	String	ID of the target code list.
assetStakeholders	-	Stakeholders associated to the asset.
enums	-	Includes details about enum groups and entries. For more information, see "enums" on page 234.
key	String	ID of the system reference data value.
label	String	Label for the system reference data value.
hierarchies	-	Includes details about the hierarchies.
codeListRelations	-	Code list relations defined by the hierarchies.

POST response

The response contains a detailed report about the import process.

The following table describes the attributes in the response:

Field	Type	Description
numImportedRds	Number	Number of imported reference data sets.
numImportedCodeList	Number	Number of imported code lists.
numImportedCrosswalk	Number	Number of imported crosswalks.
numImportedHierarchy	Number	Number of imported hierarchies.
ignoredRdsCount	Number	Number of ignored reference data sets.
ignoredCodeListCount	Number	Number of ignored code lists.
ignoredCrosswalkCount	Number	Number of ignored crosswalks.
ignoredHierarchyCount	Number	Number of ignored hierarchies.
importedEntities	-	Includes details about the imported entities.
type	String	Type of imported entity. Values are `rds`, `codelist`, `crosswalk`, or `hierarchy`.
name	String	Name of the imported entity.
oldId	String	ID of the imported entity in the source organization.

Field	Type	Description
newId	String	ID of the imported entity in the target organization.
ignoredEntities	-	Includes details about the ignored entities.
type	String	Type of ignored entity. Values are `rds`, `codelist`, `crosswalk`, or `hierarchy`.
name	String	Name of the ignored entity.
oldId	String	ID of the ignored entity in the source organization.
newId	String	ID of the ignored entity in the target organization.
dqRuleAssociationImportReport	Object	Advanced rule association import report.
stakeholderImportResult	-	Includes details about the imported stakeholders.
ignoredStakeholderEntries	Array	Includes details of the ignored stakeholders.
ignoredStakeholderEntries > reason	String	The reason for ignoring stakeholders during import.
ignoredStakeholderEntries > stakeholder	Object	Ignored stakeholder record.
ignoredStakeholderEntries > assetIdentity	Object	Details of the asset that the stakeholder is assigned to.
ignoredStakeholderEntries > assetIdentity > id	String	Identifier of the asset.
ignoredStakeholderEntries > assetIdentity > assetName	String	Name of the asset.
ignoredStakeholderEntries > assetIdentity > assetType	String	Type of asset. Values are `rds`, `codelist`, `crosswalk`, or `hierarchy`.
assetWorkflowConfigurationImportReport	-	Includes details about the imported workflow configuration.
assetWorkflowImportIgnoredTasks	Array	Includes details of the workflow tasks ignored during import.
assetWorkflowImportIgnoredTasks > assetIdentity	Object	Details of the asset to which the workflow configuration belongs.
assetWorkflowImportIgnoredTasks > assetIdentity > id	String	Identifier of the asset.
assetWorkflowImportIgnoredTasks > assetIdentity > assetName	String	Name of the asset.
assetWorkflowImportIgnoredTasks > assetIdentity > assetType	String	Type of asset. Values are `rds`, `codelist`, `crosswalk`, or `hierarchy`.

Field	Type	Description
assetWorkflowImportIgnoredTasks > reason	String	The reason for ignoring workflow tasks during import.
assetWorkflowImportIgnoredTasks > taskName	String	Name of the workflow task.
assetWorkflowImportIgnoredTasks > notFoundUserGroups	Array	List of user groups that aren't found.
assetWorkflowImportIgnoredTasks > importIgnoredWorkflowTask	Object	Ignored workflow tasks.
assetWorkflowImportIgnoredTasks > importIgnoredWorkflowTask > taskName	String	Name of the workflow task.
assetWorkflowImportIgnoredTasks > importIgnoredWorkflowTask > taskAction	Object	Workflow task action.
assetWorkflowImportIgnoredTasks > importIgnoredWorkflowTask > taskAction > approve	String	Action name of the workflow task.
assetWorkflowImportIgnoredTasks > importIgnoredWorkflowTask > taskAction > sendBack	String	Action name of the workflow task.
assetWorkflowImportIgnoredTasks > importIgnoredWorkflowTask > taskAction > reject	String	Action name of the workflow task.
assetWorkflowImportIgnoredTasks > importIgnoredWorkflowTask > approvers	Object	Work flow task approvers.
assetWorkflowImportIgnoredTasks > importIgnoredWorkflowTask > approvers > userGroups	Array	User groups of work flow task approvers.
assetWorkflowImportIgnoredTasks > importIgnoredWorkflowTask > approvers > userGroups > name	String	Names of user groups of work flow task approvers.
enumImportResult	-	Includes details about the enum entries.
newEntries	Number	Number of new entries imported.
updatedEntries	Number	Number of updated entries.
existingEntries	Number	Number of existing entries that were skipped.

POST example

The following sample request imports only the new assets and skips the existing assets:

```
POST https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v2/model/import?
isDeltaImport=true HTTP/1.1
Content-Type: application/json
Accept: application/json
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXXX

{
   "version":"3.6",
   "referenceDataSets":[
```

```
{
   "id":"fddde0de9f7740721d3ac264",
   "name":"Country",
   "description":"desc",
   "hierarchical":false,
   "levels":1,
   "defaultList":"be82a1de2bc0fbd095249478",
   "codeValueFields":[
      {
         "name":"Name",
         "labels":[
            {
               "language":"en",
               "value":"Name"
            }
         ],
         "origin":"TERM",
         "datatype":"String",
         "mandatory":true
      },
      {
         "name":"Code",
         "labels":[
            {
               "language":"en",
               "value":"Code"
            }
         ],
         "origin":"TERM",
         "datatype":"String",
         "mandatory":true
      }
   ],
   "assetStakeholders":{
      "stakeholders":[
         {

         }
      ]
   }
},
{
   "id":"aaa97c7034473568b09d65f7",
   "name":"rds2",
   "description":"desc",
   "hierarchical":false,
   "levels":1,
   "defaultList":"51cf12512e6fa0602c4fe438",
   "codeValueFields":[
      {
         "name":"Name",
         "labels":[
            {
               "language":"en",
               "value":"Name"
            }
         ],
         "origin":"TERM",
         "datatype":"String",
         "mandatory":true
      },
      {
         "name":"Code",
         "labels":[
            {
               "language":"en",
               "value":"Code"
            }
         ],
         "origin":"TERM",
```

```
                "datatype":"String",
                "mandatory":true
             },
             {
                "name":"fieldDef",
                "labels":[
                   {
                      "language":"en",
                      "value":"fieldDef"
                   }
                ],
                "origin":"TERM",
                "datatype":"Reference",
                "mandatory":false,
                "relatedTermId":"fddde0de9f7740721d3ac264",
                "displayColumns":[
                   "Name"
                ]
             }
          ],
          "dependencyDef":{
             "termId":"fddde0de9f7740721d3ac264",
             "displayColumns":[
                "Name"
             ]
          },
          "assetStakeholders":{
             "stakeholders":[
                {

                }
             ]
          }
       }
    ],
    "codeLists":[
       {
          "id":"db73317aab6d239460fdac9f",
          "termId":"fddde0de9f7740721d3ac264",
          "name":"rds1_cl1_name",
          "description":"desc",
          "hierarchical":false,
          "codeValueFields":[
             {
                "name":"Name",
                "labels":[
                   {
                      "language":"en",
                      "value":"Name"
                   }
                ],
                "origin":"TERM",
                "datatype":"String",
                "mandatory":true
             },
             {
                "name":"Code",
                "labels":[
                   {
                      "language":"en",
                      "value":"Code"
                   }
                ],
                "origin":"TERM",
                "datatype":"String",
                "mandatory":true
             }
          ],
          "assetWorkflowConfiguration":{
             "assetWorkFlowTasks":[
```

```
                {
                   "taskName":"TestTaskSendback",
                   "taskAction":{
                      "approve":"REQUIRED",
                      "reject":"REQUIRED",
                      "sendBack":"REQUIRED"
                   },
                   "approvers":{
                      "userGroups":[
                         {
                            "name":"testgroup2"
                         }
                      ]
                   }
                },
                {
                   "taskName":"TestTaskReject",
                   "taskAction":{
                      "approve":"OPTIONAL",
                      "sendBack":"REQUIRED"
                   },
                   "approvers":{
                      "userGroups":[
                         {
                            "name":"testgroup1"
                         }
                      ]
                   }
                },
                {
                   "taskName":"TestTakApprove",
                   "taskAction":{
                      "approve":"REQUIRED",
                      "sendBack":"REQUIRED"
                   },
                   "approvers":{
                      "userGroups":[
                         {
                            "name":"testgroup"
                         }
                      ]
                   }
                }
             ]
          },
          "rdsName":"Country",
          "dqRulesFrsLocation":{
             "5sciIEK0QBKftMdqHqTCeT":{
                "name":"RuleSpecName",
                "project":"Default",
                "folder":"ExampleFolder"
             }
          },
          "dqValidationInfo":{
             "validationConfiguration":{
                "dqLightWeightRuleAssociations":[
                   {
                      "eClass":"http://informatica.com/mdm/v1/DQI#//
DQLightWeightRuleAssociation",
                      "associatedFieldName":{
                         "$ref":"//@field[name='Name']"
                      },
                      "dqLightWeightRules":[
                         {
                            "eClass":"http://informatica.com/mdm/v1/DQI#//
DQLightWeightRuleConfiguration",
                            "description":"Minimum Length",
                            "validationMessage":"Enter a minimum of 3 characters.",
                            "downgradeTrustScorePercentage":100,
                            "errorSeverity":"INFO",
```

```
                              "dqRuleName":"Minimum Length",
                              "dqLightWeightRuleId":"MIN_LENGTH",
                              "dqLightWeightRuleParameters":"[{\"min\":3}]",
                              "isEnabled":true
                           }
                        ]
                     }
                  ],
                  "dqRuleAssociations":[
                     {
                        "ruleId":"5sciIEK0QBKftMdqHqTCeT",
                        "statusField":"Validation",
                        "statusCode":"Valid number",
                        "statusMessageField":"Validation",
                        "validationMessage":"",
                        "errorSeverity":"ERROR",
                        "associatedFieldName":{
                           "$ref":"//@field[name='Name']"
                        },
                        "id":"Validation_AlternateIdentifier",
                        "dqRuleName":"Validation_AlternateIdentifier",
                        "description":"Validate if the alternate identifier has a valid
format",
                        "isEnabled":true,
                        "ruleInputFields":[
                           {
                              "accessPath":{
                                 "eclass":"http://informatica.com/mdm/v2/Core#//AccessPath",
                                 "pathElements":[
                                    {
                                       "$ref":"//@field[name='Name']"
                                    }
                                 ]
                              },
                              "ruleFieldName":"value",
                              "downgradeTrustScorePercentage":100
                           }
                        ]
                     }
                  ],
                  "eClass":"http://informatica.com/mdm/v1/DQI#//DQRuleValidationConfig"
               },
               "enabled":false
            },
            "assetStakeholders":{
               "stakeholders":[
                  {

                  }
               ]
            }
         },
         {
            "id":"fb4c079c4d8376e03f72a73a",
            "termId":"aaa97c7034473568b09d65f7",
            "name":"SapCountry",
            "description":"desc",
            "hierarchical":false,
            "codeValueFields":[
               {
                  "name":"Name",
                  "labels":[
                     {
                        "language":"en",
                        "value":"Name"
                     }
                  ],
                  "origin":"TERM",
                  "datatype":"String",
                  "mandatory":true
```

```
   },
   {
      "name":"Code",
      "labels":[
         {
            "language":"en",
            "value":"Code"
         }
      ],
      "origin":"TERM",
      "datatype":"String",
      "mandatory":true
   }
],
"assetWorkflowConfiguration":{
   "assetWorkFlowTasks":[
      {
         "taskName":"TestTaskSendback",
         "taskAction":{
            "approve":"REQUIRED",
            "reject":"REQUIRED",
            "sendBack":"REQUIRED"
         },
         "approvers":{
            "userGroups":[
               {
                  "name":"testgroup2"
               }
            ]
         }
      },
      {
         "taskName":"TestTaskReject",
         "taskAction":{
            "approve":"OPTIONAL",
            "sendBack":"REQUIRED"
         },
         "approvers":{
            "userGroups":[
               {
                  "name":"testgroup1"
               }
            ]
         }
      },
      {
         "taskName":"TestTakApprove",
         "taskAction":{
            "approve":"REQUIRED",
            "sendBack":"REQUIRED"
         },
         "approvers":{
            "userGroups":[
               {
                  "name":"testgroup"
               }
            ]
         }
      }
   ]
},
"rdsName":"rds2",
"dqValidationInfo":{
   "enabled":false
},
"assetStakeholders":{
   "stakeholders":[
      {

      }
```

```
      ]
   }
},
{
   "id":"71e94e4b43e146edb370461a",
   "termId":"aaa97c7034473568b09d65f7",
   "name":"IsoCountry",
   "description":"desc",
   "hierarchical":false,
   "codeValueFields":[
      {
         "name":"Name",
         "labels":[
            {
               "language":"en",
               "value":"Name"
            }
         ],
         "origin":"TERM",
         "datatype":"String",
         "mandatory":true
      },
      {
         "name":"Code",
         "labels":[
            {
               "language":"en",
               "value":"Code"
            }
         ],
         "origin":"TERM",
         "datatype":"String",
         "mandatory":true
      }
   ],
   "assetWorkflowConfiguration":{
      "assetWorkFlowTasks":[
         {
            "taskName":"TestTaskSendback",
            "taskAction":{
               "approve":"REQUIRED",
               "reject":"REQUIRED",
               "sendBack":"REQUIRED"
            },
            "approvers":{
               "userGroups":[
                  {
                     "name":"testgroup2"
                  }
               ]
            }
         },
         {
            "taskName":"TestTaskReject",
            "taskAction":{
               "approve":"OPTIONAL",
               "sendBack":"REQUIRED"
            },
            "approvers":{
               "userGroups":[
                  {
                     "name":"testgroup1"
                  }
               ]
            }
         },
         {
            "taskName":"TestTakApprove",
            "taskAction":{
               "approve":"REQUIRED",
```

```
                  "sendBack":"REQUIRED"
               },
               "approvers":{
                  "userGroups":[
                     {
                        "name":"testgroup"
                     }
                  ]
               }
            }
         ]
      },
      "rdsName":"rds2",
      "dqValidationInfo":{
         "enabled":false
      },
      "assetStakeholders":{
         "stakeholders":[
            {

            }
         ]
      }
   }
],
"crosswalks":[
   {
      "id":"c984c9006ec3b492c2395e1a",
      "description":"description",
      "status":"status",
      "sourceCodelistId":"71e94e4b43e146edb370461a",
      "targetCodelistId":"fb4c079c4d8376e03f72a73a",
      "assetStakeholders":{
         "stakeholders":[
            {

            }
         ]
      }
   }
],
"enums":{
   "application":[
      {
         "key":"CRM",
         "label":"CRM"
      }
   ]
},
"hierarchies":[
   {
      "id":"hierarchy1",
      "name":"hierarchy1",
      "description":"hierarchy1 desc",
      "codeListRelations":{
         "relations":[
            {
               "parent":{
                  "codeListId":"71e94e4b43e146edb370461a",
                  "codeListName":"IsoCountry",
                  "termId":"aaa97c7034473568b09d65f7",
                  "termName":"rds2"
               },
               "child":{
                  "codeListId":"db73317aab6d239460fdac9f",
                  "codeListName":"rds1_cl1_name",
                  "termId":"fddde0de9f7740721d3ac264",
                  "termName":"Country"
               }
            }
```

```
            ]
         },
         "stakeholderAssignments":{
            "stakeholders":[
               {
                  "subject":{
                     "name":"rdmUserGroup",
                     "type":"USERGROUP"
                  }
               }
            ]
         }
      }
   ]
}
```

The following sample response shows the number of imported and ignored reference data sets, code lists, crosswalks, and hierarchies:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 3436

{
   "stakeholderImportResult":{
      "ignoredStakeholderEntries":[
         {
            "reason":"User or UserGroup does not exist to create stakeholder",
            "assetIdentity":{
               "id":"fddde0de9f7740721d3ac264",
               "assetType":"REFERENCEDATASET",
               "assetName":"Country"
            },
            "stakeholder":{

            }
         },
         {
            "reason":"User or UserGroup does not exist to create stakeholder",
            "assetIdentity":{
               "id":"aaa97c7034473568b09d65f7",
               "assetType":"REFERENCEDATASET",
               "assetName":"rds2"
            },
            "stakeholder":{

            }
         },
         {
            "reason":"User or UserGroup does not exist to create stakeholder",
            "assetIdentity":{
               "id":"db73317aab6d239460fdac9f",
               "assetType":"CODELIST",
               "assetName":"rds1_cl1_name"
            },
            "stakeholder":{

            }
         },
         {
            "reason":"User or UserGroup does not exist to create stakeholder",
            "assetIdentity":{
               "id":"fb4c079c4d8376e03f72a73a",
               "assetType":"CODELIST",
               "assetName":"SapCountry"
            },
            "stakeholder":{

            }
         },
```

```
            {
               "reason":"User or UserGroup does not exist to create stakeholder",
               "assetIdentity":{
                  "id":"71e94e4b43e146edb370461a",
                  "assetType":"CODELIST",
                  "assetName":"IsoCountry"
               },
               "stakeholder":{

               }
            },
            {
               "reason":"User or UserGroup does not exist to create stakeholder",
               "assetIdentity":{
                  "id":"c984c9006ec3b492c2395e1a",
                  "assetType":"CROSSWALK"
               },
               "stakeholder":{

               }
            },
            {
               "reason":"The user group does not exists in the system to import as
stakeholder assignment to Hierarchy hierarchy1",
               "assetIdentity":{
                  "id":"hierarchy1",
                  "assetType":"HIERARCHY",
                  "assetName":"hierarchy1"
               },
               "stakeholder":{
                  "subject":{
                     "name":"rdmUserGroup",
                     "type":"USERGROUP"
                  }
               }
            }
         ]
      },
      "assetWorkflowConfigurationImportReport":{

      },
      "dqRuleAssociationImportReport":{

      },
      "numImportedRds":1,
      "numImportedCodeList":3,
      "numImportedCrosswalk":1,
      "numImportedHierarchy":1,
      "importedEntities":[
         {
            "type":"rds",
            "name":"rds2",
            "oldId":"aaa97c7034473568b09d65f7",
            "newId":"2a5cf980972eaf022eca5171"
         },
         {
            "type":"codelist",
            "name":"rds1_cl1_name",
            "oldId":"db73317aab6d239460fdac9f",
            "newId":"15ae578c5d9a5033c4aa7a20"
         },
         {
            "type":"codelist",
            "name":"SapCountry",
            "oldId":"fb4c079c4d8376e03f72a73a",
            "newId":"4d58dc28b5f05bc79364deaf"
         },
         {
            "type":"codelist",
            "name":"IsoCountry",
```

```
            "oldId":"71e94e4b43e146edb370461a",
            "newId":"f5b8669f65d5c9d5fbac2239"
         },
         {
            "type":"crosswalk",
            "name":"description",
            "oldId":"c984c9006ec3b492c2395e1a",
            "newId":"981302a3652292f4d119d124"
         },
         {
            "type":"hierarchy",
            "name":"hierarchy1",
            "oldId":"hierarchy1",
            "newId":"236b199ca17891054dfa2616"
         }
      ],
      "ignoredRdsCount":1,
      "ignoredCodeListCount":0,
      "ignoredCrosswalkCount":0,
      "ignoredHierarchyCount":0,
      "ignoredEntities":[
         {
            "type":"rds",
            "name":"Country",
            "oldId":"fddde0de9f7740721d3ac264",
            "newId":"aa8f0e009e1f2d002e9a8be6"
         }
      ],
      "enumImportResult":{
         "newEntries":2,
         "existingEntries":3,
         "updatedEntries":0
      }
   }
```

Import failure

Some entities might fail to import. The ID of the entities that failed to import and the reason why they failed are listed in the response.

Attribute	Type	Description
errorMessage	Object	Additional information about why the import failed.

The following sample response shows import failures:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 1713

{
   "errorCode":"RDM.0010000",
   "errorSummary":"Validation error",
   "errorLink":"RDM.0010000",
   "errorId":"ef12db22-a813-44aa-92c9-ce7ce72e15d0",
   "stackTraceMessage":[
      "com.informatica.mdm.rdm.errorhandling.RDMApplicationException: {",
      "  \"errorCode\" : \"RDM.0010000\",",
      "  \"errorSummary\" : \"Validation error\",",
      "  \"errorLink\" : \"RDM.0010000\",",
      "  \"errorId\" : \"ef12db22-a813-44aa-92c9-ce7ce72e15d0\",",
      "  \"errorCauses\" : [ {",
      "    \"errorCode\" : \"RDM.0010074\",",
      "    \"errorSummary\" : \"The model misses a referred term.\",",
      "    \"errorParameter\" : {",
      "      \"id\" : \"638f1bfec541e907948f9bf0\"",
      "    }",
      "  } ]",
```

```
      "} ",
   "errorCauses":[
      {
         "errorCode":"RDM.0010074",
         "errorSummary":"The model misses a referred term.",
         "errorParameter":{
            "id":"638f1bfec541e907948f9bf0"
         }
      }
   ]
}
```

import version 1

Use this resource to import code values and value mappings, retrieve the status of an import job, and retrieve the detailed report for a failed import job.

Import code values

Imports code values into a code list.

Note: Validation checks are not performed when you import code values.

POST request

To import code values into a code list, submit a POST request with the following URI:

```
/rdm-service/external/v1/import
```

The request contains form-data with the following parameters:

file

Specify a CSV file that contains the code value attributes. The columns specified depend on your data model. The CSV must start with two header rows, followed by the data rows.

For example, you might have the following code values:

```
Name,Code
Name,Code
Afghanistan,AFG
Aland Islands,ALA
Albania,ALB
Algeria,DZA
American Samoa,ASM
```

Note: You can provide additional information about a code value. For example, you might want to assign a code value to the Approved status. In the `status.key` column for the code value, enter the key value for the system reference data value that you want to assign. For more information about your configured system reference data values, see "List system reference data values" on page 234.

importSettings

Specify the file specific configuration and container details.

The `importSettings` parameter includes the following attributes:

Field	Type	Description
delimiter	String	Delimiter used to separate values. Value must be `ASTERISK`, `CIRCUMFLEX`, `COLON`, `COMMA`, `PIPE`, `SECTION`, `SEMICOLON`, `SPACE`, or `TAB`.
textQualifier	String	Symbol used to indicate where a text begins and ends. Value must be `DOUBLE_QUOTE`, `SINGLE_QUOTE`, or `NO_QUOTE`.
codepage	String	Code page used for the export file. Value must be `UTF8`.
dateFormat	String	Format used for dates. Use one of the following formats: - `DE`. For dd.mm.yyyy format. - `ISO`. For yyyy-mm-dd format. - `US`. For mm/dd/yyyy format.
containerType	String	Type of asset that contains code values. Value must be `codelist`.
containerId	String	The ID of the code list to which you want to import code values. **Note:** You can find the ID of assets in Reference 360 or retrieve the IDs by using REST APIs. For more information, see "Asset IDs" on page 117.
startingRow	String	Line number from which to start importing data. By default, all rows are imported.

POST response

The response contains the details of the import job.

The response contains the following attributes:

Field	Type	Description
jobId	String	ID of the job.
state	String	Status of the job. Values are `CREATED`, `INPROGRESS`, `COMPLETED`, `FAILED`, or `STOPPED`.
startTime	Number	Time, in milliseconds, when the job started.
numOfRecordsProcessed	Number	Number of records processed.
numOfRecordsFailed	Number	Number of records failed to import.
numOfRecordsSucceeded	Number	Number of records successfully imported.

POST example

To import code values, you might use the following request:

```
POST https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/import HTTP/1.1
Content-Type: multipart/form-data; boundary=6o2knFse3p53ty9dmcQvWAIx1zInP11uCfbm
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXXXX

--6o2knFse3p53ty9dmcQvWAIx1zInP11uCfbm
Content-Disposition: form-data; name=file; filename=import-code-values.csv

--6o2knFse3p53ty9dmcQvWAIx1zInP11uCfbm
Content-Disposition: form-data; name=importSettings
```

```
Content-Type: application/json;charset=UTF-8

{
  "delimiter":"COMMA",
  "textQualifier":"DOUBLE_QUOTE",
  "codepage":"UTF8",
  "dateFormat":"ISO",
  "containerType":"CODELIST",
  "containerId":"9ab3201990a54dcdc86f54cf",
  "startingRow":null
}
--6o2knFse3p53ty9dmcQvWAIx1zInP11uCfbm--
```

The CSV file might contain the following header rows and data rows:

```
Name,Code
Name,Code
Afghanistan,AFG
Aland Islands,ALA
Albania,ALB
Algeria,DZA
American Samoa,ASM
```

The following sample response shows the status of the import job:

```
{
  "jobId":"dd1b2018cb47cef99f8d0f42",
  "state":"INPROGRESS",
  "startTime":1561367377428,
  "numOfRecordsProcessed":100,
  "numOfRecordsFailed":25,
  "numOfRecordsSucceeded":75
}
```

Import value mappings

Imports value mappings into a crosswalk.

POST request

To import value mappings into a crosswalk, submit a POST request with the following URI:

```
/rdm-service/external/v1/import
```

The request contains form-data with two parameters:

file

Specify a CSV file that contains the code value attributes. The columns specified depend on your data model. The CSV starts with two header rows, followed by the data rows.

For example, you might have the following code values:

```
sourcePKey,_from.id.sourcePKey,_to.id.sourcePKey
sourcePKey,_from.id.sourcePKey,_to.id.sourcePKey
AF_AFG,AF,AFG
AL_ALA,AL,ALA
ALB_ALB,ALB,ALB
DZ_DZA,DZ,DZA
AS_ASM,AS,ASM
```

importSettings

Specify the file specific configuration and container details.

The `importSettings` includes the following attributes:

Field	Type	Description
delimiter	String	Delimiter used to separate values. Value must be `ASTERISK`, `CIRCUMFLEX`, `COLON`, `COMMA`, `PIPE`, `SECTION`, `SEMICOLON`, `SPACE`, or `TAB`.
textQualifier	String	Symbol used to indicate where a text begins and ends. Value must be `DOUBLE_QUOTE`, `SINGLE_QUOTE`, or `NO_QUOTE`.
codepage	String	Code page used for the export file. Value must be `UTF8`.
dateFormat	String	Format used for dates. Use one of the following formats: - `DE`. For dd.mm.yyyy format. - `ISO`. For yyyy-mm-dd format. - `US`. For mm/dd/yyyy format.
containerType	String	Type of asset that contains value mappings. Value must be `crosswalk`.
containerId	String	The ID of the crosswalk to which you want to import value mappings. **Note:** You can find the ID of assets in Reference 360 or retrieve the IDs by using REST APIs. For more information, see "Asset IDs" on page 117.
startingRow	String	Line number from which to start importing data. By default, all rows are imported.

POST response

The response contains the details of the import job.

The following table describes the attributes in the response:

Field	Type	Description
jobId	String	ID of the job.
state	String	Status of the job. Values are `CREATED`, `INPROGRESS`, `COMPLETED`, `FAILED`, or `STOPPED`.
startTime	Number	Time, in milliseconds, when the job started.
numOfRecordsProcessed	Number	Number of records processed.
numOfRecordsFailed	Number	Number of records failed to import.
numOfRecordsSucceeded	Number	Number of records successfully imported.

POST example

To import value mappings into a crosswalk, you might use the following request:

```
POST https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/import HTTP/1.1
Content-Type: multipart/form-data; boundary=6o2knFse3p53ty9dmcQvWAIx1zInP11uCfbm
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXXXX

--6o2knFse3p53ty9dmcQvWAIx1zInP11uCfbm
Content-Disposition: form-data; name=file; filename=import-value-mappings.csv

--6o2knFse3p53ty9dmcQvWAIx1zInP11uCfbm
Content-Disposition: form-data; name=importSettings
```

```
Content-Type: application/json;charset=UTF-8

{
  "delimiter":"COMMA",
  "textQualifier":"DOUBLE_QUOTE",
  "codepage":"UTF8",
  "dateFormat":"ISO",
  "containerType":"CROSSWALK",
  "containerId":"9ab3201990a54dcdc86f53AB",
  "startingRow":null
}
--6o2knFse3p53ty9dmcQvWAIx1zInP11uCfbm--
```

The CSV file might contain the following header rows and data rows:

```
sourcePKey,_from.id.sourcePKey,_to.id.sourcePKey
sourcePKey,_from.id.sourcePKey,_to.id.sourcePKey
AF_AFG,AF,AFG
AL_ALA,AL,ALA
ALB_ALB,ALB,ALB
DZ_DZA,DZ,DZA
AS_ASM,AS,ASM
```

The following sample response shows the status of the import job:

```
{
    "jobId": "dd1b2018cb47cef99f8d0f43",
    "state": "INPROGRESS",
    "startTime": 1561367377428,
    "numOfRecordsProcessed": 100,
    "numOfRecordsFailed": 25,
    "numOfRecordsSucceeded": 75
}
```

Import hierarchy relationships

Imports hierarchy relationships for code values and top-level code values into a hierarchy.

POST request

To import hierarchy relationships, submit a POST request with the following URI:

```
/rdm-service/external/v1/import/hierarchy
```

The request contains form-data with two parameters:

file

Specify a CSV file that contains the hierarchy relationships. The CSV file contains one header row with two columns: Code and ParentCode. You can list the related code values below the header.

For example, you might have the following relationships in the CSV file:

```
Code,ParentCode
C1,P1
C2,P2
C1,P3
```

If you import top-level code values, only use the Code column in the header row.

For example, you might have the following top-level code values:

```
Code
P1
P2
P3
```

importSettings

Specify the file specific configuration and container details.

The `importSettings` includes the following attributes:

Field	Type	Description
delimiter	String	Delimiter used to separate values. Value must be `ASTERISK`, `CIRCUMFLEX`, `COLON`, `COMMA`, `PIPE`, `SECTION`, `SEMICOLON`, `SPACE`, or `TAB`.
textQualifier	String	Symbol used to indicate where a text begins and ends. Value must be `DOUBLE_QUOTE`, `SINGLE_QUOTE`, or `NO_QUOTE`.
codepage	String	Code page used for the export file. Value must be `UTF8`.
hierarchyId	String	ID of the hierarchy to which you want to import relationships. **Note:** You can find the ID of assets in Reference 360 or retrieve the IDs by using REST APIs. For more information, see "Asset IDs" on page 117.
parentCodeListId	String	ID of the parent code list. **Note:** If you import top-level code values, you do not need to provide this attribute.
childCodeListId	String	ID of the child code list.
startingRow	String	Line number from which to start importing data. By default, all rows are imported.

POST response

The response contains details of the import job.

The following table describes the attributes in the response:

Field	Type	Description
jobId	String	ID of the job.
state	String	Status of the job. Values are `CREATED`, `INPROGRESS`, `COMPLETED`, `FAILED`, or `STOPPED`.
startTime	Number	Time, in milliseconds, when the job started.
numOfRecordsProcessed	Number	Number of records processed.
numOfRecordsFailed	Number	Number of records failed to import.
numOfRecordsSucceeded	Number	Number of records successfully imported.

POST example

To import hierarchy relationships into a hierarchy, you might use the following request:

```
POST https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/import/
hierarchy HTTP/1.1
Content-Type: multipart/form-data; boundary=6o2knFse3p53ty9dmcQvWAIx1zInP11uCfbm
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXXX

--6o2knFse3p53ty9dmcQvWAIx1zInP11uCfbm
Content-Disposition: form-data; name=file; filename=import.csv
Content-Type: text/plain
```

```
--6o2knFse3p53ty9dmcQvWAIx1zInP11uCfbm
Content-Disposition: form-data; name=importSettings
Content-Type: application/json;charset=UTF-8

{
  "delimiter":"COMMA",
  "textQualifier":"DOUBLE_QUOTE",
  "startingRow":0,
  "codepage":"UTF8",
  "hierarchyId":"c79ab91c19b13b11d8d43770",
  "childCodeListId":"96f06071e4aaea81ff203abe",
  "parentCodeListId":"9b300f793470882ca23e6091"
}
--6o2knFse3p53ty9dmcQvWAIx1zInP11uCfbm--
```

The CSV file might contain the following header row and data rows:

```
Code,ParentCode
C1,P1
C2,P2
C3,P3
```

To import top-level code values into a hierarchy, you might use the following request:

```
POST https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/import/
hierarchy HTTP/1.1
Content-Type: multipart/form-data; boundary=6o2knFse3p53ty9dmcQvWAIx1zInP11uCfbm
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXXX

--6o2knFse3p53ty9dmcQvWAIx1zInP11uCfbm
Content-Disposition: form-data; name=file; filename=import.csv
Content-Type: text/plain

--6o2knFse3p53ty9dmcQvWAIx1zInP11uCfbm
Content-Disposition: form-data; name=importSettings
Content-Type: application/json;charset=UTF-8

{
  "delimiter":"COMMA",
  "textQualifier":"DOUBLE_QUOTE",
  "startingRow":0,
  "codepage":"UTF8",
  "hierarchyId":"c79ab91c19b13b11d8d43770",
  "childCodeListId":"96f06071e4aaea81ff203abe"
}
--6o2knFse3p53ty9dmcQvWAIx1zInP11uCfbm--
```

The CSV file might contain the following header row and data rows:

```
Code
P1
P2
P3
```

The following sample response shows that status of the import job:

```
{
  "jobId":"73580d323feb170be5ec0fd5",
  "state":"INPROGRESS",
  "startTime":1603092643055,
  "numOfRecordsProcessed":100,
  "numOfRecordsFailed":25,
  "numOfRecordsSucceeded":75
}
```

Get import job status

Retrieves the status of an import job.

GET request

To get the status of an import job, submit a GET request with the following URI:

```
/rdm-service/external/v1/import/job/<job ID>
```

GET response

The response contains the details of the import job, such as the status of the import job, start time, and number of records processed for import.

The following table describes the attributes in the response:

Field	Type	Description
jobId	String	ID of the job.
state	String	Status of the job. Values are `CREATED`, `INPROGRESS`, `COMPLETED`, `FAILED`, or `STOPPED`.
startTime	Number	Time, in milliseconds, when the job started.
numOfRecordsProcessed	Number	Number of records processed.
numOfRecordsFailed	Number	Number of records failed to import.
numOfRecordsSucceeded	Number	Number of records successfully imported.

GET example

To get the status of an import job, you might use the following request:

```
GET https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/import/job/
dd1b2018cb47cef99f8d0f42 HTTP/1.1
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXXX
```

The following sample response shows the status of an import job:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 193

{
  "jobId":"dd1b2018cb47cef99f8d0f42",
  "state":"INPROGRESS",
  "startTime":1561367376330,
  "numOfRecordsProcessed":100,
  "numOfRecordsFailed":25,
  "numOfRecordsSucceeded":75
}
```

Get failed import job report

Retrieves an error report for a failed import job.

By default, the error report shows the first 100,000 records. To retrieve more records or to view the next page of records in the error report, use the query parameters.

GET request

To retrieve an error report for a failed import job, submit a GET request with the following URI:

```
/rdm-service/external/v1/import/job/<job Id>/errorDetails
```

To retrieve the paginated error report, submit a GET request with the following query parameters appended to the URI:

```
/rdm-service/external/v1/import/job/<job Id>/errorDetails?pageNum=<page
number>&recordsPerPage=<records per page>
```

GET request query parameters

You can append the query parameters to the URI to retrieve paginated errors.

The following table lists the query parameters:

Parameter	Description
pageNum	Optional. Page number to display. Default value is `0`.
recordsPerPage	Optional. Number of records to display per page. Default value is `100000`.

GET response

The response contains the error details, such as the line where the error occurred and the reason.

The following table describes the attributes in the response:

Field	Type	Description
jobId	String	ID of the job.
entityType	String	Type of entity imported. Value is `BusinessEntity` or `Relationship`.
fileName	String	Name of the file. Value must end with the `.csv` file extension.
entityName	String	Name of the entity.
errorDetails	-	Includes the error details.
lineNumber	Number	Line number where the error occurred.
entitySourcePkey	String	Column identifier.
reasons	Array	Explanation of failure.

GET example

To retrieve an error report for a failed import job, you might use the following request:

```
GET https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/import/job/
dae4301e9369c16c08bf0881/errorDetails HTTP/1.1
```

To retrieve the second page of records in a paginated error report with 100 records per page, you might use the following request:

```
GET https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/import/job/
dae4301e9369c16c08bf0881/errorDetails?pageNum=2&recordsPerPage=100 HTTP/1.1
```

The following sample response shows the error report for a failed code values import job:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 395
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXX

{
  "jobId":"5984980d317a4b00cfe18880",
  "entityType":"BusinessEntity",
  "fileName":"import.csv",
  "entityName":"rdm.value.be.442ac56e11d5fc9bb11f6a3f",
  "errorDetails":[
    {
      "lineNumber":1,
      "entitySourcePkey":"Code-1101",
      "reasons":[
        "The code value INX does not exist in the picklist that is in the path Country.
Specify a code value from the picklist."
      ]
    }
  ]
}
```

The following sample response shows the error report for a failed value mappings import job:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 354

{
  "jobId":"dae4301e9369c16c08bf0881",
  "entityType":"Relationship",
  "fileName":"import.csv",
  "entityName":"rdm.crosswalk.rel.21ffd6b5f92d10c744acc27c.fc66c441288cf898c6fe5023",
  "errorDetails":[
    {
      "lineNumber":1,
      "entitySourcePkey":"AF_AFG",
      "reasons":[
        "The requested resource with ID 'AFG' does not exist."
      ]
    }
  ]
}
```

import version 2

Use this resource to import code values, value mappings, and hierarchy relationships.

Import code values

Imports code values with optional custom headers into a code list.

Note: Validation checks are not performed when you import code values.

POST request

To import code values into a code list, submit a POST request with the following URI:

```
/rdm-service/external/v2/import
```

The request contains form-data with the following parameters:

file

Specify a CSV file that contains the code value attributes. The columns specified depend on your data model. The CSV must start with one or two header rows, followed by the data rows. If the occurrence of header row is one in the import CSV file, you must set the repeatHeaders field to false.

Note: You can provide a single header row in the CSV file as it no longer requires two header rows.

For example, you might have the following code values:

```
CountryName,CountryCode
CountryName,CountryCode
Afghanistan,AFG
Aland Islands,ALA
Albania,ALB
Algeria,DZA
American Samoa,ASM
```

Note: You can provide additional information about a code value. For example, you might want to assign a code value to the Approved status. In the `status.key` column for the code value, enter the key value for the system reference data value that you want to assign. For more information about your configured system reference data values, see "List system reference data values" on page 234.

importSettings

Specify the file specific configuration and container details.

The `importSettings` parameter includes the following attributes:

Field	Type	Description
delimiter	String	Delimiter used to separate values. Value must be `COMMA`, `SEMICOLON`, `SPACE`, or `TAB`.
textQualifier	String	Symbol used to indicate where a text begins and ends. Value must be `DOUBLE_QUOTE`, `SINGLE_QUOTE`, or `NO_QUOTE`.
codepage	String	Code page used for the import file. Value must be `UTF8` or `MS_WINDOWS`.
dateFormat	String	Format used for dates. Use one of the following formats: - `DE`. For dd.mm.yyyy format. - `ISO`. For yyyy-mm-dd format. - `US`. For mm/dd/yyyy format.
containerType	String	Type of asset that contains code values. Value must be `codelist`.
containerId	String	The ID of the code list to which you want to import code values. **Note:** You can find the ID of assets in Reference 360 or retrieve the IDs by using REST APIs. For more information, see "Asset IDs" on page 117.
startingRow	String	Line number from which to start importing data. By default, all rows are imported.

Field	Type	Description
repeatHeaders	Boolean	Optional. Indicates whether the file has two headers. Value is `true` or `false`. Default is `true`.
mappings	Object	Optional. Maps the column headers in the CSV file with the target attributes of code lists in the key-value format. - You can map the user-defined column name to source field name if required. - You can either choose to ignore the mapping or map all the columns in an import CSV file. **Note:** The key represents the header in the CSV file, and the value represents the field names in Reference 360.

POST response

The response contains the details of the import job.

The response contains the following attributes:

Field	Type	Description
jobId	String	ID of the job.
state	String	Status of the job. Values are `CREATED`, `INPROGRESS`, `COMPLETED`, `FAILED`, or `STOPPED`.
startTime	Number	Time, in milliseconds, when the job started.
numOfRecordsProcessed	Number	Number of records processed.
numOfRecordsFailed	Number	Number of records failed to import.
numOfRecordsSucceeded	Number	Number of records successfully imported.

POST example

To import code values, you might use the following request:

```
POST https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v2/import HTTP/1.1
Content-Type: multipart/form-data; boundary=6o2knFse3p53ty9dmcQvWAIx1zInP11uCfbm
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXXX

--6o2knFse3p53ty9dmcQvWAIx1zInP11uCfbm
Content-Disposition: form-data; name=file; filename=import-code-values.csv

--6o2knFse3p53ty9dmcQvWAIx1zInP11uCfbm
Content-Disposition: form-data; name=importSettings
Content-Type: application/json;charset=UTF-8

{
   "delimiter":"COMMA",
   "textQualifier":"DOUBLE_QUOTE",
   "startingRow":null,
   "codepage":"UTF8",
   "dateFormat":"ISO",
   "containerType":"CODELIST",
   "containerId":"c31851709a749e2d53188ec0",
   "repeatHeaders":true,
   "mappings":{
      "CountryName":"Name",
      "CountryCode":"Code"
```

```
   }
}
--6o2knFse3p53ty9dmcQvWAIx1zInP11uCfbm--
```

The CSV file might contain the following header rows and data rows:

```
CountryName,CountryCode
CountryName,CountryCode
Afghanistan,AFG
Aland Islands,ALA
Albania,ALB
Algeria,DZA
American Samoa,ASM
```

The following sample response shows the status of the import job:

```
{
   "jobId":"32121c139c84a8edc8696c0c",
   "state":"INPROGRESS",
   "startTime":1631086520325,
   "numOfRecordsProcessed":100,
   "numOfRecordsFailed":25,
   "numOfRecordsSucceeded":75
}
```

Import crosswalk values

Imports value mappings into a crosswalk.

POST request

To import value mappings into a crosswalk, submit a POST request with the following URI:

```
/rdm-service/external/v2/import
```

The request contains form-data with two parameters:

file

Specify a CSV file that contains the code value attributes. The columns specified depend on your data model. The CSV must start with one or two header rows, followed by the data rows. If the occurrence of header row is one in the import CSV file, you must set the repeatHeaders field to false.

Note: You can provide a single header row in the CSV file as it no longer requires two header rows.

For example, you might have the following code values:

```
key,from, to
key,from, to
AF_AFG,AF,AFG
AL_ALA,AL,ALA
ALB_ALB,ALB,ALB
DZ_DZA,DZ,DZA
AS_ASM,AS,ASM
```

importSettings

Specify the file specific configuration and container details.

The `importSettings` includes the following attributes:

Field	Type	Description
delimiter	String	Delimiter used to separate values. Value must be `COMMA`, `SEMICOLON`, `SPACE`, or `TAB`.
textQualifier	String	Symbol used to indicate where a text begins and ends. Value must be `DOUBLE_QUOTE`, `SINGLE_QUOTE`, or `NO_QUOTE`.
codepage	String	Code page used for the import file. Value must be `UTF8` or `MS_WINDOWS`.
dateFormat	String	Format used for dates. Use one of the following formats: - `DE`. For dd.mm.yyyy format. - `ISO`. For yyyy-mm-dd format. - `US`. For mm/dd/yyyy format.
containerType	String	Type of asset that contains value mappings. Value must be `crosswalk`.
containerId	String	The ID of the crosswalk to which you want to import value mappings. **Note:** You can find the ID of assets in Reference 360 or retrieve the IDs by using REST APIs. For more information, see "Asset IDs" on page 117.
startingRow	String	Line number from which to start importing data. By default, all rows are imported.
repeatHeaders	Boolean	Optional. Indicates whether the file has two headers. Value is `true` or `false`. Default is `true`.
mappings	Object	Optional. Maps the column headers in the CSV file with the target attributes of code lists in the key-value format. - You can map the user-defined column name to source field name if required. - You can either choose to ignore the mapping or map all the columns in an import CSV file. **Note:** The key represents the header in the CSV file, and the value represents the field names in Reference 360.

POST response

The response contains the details of the import job.

The following table describes the attributes in the response:

Field	Type	Description
jobId	String	ID of the job.
state	String	Status of the job. Values are `CREATED`, `INPROGRESS`, `COMPLETED`, `FAILED`, or `STOPPED`.
startTime	Number	Time, in milliseconds, when the job started.
numOfRecordsProcessed	Number	Number of records processed.
numOfRecordsFailed	Number	Number of records failed to import.
numOfRecordsSucceeded	Number	Number of records successfully imported.

POST example

To import value mappings into a crosswalk, you might use the following request:

```
POST https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v2/import HTTP/1.1
Content-Type: multipart/form-data; boundary=6o2knFse3p53ty9dmcQvWAIx1zInP11uCfbm
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXXX

--6o2knFse3p53ty9dmcQvWAIx1zInP11uCfbm
Content-Disposition: form-data; name=file; filename=import-value-mappings.csv

--6o2knFse3p53ty9dmcQvWAIx1zInP11uCfbm
Content-Disposition: form-data; name=importSettings
Content-Type: application/json;charset=UTF-8

{
   "delimiter":"COMMA",
   "textQualifier":"DOUBLE_QUOTE",
   "startingRow":null,
   "codepage":"UTF8",
   "dateFormat":"ISO",
   "containerType":"CROSSWALK",
   "containerId":"c31851709a749e2d53188ec0",
   "repeatHeaders":true,
   "mappings":{
      "key":"sourcePKey",
      "from":"_from.id.sourcePKey",
      "to":"_to.id.sourcePKey"
   }
}--6o2knFse3p53ty9dmcQvWAIx1zInP11uCfbm--
```

The CSV file might contain the following header rows and data rows:

```
key,from, to
key,from, to
AF_AFG,AF,AFG
AL_ALA,AL,ALA
ALB_ALB,ALB,ALB
DZ_DZA,DZ,DZA
AS_ASM,AS,ASM
```

The following sample response shows the status of the import job:

```
{
   "jobId":"32121c139c84a8edc8696c0c",
   "state":"INPROGRESS",
   "startTime":1631086520406,
   "numOfRecordsProcessed":100,
   "numOfRecordsFailed":25,
   "numOfRecordsSucceeded":75
}
```

Import hierarchy relationships

Imports hierarchy relationships for code values and top-level code values into a hierarchy, with custom headers as optional settings.

POST request

To import hierarchy relationships, submit a POST request with the following URI:

```
/rdm-service/external/v2/import/hierarchy
```

The request contains form-data with two parameters:

file

Specify a CSV file that contains the hierarchy relationships. The CSV file contains one header row with two columns: Code and ParentCode. You can list the related code values below the header.

For example, you might have the following relationships in the CSV file:

```
Code,ParentCode
C1,P1
C2,P2
C1,P3
```

If you import top-level code values, only use the Code column in the header row.

For example, you might have the following top-level code values:

```
Code
P1
P2
P3
```

importSettings

Specify the file specific configuration and container details.

The `importSettings` includes the following attributes:

Field	Type	Description
delimiter	String	Delimiter used to separate values. Value must be `COMMA`, `SEMICOLON`, `SPACE`, or `TAB`.
textQualifier	String	Symbol used to indicate where a text begins and ends. Value must be `DOUBLE_QUOTE`, `SINGLE_QUOTE`, or `NO_QUOTE`.
codepage	String	Code page used for the import file. Value must be `UTF8` or `MS_WINDOWS`.
hierarchyId	String	ID of the hierarchy to which you want to import relationships. **Note:** You can find the ID of assets in Reference 360 or retrieve the IDs by using REST APIs. For more information, see "Asset IDs" on page 117.
parentCodeListId	String	ID of the parent code list. **Note:** If you import top-level code values, you do not need to provide this attribute.
childCodeListId	String	ID of the child code list.
startingRow	String	Line number from which to start importing data. By default, all rows are imported.
repeatHeaders	Boolean	Optional. Indicates whether the file has two headers. Value is `true` or `false`. Default is `true`.
mappings	Object	Optional. Maps the column headers in the CSV file with the target attributes of code lists in the key-value format. - You can map the user-defined column name to source field name if required. - You can either choose to ignore the mapping or map all the columns in an import CSV file. **Note:** The key represents the header in the CSV file, and the value represents the field names in Reference 360.

POST response

The response contains details of the import job.

The following table describes the attributes in the response:

Field	Type	Description
jobId	String	ID of the job.
state	String	Status of the job. Values are `CREATED`, `INPROGRESS`, `COMPLETED`, `FAILED`, or `STOPPED`.
startTime	Number	Time, in milliseconds, when the job started.
numOfRecordsProcessed	Number	Number of records processed.
numOfRecordsFailed	Number	Number of records failed to import.
numOfRecordsSucceeded	Number	Number of records successfully imported.

POST example

To import hierarchy relationships into a hierarchy, you might use the following request:

```
POST https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v2/import/
hierarchy HTTP/1.1
Content-Type: multipart/form-data; boundary=6o2knFse3p53ty9dmcQvWAIx1zInP11uCfbm
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXXX

--6o2knFse3p53ty9dmcQvWAIx1zInP11uCfbm
Content-Disposition: form-data; name=file; filename=import.csv
Content-Type: text/plain

--6o2knFse3p53ty9dmcQvWAIx1zInP11uCfbm
Content-Disposition: form-data; name=importSettings
Content-Type: application/json;charset=UTF-8

{
   "delimiter":"COMMA",
   "textQualifier":"DOUBLE_QUOTE",
   "startingRow":0,
   "codepage":"UTF8",
   "hierarchyId":"eeb38899dd463330df844b8e",
   "childCodeListId":"3ebff6c89e03c41d5dd08acc",
   "parentCodeListId":"68c3ba1b4c3402a7cc18169a",
   "mappings":{
      "state":"Code",
      "country":"ParentCode"
   }
}--6o2knFse3p53ty9dmcQvWAIx1zInP11uCfbm--
```

The CSV file might contain the following header row and data rows:

```
Code,ParentCode
C1,P1
C2,P2
C3,P3
```

To import top-level code values into a hierarchy, you might use the following request:

```
POST https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v2/import/
hierarchy HTTP/1.1
Content-Type: multipart/form-data; boundary=6o2knFse3p53ty9dmcQvWAIx1zInP11uCfbm
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXXX

--6o2knFse3p53ty9dmcQvWAIx1zInP11uCfbm
Content-Disposition: form-data; name=file; filename=import.csv
Content-Type: text/plain

--6o2knFse3p53ty9dmcQvWAIx1zInP11uCfbm
```

```
Content-Disposition: form-data; name=importSettings
Content-Type: application/json;charset=UTF-8

{
   "delimiter":"COMMA",
   "textQualifier":"DOUBLE_QUOTE",
   "startingRow":0,
   "codepage":"UTF8",
   "hierarchyId":"eeb38899dd463330df844b8e",
   "childCodeListId":"3ebff6c89e03c41d5dd08acc",
   "parentCodeListId":"68c3ba1b4c3402a7cc18169a",
   "mappings":{
      "state":"Code",
      "country":"ParentCode"
   }
}--6o2knFse3p53ty9dmcQvWAIx1zInP11uCfbm--
```

The CSV file might contain the following header row and data rows:

```
Code
P1
P2
P3
```

The following sample response shows that status of the import job:

```
{
   "jobId":"32121c139c84a8edc8696c0c",
   "state":"INPROGRESS",
   "startTime":1631086520353,
   "numOfRecordsProcessed":100,
   "numOfRecordsFailed":25,
   "numOfRecordsSucceeded":75
}
```

export version 1

Use this resource to export data to CSV or JSON format. You can export code values in a code list and value mappings in a crosswalk.

Export code values to a CSV file

Exports the code values in a code list to a CSV file.

POST request

To export code values in a code list, submit a POST request with the following URI:

```
/rdm-service/external/v1/export
```

Note: In the request header, you must specify the `Accept` attribute to `application/octet-stream`.

Use the following parameters in the request body to specify the code list to export:

Field	Type	Description
delimiter	String	Delimiter used to separate values. Value must be `ASTERISK`, `CIRCUMFLEX`, `COLON`, `COMMA`, `PIPE`, `SECTION`, `SEMICOLON`, `SPACE`, or `TAB`.
codepage	String	Code page used for the export file. Value must be `UTF8`.
decimalSeparator	String	Decimal separator used for numbers. Value must be `COMMA` or `DOT`.
thousandSeparator	String	Grouping separator used for numbers. Value must be `COMMA`, `DOT`, `SPACE`, `SINGLEQUOTE`, or `NONE`.
dateFormat	String	Format used for dates. Use one of the following formats: - `DE`. For dd.mm.yyyy format. - `ISO`. For yyyy-mm-dd format. - `US`. For mm/dd/yyyy format.
filename	String	File name for the exported file. Value must end with the `.csv` file extension.
containerType	String	Type of asset that contains code values. Value must be `codelist`.
containerId	String	The ID of the code list. **Note:** You can find the ID of assets in Reference 360 or retrieve the IDs by using REST APIs. For more information, see "Asset IDs" on page 117.
filter	Object	Optional. Filter criteria for exporting filtered code values. For more information, see "Filter criteria" on page 240.
columns	Array	Optional. Attribute columns you want to export. If you do not specify attribute columns, the export includes all attribute columns.
excludeParentId	Boolean	Optional. Indicates whether to include the parent ID of code values in the CSV file. Value is `true` or `false`.

POST response

The response is a CSV file. The CSV file contains two header rows followed by data rows.

The CSV file contains the following columns: `status.key` and `effectiveDate`. Based on your data model, you might have additional columns in the file.

POST example

To export the code values in a code list, you might use the following request:

```
POST https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/export HTTP/1.1
Content-Type: application/json
Accept: application/octet-stream
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXXX

{
  "delimiter":"SEMICOLON",
  "codepage":"UTF8",
  "decimalSeparator":"COMMA",
  "thousandSeparator":"DOT",
  "dateFormat":"ISO",
  "filename":"testdata.csv",
  "containerType":"codelist",
  "containerId":"c42dcf614044646d678bf5af",
  "filter":{
```

```
    "_and":[
      {
        "Name":{
          "_contains":"Jo"
        }
      }
    ]
  },
  "columns":[
    "Name",
    "Code"
  ],
  "excludeParentId":false
}
```

To export code values with `US` in the Name attribute, you might use the following request:

```
POST https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/export HTTP/1.1
Content-Type: application/json
Accept: application/octet-stream
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXXX

{
  "delimiter": "SEMICOLON",
  "codepage": "UTF8",
  "decimalSeparator": "COMMA",
  "thousandSeparator": "DOT",
  "dateFormat": "ISO",
  "filename": "testdata.csv",
  "containerType": "codelist",
  "containerId": "9e42b406d59583f15838adf8",
  "filter": {
    "_and": [
      {
        "Name": {
          "_contains": "US"
        }
      }
    ]
  }
}
```

The following sample response shows the exported data in a CSV file:

```
HTTP/1.1 200 OK
Content-Disposition: attachment;filename=testdata.csv
Content-Type: application/octet-stream
Content-Length: 130

Name,Code
Name0,Code0
Name1,Code1
Name2,Code2
Name3,Code3
Name4,Code4
Name5,Code5
Name6,Code6
Name7,Code7
Name8,Code8
Name9,Code9
```

For more information about exporting filtered code values, see "Exporting filtered code values" on page 247.

Export code values to JSON format

Exports the code values in a code list to the JSON format.

POST request

To export code values in a code list, submit a POST request with the following URI:

```
/rdm-service/external/v1/export
```

Note: In the request header, you must specify the `Accept` attribute to `application/json`.

Use the following parameters in the request body to specify the code list to export:

Field	Type	Description
dateFormat	String	Format used for dates. Use one of the following formats: - `DE`. For dd.mm.yyyy format. - `ISO`. For yyyy-mm-dd format. - `US`. For mm/dd/yyyy format.
containerType	String	Type of asset that contains code values. Value must be `codelist`.
containerId	String	The ID of the code list. **Note:** You can find the ID of assets in Reference 360 or retrieve the IDs by using REST APIs. For more information, see "Asset IDs" on page 117.

POST response

The response is in JSON format.

POST example

To export the code values in a code list, you might use the following request:

```
POST https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/export HTTP/1.1
Content-Type: application/json
Accept: application/json
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXXX

{
  "dateFormat" : "ISO",
  "containerType" : "codelist",
  "containerId" : "1c03a6555058fdd134f7f417"
}
```

The following sample response shows the exported data in JSON format:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 292

{
  "content":[
    {
      "Code":"UNIVERSITY-001",
      "children":[
        {
          "Code":"STUDENT-1001",
          "fields":{
            "Code":"STUDENT-1001",
            "Name":"BKImL"
          }
        }
      ],
      "fields":{
```

```
        "Code":"UNIVERSITY-001",
        "Name":"ABC University"
      }
    }
  ]
}
```

Export value mappings to a CSV file

Exports the value mappings in a crosswalk to a CSV file.

POST request

To export value mappings in a crosswalk, submit a POST request with the following URI:

```
/rdm-service/external/v1/export
```

Note: In the request header, you must specify the `Accept` attribute to `application/octet-stream`.

Use the following parameters in the request body to specify the crosswalk to export:

Field	Type	Description
delimiter	String	Delimiter used to separate values. Value must be `ASTERISK`, `CIRCUMFLEX`, `COLON`, `COMMA`, `PIPE`, `SECTION`, `SEMICOLON`, `SPACE`, or `TAB`.
codepage	String	Code page used for the export file. Value must be `UTF8`.
decimalSeparator	String	Decimal separator used for numbers. Value must be `COMMA` or `DOT`.
thousandSeparator	String	Grouping separator used for numbers. Value must be `COMMA`, `DOT`, `SPACE`, `SINGLEQUOTE`, or `NONE`.
dateFormat	String	Format used for dates. Use one of the following formats: - `DE`. For dd.mm.yyyy format. - `ISO`. For yyyy-mm-dd format. - `US`. For mm/dd/yyyy format.
filename	String	File name for the exported file. Value must end with the `.csv` file extension.
containerType	String	Type of asset that contains value mappings. Value must be `crosswalk`.
containerId	String	The ID of the crosswalk. **Note:** You can find the ID of assets in Reference 360 or retrieve the IDs by using REST APIs. For more information, see "Asset IDs" on page 117.

POST response

The response is a CSV file. The CSV file contains two header rows followed by data rows.

The CSV file uses the following columns: `status.key` and `effectiveDate`. Based on your data model, you might have additional columns in the file.

POST example

To export the value mappings in a crosswalk, you might use the following request:

```
POST https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/export HTTP/1.1
Content-Type: application/json
Accept: application/octet-stream
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXXX
```

```
{
  "delimiter" : "SEMICOLON",
  "codepage" : "UTF8",
  "decimalSeparator" : "COMMA",
  "thousandSeparator" : "DOT",
  "dateFormat" : "ISO",
  "filename" : "testdata.csv",
  "containerType" : "crosswalk",
  "containerId" : "5d123f6e4077c700010d59e4"
}
```

The following sample response shows the exported data in a CSV file:

```
HTTP/1.1 200 OK
Content-Disposition: attachment;filename=testdata.csv
Content-Type: application/octet-stream
Content-Length: 578

sourcePKey,_from.id.sourcePKey,_to.id.sourcePKey
sourcePKey,_from.id.sourcePKey,_to.id.sourcePKey
sourceCode0_targetCode0,sourceCode0,targetCode0
sourceCode1_targetCode1,sourceCode1,targetCode1
sourceCode2_targetCode2,sourceCode2,targetCode2
sourceCode3_targetCode3,sourceCode3,targetCode3
sourceCode4_targetCode4,sourceCode4,targetCode4
sourceCode5_targetCode5,sourceCode5,targetCode5
sourceCode6_targetCode6,sourceCode6,targetCode6
sourceCode7_targetCode7,sourceCode7,targetCode7
sourceCode8_targetCode8,sourceCode8,targetCode8
sourceCode9_targetCode9,sourceCode9,targetCode9
```

Export value mappings to JSON format

Exports the value mappings in a crosswalk to the JSON format.

POST request

To export value mappings in a crosswalk, submit a POST request with the following URI:

```
/rdm-service/external/v1/export
```

Note: In the request header, you must specify the `Accept` attribute to `application/json`.

Use the following parameters in the request body to specify the crosswalk to export:

Field	Type	Description
dateFormat	String	Format used for dates. Use one of the following formats: - `DE`. For dd.mm.yyyy format. - `ISO`. For yyyy-mm-dd format. - `US`. For mm/dd/yyyy format.
containerType	String	Type of asset that contains code values. Value must be `crosswalk`.
containerId	String	The ID of the crosswalk. **Note:** You can find the ID of assets in Reference 360 or retrieve the IDs by using REST APIs. For more information, see "Asset IDs" on page 117.

POST response

The response is in JSON format.

POST example

To export the value mappings in a crosswalk, you might use the following request:

```
POST https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/export HTTP/1.1
Content-Type: application/json
Accept: application/json
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXX

{
  "dateFormat" : "ISO",
  "containerType" : "crosswalk",
  "containerId" : "1c03a6555058fdd134f7f417"
}
```

The following sample response shows the exported data:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 234

{
  "content":[
    {
      "fields":{
        "toCode":"DEU",
        "fromCode":"DE"
      },
      "sourcePKey":"DE_DEU"
    },
    {
      "fields":{
        "toCode":"AFG",
        "fromCode":"AF"
      },
      "sourcePKey":"AF_AFG"
    }
  ]
}
```

export version 2

Use this resource to export data to CSV or JSON format. You can export code values in a code list, value mappings in a crosswalk, or relationships in a hierarchy.

Export code values to a CSV file

Exports the code values in a code list to a CSV file.

POST request

To export code values in a code list, submit a POST request with the following URI:

```
/rdm-service/external/v2/export
```

Note: In the request header, you must specify the `Accept` attribute to `application/octet-stream`.

Use the following parameters in the request body to specify the code list to export:

Field	Type	Description
delimiter	String	Delimiter used to separate values. Value must be `ASTERISK`, `CIRCUMFLEX`, `COLON`, `COMMA`, `PIPE`, `SECTION`, `SEMICOLON`, `SPACE`, or `TAB`.
codepage	String	Code page used for the export file. Value must be `UTF8`.
decimalSeparator	String	Decimal separator used for numbers. Value must be `COMMA` or `DOT`.
thousandSeparator	String	Grouping separator used for numbers. Value must be `COMMA`, `DOT`, `SPACE`, `SINGLEQUOTE`, or `NONE`.
dateFormat	String	Format used for dates. Use one of the following formats: - `DE`. For dd.mm.yyyy format. - `ISO`. For yyyy-mm-dd format. - `US`. For mm/dd/yyyy format.
filename	String	File name for the exported file. Value must end with the `.csv` file extension.
containerType	String	Type of asset that contains code values. Value must be `codelist`.
containerId	String	The ID of the code list. **Note:** You can find the ID of assets in Reference 360 or retrieve the IDs by using REST APIs. For more information, see "Asset IDs" on page 117.
filter	Object	Optional. Filter criteria for exporting filtered code values. For more information, see "Filter criteria" on page 240.
columns	Array	Optional. Attribute columns you want to export. If you do not specify attribute columns, the export includes all attribute columns.
fieldName	String	Optional. Name of attribute column to include in the exported file.
excludeParentId	Boolean	Optional. Indicates whether to include the parent ID of code values in the CSV file. Value is `true` or `false`.

POST response

The response is a CSV file. The CSV file contains two header rows followed by data rows.

The CSV file contains the following columns: `status.key` and `effectiveDate`. Based on your data model, you might have additional columns in the file.

POST example

To export the code values in a code list, you might use the following request:

```
POST https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v2/export HTTP/1.1
Content-Type: application/json
Accept: application/octet-stream
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXXX

{
  "delimiter":"SEMICOLON",
  "codepage":"UTF8",
  "decimalSeparator":"COMMA",
  "thousandSeparator":"DOT",
  "dateFormat":"ISO",
  "filename":"testdata.csv",
  "containerType":"codelist",
```

```
  "containerId":"dd318e233414acf05ffa451b",
  "columns":[
    {
      "fieldName":"Name"
    },
    {
      "fieldName":"Code"
    }
  ],
  "excludeParentId":false
}
```

To export code values with `Jo` in the Name attribute, you might use the following request:

```
POST https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v2/export HTTP/1.1
Content-Type: application/json
Accept: application/octet-stream
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXXX

{
  "delimiter":"SEMICOLON",
  "codepage":"UTF8",
  "decimalSeparator":"COMMA",
  "thousandSeparator":"DOT",
  "dateFormat":"ISO",
  "filename":"testdata.csv",
  "containerType":"codelist",
  "containerId":"dd318e233414acf05ffa451b",
  "filter":{
    "_and":[
      {
        "Name":{
          "_contains":"Jo"
        }
      }
    ]
  },
  "columns":[
    {
      "fieldName":"Name"
    },
    {
      "fieldName":"Code"
    }
  ],
  "excludeParentId":false
}
```

The following sample response shows the exported data in a CSV file:

```
HTTP/1.1 200 OK
Content-Disposition: attachment;filename=testdata.csv
Content-Type: application/octet-stream
Content-Length: 130

Name,Code
Name0,Code0
Name1,Code1
Name2,Code2
Name3,Code3
Name4,Code4
Name5,Code5
Name6,Code6
Name7,Code7
Name8,Code8
Name9,Code9
```

For more information about exporting filtered code values, see "Exporting filtered code values" on page 247.

Export code values to JSON format

Exports the code values in a code list to the JSON format.

POST request

To export code values in a code list, submit a POST request with the following URI:

```
/rdm-service/external/v2/export
```

Note: In the request header, you must specify the `Accept` attribute to `application/json`.

Use the following parameters in the request body to specify the code list to export:

Field	Type	Description
dateFormat	String	Format used for dates. Use one of the following formats: - `DE`. For dd.mm.yyyy format. - `ISO`. For yyyy-mm-dd format. - `US`. For mm/dd/yyyy format.
containerType	String	Type of asset that contains code values. Value must be `codelist`.
containerId	String	The ID of the code list. **Note:** You can find the ID of assets in Reference 360 or retrieve the IDs by using REST APIs. For more information, see "Asset IDs" on page 117.

POST response

The response is in JSON format.

POST example

To export the code values in a code list, you might use the following request:

```
POST https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v2/export HTTP/1.1
Content-Type: application/json
Accept: application/json
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXX

{
  "dateFormat" : "ISO",
  "containerType" : "codelist",
  "containerId" : "1c03a6555058fdd134f7f417"
}
```

The following sample response shows the exported data in JSON format:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 292

{
  "content":[
    {
      "Code":"UNIVERSITY-001",
      "children":[
        {
          "Code":"STUDENT-1001",
          "fields":{
            "Code":"STUDENT-1001",
            "Name":"BKImL"
          }
        }
      ],
      "fields":{
```

```
            "Code":"UNIVERSITY-001",
            "Name":"ABC University"
        }
    }
  ]
}
```

Export value mappings to a CSV file

Exports the value mappings in a crosswalk to a CSV file.

POST request

To export value mappings in a crosswalk, submit a POST request with the following URI:

```
/rdm-service/external/v2/export
```

Note: In the request header, you must specify the `Accept` attribute to `application/octet-stream`.

Use the following parameters in the request body to specify the crosswalk to export:

Field	Type	Description
delimiter	String	Delimiter used to separate values. Value must be `ASTERISK`, `CIRCUMFLEX`, `COLON`, `COMMA`, `PIPE`, `SECTION`, `SEMICOLON`, `SPACE`, or `TAB`.
codepage	String	Code page used for the export file. Value must be `UTF8`.
decimalSeparator	String	Decimal separator used for numbers. Value must be `COMMA` or `DOT`.
thousandSeparator	String	Grouping separator used for numbers. Value must be `COMMA`, `DOT`, `SPACE`, `SINGLEQUOTE`, or `NONE`.
dateFormat	String	Format used for dates. Use one of the following formats: - `DE`. For dd.mm.yyyy format. - `ISO`. For yyyy-mm-dd format. - `US`. For mm/dd/yyyy format.
filename	String	File name for the exported file. Value must end with the `.csv` file extension.
containerType	String	Type of asset that contains value mappings. Value must be `crosswalk`.
containerId	String	The ID of the crosswalk. **Note:** You can find the ID of assets in Reference 360 or retrieve the IDs by using REST APIs. For more information, see "Asset IDs" on page 117.

POST response

The response is a CSV file. The CSV file contains two header rows followed by data rows.

The CSV file uses the following columns: `status.key` and `effectiveDate`. Based on your data model, you might have additional columns in the file.

POST example

To export the value mappings in a crosswalk, you might use the following request:

```
POST https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v2/export HTTP/1.1
Content-Type: application/json
Accept: application/octet-stream
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXXX
```

```
{
  "delimiter" : "SEMICOLON",
  "codepage" : "UTF8",
  "decimalSeparator" : "COMMA",
  "thousandSeparator" : "DOT",
  "dateFormat" : "ISO",
  "filename" : "testdata.csv",
  "containerType" : "crosswalk",
  "containerId" : "5d123f6e4077c700010d59e4"
}
```

The following sample response shows the exported data in a CSV file:

```
HTTP/1.1 200 OK
Content-Disposition: attachment;filename=testdata.csv
Content-Type: application/octet-stream
Content-Length: 578

sourcePKey,_from.id.sourcePKey,_to.id.sourcePKey
sourcePKey,_from.id.sourcePKey,_to.id.sourcePKey
sourceCode0_targetCode0,sourceCode0,targetCode0
sourceCode1_targetCode1,sourceCode1,targetCode1
sourceCode2_targetCode2,sourceCode2,targetCode2
sourceCode3_targetCode3,sourceCode3,targetCode3
sourceCode4_targetCode4,sourceCode4,targetCode4
sourceCode5_targetCode5,sourceCode5,targetCode5
sourceCode6_targetCode6,sourceCode6,targetCode6
sourceCode7_targetCode7,sourceCode7,targetCode7
sourceCode8_targetCode8,sourceCode8,targetCode8
sourceCode9_targetCode9,sourceCode9,targetCode9
```

Export value mappings to JSON format

Exports the value mappings in a crosswalk to the JSON format.

POST request

To export value mappings in a crosswalk, submit a POST request with the following URI:

```
/rdm-service/external/v2/export
```

Note: In the request header, you must specify the `Accept` attribute to `application/json`.

Use the following parameters in the request body to specify the crosswalk to export:

Field	Type	Description
dateFormat	String	Format used for dates. Use one of the following formats: - `DE`. For dd.mm.yyyy format. - `ISO`. For yyyy-mm-dd format. - `US`. For mm/dd/yyyy format.
containerType	String	Type of asset that contains code values. Value must be `crosswalk`.
containerId	String	The ID of the crosswalk. **Note:** You can find the ID of assets in Reference 360 or retrieve the IDs by using REST APIs. For more information, see "Asset IDs" on page 117.

POST response

The response is in JSON format.

POST example

To export the value mappings in a crosswalk, you might use the following request:

```
POST https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v2/export HTTP/1.1
Content-Type: application/json
Accept: application/json
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXX

{
  "dateFormat" : "ISO",
  "containerType" : "crosswalk",
  "containerId" : "1c03a6555058fdd134f7f417"
}
```

The following sample response shows the exported data:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 234

{
  "content":[
    {
      "fields":{
        "toCode":"DEU",
        "fromCode":"DE"
      },
      "sourcePKey":"DE_DEU"
    },
    {
      "fields":{
        "toCode":"AFG",
        "fromCode":"AF"
      },
      "sourcePKey":"AF_AFG"
    }
  ]
}
```

Export hierarchies to a CSV file

Exports a hierarchy to a CSV file.

POST request

To export a hierarchy, submit a POST request with the following URI:

```
/rdm-service/external/v2/export
```

Note: In the request header, you must specify the `Accept` attribute to `application/octet-stream`.

Use the following parameters in the request body to specify the hierarchy to export:

Field	Type	Description
delimiter	String	Delimiter used to separate values. Value must be `ASTERISK`, `CIRCUMFLEX`, `COLON`, `COMMA`, `PIPE`, `SECTION`, `SEMICOLON`, `SPACE`, or `TAB`.
codepage	String	Code page used for the export file. Value must be `UTF8`.
decimalSeparator	String	Decimal separator used for numbers. Value must be `COMMA` or `DOT`.

Field	Type	Description
thousandSeparator	String	Grouping separator used for numbers. Value must be `COMMA`, `DOT`, `SPACE`, `SINGLEQUOTE`, or `NONE`.
dateFormat	String	Format used for dates. Use one of the following formats: - `DE`. For dd.mm.yyyy format. - `ISO`. For yyyy-mm-dd format. - `US`. For mm/dd/yyyy format.
filename	String	File name for the exported file. Value must end with the `.csv` file extension.
containerType	String	Type of asset that contains code values. Value must be `hierarchy`.
containerId	String	The ID of the hierarchy. **Note:** You can find the ID of assets in Reference 360 or retrieve the IDs by using REST APIs. For more information, see "Asset IDs" on page 117.
filter	Object	Optional. Filter criteria for exporting filtered code values. For more information, see "Filter criteria" on page 240.
columns	Array	Optional. Attribute columns you want to export. If you do not specify attribute columns, the export includes all attribute columns.
fieldName	String	Optional. Name of attribute column to include in the exported file.
codeListId	String	The ID of the code list associated to the attribute.
excludeParentId	Boolean	Optional. Indicates whether to include the parent ID of code values in the CSV file. Value is `true` or `false`.

POST response

The response is a CSV file. The CSV file contains two header rows followed by data rows.

The CSV file uses the following columns: `status.key` and `effectiveDate`. Based on your data model, you might have additional columns in the file.

POST example

To export the code values in a hierarchy, you might use the following request:

```
POST https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v2/export HTTP/1.1
Content-Type: application/json
Accept: application/octet-stream
Content-Length: 456
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXXX

{
  "delimiter":"SEMICOLON",
  "codepage":"UTF8",
  "decimalSeparator":"COMMA",
  "thousandSeparator":"DOT",
  "dateFormat":"ISO",
  "filename":"testdata.csv",
  "containerType":"hierarchy",
  "containerId":"7681b69b2f02a8f81389307d",
  "columns":[
    {
      "fieldName":"Name",
      "codeListId":"5495ed772814d1a29c588599"
    },
```

```
    {
      "fieldName":"Code",
      "codeListId":"5495ed772814d1a29c588599"
    }
  ],
  "excludeParentId":false
}
```

To export code values with `US` in the Name attribute, you might use the following request:

```
POST https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v2/export HTTP/1.1
Content-Type: application/json
Accept: application/octet-stream
Content-Length: 346
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXXX

{
  "delimiter":"SEMICOLON",
  "codepage":"UTF8",
  "decimalSeparator":"COMMA",
  "thousandSeparator":"DOT",
  "dateFormat":"ISO",
  "filename":"testdata.csv",
  "containerType":"crosswalk",
  "containerId":"350e0fcf85d589f5578226e9",
  "filter": {
    "_and": [
      {
        "Name": {
          "_contains": "US"
        }
      }
    ]
  }
}
```

The following sample response shows the exported data in a CSV file:

```
HTTP/1.1 200 OK
Content-Disposition: attachment;filename=testdata.csv
Content-Type: application/octet-stream
Content-Length: 140

Name,Code
Name,Code
Name0,Code0
Name1,Code1
Name2,Code2
Name3,Code3
Name4,Code4
Name5,Code5
Name6,Code6
Name7,Code7
Name8,Code8
Name9,Code9
```

For more information about exporting filtered code values, see "Exporting filtered code values" on page 247.

Export hierarchies to JSON format

Exports a hierarchy to the JSON format.

POST request

To export a hierarchy, submit a POST request with the following URI:

```
/rdm-service/external/v2/export
```

Note: In the request header, you must specify the `Accept` attribute to `application/json`.

Use the following parameters in the request body to specify the hierarchy to export:

Field	Type	Description
dateFormat	String	Format used for dates. Use one of the following formats: - `DE`. For dd.mm.yyyy format. - `ISO`. For yyyy-mm-dd format. - `US`. For mm/dd/yyyy format.
containerType	String	Type of asset that contains code values. Value must be `hierarchy`.
containerId	String	The ID of the code list. **Note:** You can find the ID of assets in Reference 360 or retrieve the IDs by using REST APIs. For more information, see "Asset IDs" on page 117.
excludeParentId	Boolean	Optional. Indicates whether to include the parent ID of code values in the CSV file. Value is `true` or `false`.

POST response

The response is in JSON format.

POST example

To export a hierarchy, you might use the following request:

```
POST https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v2/export HTTP/1.1
Content-Type: application/json
Accept: application/json
Host: localhost:8080
Content-Length: 134
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXXX

{
  "dateFormat":"ISO",
  "containerType":"hierarchy",
  "containerId":"44791d39f4e167b61891438d",
  "excludeParentId":false
}
```

The following sample response shows the exported data in JSON format:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 794

{
  "content":[
    {
      "Code":"root1",
      "children":[
        {
          "Code":"child1_1",
          "fields":{
            "Name":"Name of child 1.1",
            "Code":"child1_1"
          }
        },
        {
          "Code":"child1_2",
          "fields":{
            "Name":"Name of child 1.2",
            "Code":"child1_2"
          }
        }
      ],
      "fields":{
```

```
        "Name":"Name of root 1",
        "Code":"root1"
      }
    },
    {
      "Code":"root2",
      "children":[
        {
          "Code":"child2_1",
          "fields":{
            "Name":"Name of child 2.1",
            "Code":"child2_1"
          }
        },
        {
          "Code":"child2_2",
          "fields":{
            "Name":"Name of child 2.2",
            "Code":"child2_2"
          }
        }
      ],
      "fields":{
        "Name":"Name of root 2",
        "Code":"root2"
      }
    }
  ]
}
```

export version 3

Use this resource to export code values in a code list and value mappings in a crosswalk at a point in time to JSON and CSV formats.

Export code lists at a point in time to JSON format

Exports the code values in a code list at a point in time to the JSON format.

POST request

To export code values in a code list, submit a POST request with the following URI:

```
/rdm-service/external/v3/export
```

Note: In the request header, you must specify the `Accept` attribute to `application/json`.

Use the following parameters in the request body to specify the code list to export:

Field	Type	Description
dateFormat	String	Format used for dates. Use one of the following formats: - `DE`. For dd.mm.yyyy format. - `ISO`. For yyyy-mm-dd format. - `US`. For mm/dd/yyyy format.
containerType	String	Type of asset that contains code values. Value must be `codelist`.

Field	Type	Description
containerId	String	The ID of the code list. **Note:** You can find the ID of assets in Reference 360 or retrieve the IDs by using REST APIs. For more information, see "Asset IDs" on page 117.
filter	Object	Optional. Filter criteria for exporting filtered code values. For more information, see "Filter criteria for export version 3" on page 200.
excludeParentId	Boolean	Optional. Indicates whether to include the parent ID of code values in the JSON file. Value is `true` or `false`. Default is `false`.
pageSize	Number	Optional. Number of records to display on each page. The maximum value is `10000`. Default is `10000`.
page	Number	Optional. Page number to display. Default is `0`.
sort	Array	Optional. Sort criteria for exporting sorted code values. Values must be comma-separated field names prefixed with their sort directions. The plus symbol (+) indicates ascending order, and the minus symbol (-) indicates descending order. For example, `"_sort": ["+Name","-Code"]` indicates to sort the names in the ascending order and codes in the descending order.
pit	String	Optional. Date to retrieve the point in time information about the code list. Use the ISO format: yyyy-mm-dd.
addLabelsForReferenceAttribute	Boolean	Optional. Displays values for reference and dependent attributes based on the display attributes of reference and dependent code lists, respectively.

POST response

The response is in the JSON format.

Note: When you export code values in a code list, the API response does not include the code values that do not contain relationships. The number of exported code values might result in mismatch with the number of imported code values.

POST example

To export the code values in a code list at a point in time, you might use the following request:

```
POST https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v3/export HTTP/1.1
Content-Type: application/json
Accept: application/json
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXXX
Content-Length: 356

{
    "dateFormat": "ISO",
    "containerType": "codelist",
    "containerId": "3dac5bc1b65697fba1ccf8f4",
    "filter": {
        "_and": [{
            "Name": {
                "_eq": "ABC University"
            }
        }]
    },
    "excludeParentId": false,
    "pageSize": 10000,
    "page": 0,
```

```
    "sort": ["+Name", "+Code"],
    "pit": "2022-08-12",
 "repeatHeaders" : false,
 "addLabelsForReferenceAttribute": true
}
```

The following sample response shows the exported data in the JSON format:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 426

{
   "pageSize":10,
   "page":0,
   "totalNumberOfElements":1,
   "numberOfElements":1,
   "lastPage":true,
   "firstPage":true,
   "content":[
      {
         "Code":"GBS",
         "fields":{
            "Description":"Description for GBS",
            "GeoRefAttr":"GER",
            "GeoRefAttr.label":"GERMANY-GER-GERMANYDESC",
            "Code":"GBS",
            "Name":"Global Business Services"
         }
      }
   ]
}
```

Export code lists at point in time to CSV format

Exports the code values in a code list at a point in time to the CSV format.

POST request

To export code values in a code list, submit a POST request with the following URI:

```
/rdm-service/external/v3/export
```

Note: In the request header, you must specify the `Accept` attribute to `application/octet-stream`.

Use the following parameters in the request body to specify the code list to export:

Field	Type	Description
delimiter	String	Optional. Delimiter used to separate values. Value must be `ASTERISK`, `CIRCUMFLEX`, `COLON`, `COMMA`, `PIPE`, `SECTION`, `SEMICOLON`, `SPACE` or `TAB`. Default is `COMMA`.
codepage	String	Optional. Code page used for the export file. Value must be `UTF8` or `MS_WINDOWS`. Default is `UTF8`.
decimalSeparator	String	Optional. Decimal separator used for numbers. Value must be `COMMA` or `DOT`. Default is `DOT`.
thousandSeparator	String	Optional. Grouping separator used for numbers. Value must be `COMMA`, `DOT`, `SPACE`, `SINGLEQUOTE` or `NONE`. Default is `NONE`.

Field	Type	Description
dateFormat	String	Format used for dates. Use one of the following formats: - `DE`. For dd.mm.yyyy format. - `ISO`. For yyyy-mm-dd format. - `US`. For mm/dd/yyyy format.
filename	String	File name for the exported file. Value must end with the `.csv` file extension.
containerType	String	Type of asset that contains code values. Value must be `codelist`.
containerId	String	The ID of the code list. **Note:** You can find the ID of assets in Reference 360 or retrieve the IDs by using REST APIs. For more information, see "Asset IDs" on page 117.
filter	Object	Optional. Filter criteria for exporting filtered code values. For more information, see "Filter criteria for export version 3" on page 200.
columns	Array	Optional. Attribute columns you want to export. If you do not specify attribute columns, the export includes all the attribute columns.
fieldName	String	Optional. Name of attribute column to include in the exported file.
excludeParentId	Boolean	Optional. Indicates whether to include the parent ID of code values in the CSV file. Value is `true` or `false`. Default is `false`.
pageSize	Number	Optional. Number of records to display on each page. The maximum value is `10000`. Default is `10000`.
page	Number	Optional. Page number to display. Default is `0`.
sort	Array	Optional. Sort criteria for exporting sorted code values. Values must be comma-separated field names prefixed with their sort directions. The plus symbol (+) indicates ascending order, and the minus symbol (-) indicates descending order. For example, `"_sort": ["+Name","-Code"]` indicates to sort the names in the ascending order and codes in the descending order.
pit	String	Optional. Date to retrieve the point in time information about the code list. Use the ISO format: yyyy-mm-dd.
repeatHeaders	Boolean	Optional. Indicates whether to populate headers in the CSV response. Value is `true` or `false`. Default is `true`.
addLabelsForReferenceAttribute	Boolean	Optional. Displays values for reference and dependent attributes based on the display attributes of reference and dependent code lists, respectively.

POST response

The response is a CSV file.

Note: When you export code values in a code list, the API response does not include the code values that do not contain relationships. The number of exported code values might result in mismatch with the number of imported code values.

POST example

To export the code values in a code list at a point in time, you might use the following request:

```
POST https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v3/export HTTP/1.1
Content-Type: application/json
Accept: application/octet-stream
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXX
Content-Length: 580

{
   "delimiter":"COMMA",
   "codepage":"UTF8",
   "decimalSeparator":"COMMA",
   "thousandSeparator":"DOT",
   "dateFormat":"ISO",
   "filename":"testdata.csv",
   "containerType":"codelist",
   "containerId":"f6770b657b381038f9e9db3c",
   "filter":{
      "_and":[
         {
            "Name":{
               "_eq":"ABC University"
            }
         }
      ]
   },
   "columns":[
      {
         "fieldName":"Name"
      },
      {
         "fieldName":"Code"
      }
   ],
   "excludeParentId":false,
   "pageSize":10000,
   "page":0,
   "sort":[
      "+Name",
      "+Code"
   ],
   "pit":"2022-08-12",
   "repeatHeaders":false,
   "addLabelsForReferenceAttribute":true
}
```

The following sample response shows the exported data in the CSV format:

```
HTTP/1.1 200 OK
Content-Disposition: attachment;filename=testdata.csv
Content-Type: application/octet-stream
RDM-PAGE: 0
RDM-TOTAL-NUMBER-OF-ELEMENTS: 10
RDM-NUMBER-OF-ELEMENTS: 10
RDM-PAGE-SIZE: 10000
RDM-FIRST-PAGE: true
RDM-LAST-PAGE: true
Content-Length: 950

status.key;effectiveDate;Name;Code;Description;GeoRefAttr.Code;GeoRefAttr.label
status.status.key;effectiveDate;Name;Code;Description;GeoRefAttr.Code;GeoRefAttr.label
;;Global Business Services;GBS;Description for GBS;GER;GERMANY-GER-GERMANYDESC
```

Export value mappings at a point in time to JSON format

Exports the value mappings in a crosswalk at a point in time to the JSON format.

POST request

To export value mappings in a crosswalk, submit a POST request with the following URI:

```
/rdm-service/external/v3/export
```

Note: In the request header, you must specify the `Accept` attribute to `application/json`.

Use the following parameters in the request body to specify the crosswalk to export:

Field	Type	Description
dateFormat	String	Format used for dates. Use one of the following formats: - `DE`. For dd.mm.yyyy format. - `ISO`. For yyyy-mm-dd format. - `US`. For mm/dd/yyyy format.
containerType	String	Type of asset that contains code values. Value must be `crosswalk`.
containerId	String	ID of the crosswalk. **Note:** You can find the ID of assets in Reference 360 or retrieve the IDs by using REST APIs. For more information, see "Asset IDs" on page 117.
pageSize	Number	Optional. Number of records to display on each page. The maximum value is `10000`. Default is `10000`.
page	Number	Optional. Page number to display. Default is `0`.
pit	String	Date to retrieve the point in time information about the crosswalk. Use the ISO format: yyyy-mm-dd.

POST response

The response is in the JSON format.

POST example

To export the value mappings in a crosswalk at a point in time, you might use the following request:

```
POST https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v3/export HTTP/1.1
Content-Type: application/json
Accept: application/json
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXXX
Content-Length: 220

{
   "dateFormat":"ISO",
   "containerType":"crosswalk",
   "containerId":"c27e43cad990f4e57361ae60",
   "pageSize":10000,
   "page":0,
   "pit":"2021-05-21",
   "repeatHeaders":false
}
```

The following sample response shows the exported data in the JSON format:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 366
```

```
{
   "pageSize":2,
   "page":0,
   "totalNumberOfElements":2,
   "numberOfElements":2,
   "lastPage":true,
   "firstPage":true,
   "content":[
      {
         "fields":{
            "toCode":"DEU",
            "fromCode":"DE"
         },
         "sourcePKey":"DE_DEU"
      },
      {
         "fields":{
            "toCode":"AFG",
            "fromCode":"AF"
         },
         "sourcePKey":"AF_AFG"
      }
   ]
}
```


Export value mappings at a point in time to CSV format

Exports the value mappings in a crosswalk at a point in time to the CSV format.

POST request

To export value mappings in a crosswalk, submit a POST request with the following URI:

```
/rdm-service/external/v3/export
```

Note: In the request header, you must specify the `Accept` attribute to `application/octet-stream`.

Use the following parameters in the request body to specify the crosswalk to export:

Field	Type	Description
delimiter	String	Optional. Delimiter used to separate values. Value must be `ASTERISK`, `CIRCUMFLEX`, `COLON`, `COMMA`, `PIPE`, `SECTION`, `SEMICOLON`, `SPACE` or `TAB`. Default is `COMMA`.
codepage	String	Optional. Code page used for the export file. Value must be `UTF8` or `MS_WINDOWS`. Default is `UTF8`.
decimalSeparator	String	Optional. Decimal separator used for numbers. Value must be `COMMA` or `DOT`. Default is `DOT`.
thousandSeparator	String	Optional. Grouping separator used for numbers. Value must be `COMMA`, `DOT`, `SPACE`, `SINGLEQUOTE` or `NONE`. Default is `NONE`.

Field	Type	Description
dateFormat	String	Format used for dates. Use one of the following formats: - `DE`. For dd.mm.yyyy format. - `ISO`. For yyyy-mm-dd format. - `US`. For mm/dd/yyyy format.
filename	String	File name for the exported file. Value must end with the .csv file extension.
containerType	String	Type of asset that contains code values. Value must be `crosswalk`.
containerId	String	ID of the crosswalk. **Note:** You can find the ID of assets in Reference 360 or retrieve the IDs by using REST APIs. For more information, see "Asset IDs" on page 117.
pageSize	Number	Optional. Number of records to display on each page. The maximum value is `10000`. Default is `10000`.
page	Number	Optional. Page number to display. Default is `0`.
pit	String	Date to retrieve the point in time information about the crosswalk. Use the ISO format: yyyy-mm-dd.
repeatHeaders	Boolean	Indicates whether to populate headers in the CSV response.

POST response

The response is in the CSV format.

POST example

To export the value mappings in a crosswalk at a point in time, you might use the following request:

```
POST https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v3/export HTTP/1.1
Content-Type: application/json
Accept: application/octet-stream
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXXX
Content-Length: 338

{
  "delimiter" : "SEMICOLON",
  "codepage" : "UTF8",
  "decimalSeparator" : "COMMA",
  "thousandSeparator" : "DOT",
  "dateFormat" : "ISO",
  "filename" : "testdata.csv",
  "containerType" : "crosswalk",
  "containerId" : "2029dbbcd5596d905d15fa83",
  "pageSize" : 10000,
  "page" : 0,
  "pit" : "2021-06-11",
```

```
  "repeatHeaders" : false
}
```

The following sample response shows the exported data in the CSV format:

```
HTTP/1.1 200 OK
Content-Disposition: attachment;filename=testdata.csv
Content-Type: application/octet-stream
RDM-PAGE: 0
RDM-TOTAL-NUMBER-OF-ELEMENTS: 10
RDM-NUMBER-OF-ELEMENTS: 10
RDM-PAGE-SIZE: 10000
RDM-FIRST-PAGE: true
RDM-LAST-PAGE: true
Content-Length: 578

sourcePKey,_from.id.sourcePKey,_to.id.sourcePKey
sourcePKey,_from.id.sourcePKey,_to.id.sourcePKey
sourceCode0_targetCode0,sourceCode0,targetCode0
sourceCode1_targetCode1,sourceCode1,targetCode1
sourceCode2_targetCode2,sourceCode2,targetCode2
sourceCode3_targetCode3,sourceCode3,targetCode3
sourceCode4_targetCode4,sourceCode4,targetCode4
sourceCode5_targetCode5,sourceCode5,targetCode5
sourceCode6_targetCode6,sourceCode6,targetCode6
sourceCode7_targetCode7,sourceCode7,targetCode7
sourceCode8_targetCode8,sourceCode8,targetCode8
sourceCode9_targetCode9,sourceCode9,targetCode9
```

Export incoming crosswalk mappings to CSV format

Exports incoming crosswalk value mappings to CSV format.

POST request

To export incoming crosswalk value mappings, submit a POST request with the following URI:

```
/rdm-service/external/v3/export
```

Note: In the request header, you must specify the `Accept` attribute to `application/octet-stream`.

Use the following parameters in the request body to specify the incoming crosswalk value mappings to export:

Field	Type	Description
Delimiter	String	Optional. Delimiter used to separate values that belong to a mapping. Value must be `COMMA`, `SEMICOLON`, `SPACE`, `TAB`, `PIPE`, `COLON`, `SECTION`, `CIRCUMFLEX`, or `ASTERISK`. Default is `COMMA`.
codepage	String	Code page used for the export file. Value must be `UTF8` or `MS_WINDOWS`. Default is `UTF8`.
decimalSeparator	String	Optional. Decimal separator used for numbers. Value must be `COMMA` or `DOT`. Default is `DOT`.
thousandSeparator	String	Optional. Grouping separator used for numbers. Value must be `COMMA`, `DOT`, `SPACE`, `SINGLEQUOTE` or `NONE`. Default is `NONE`.

Field	Type	Description
dateFormat	String	Format used for dates. Use one of the following formats: - `DE`. For dd.mm.yyyy format. - `ISO`. For yyyy-mm-dd format. - `US`. For mm/dd/yyyy format.
filename	String	File name for the exported file. Value must end with the `.csv` file extension.
containerType	String	Type of asset that contains incoming value mappings from a codelist. Value must be `codelist_crosswalk`.
containerId	String	ID of the codelist from which the incoming crosswalks must be exported. **Note:** You can find the ID of assets in Reference 360 or retrieve the IDs by using REST APIs. For more information, see "Asset IDs" on page 117.
columns	Array	Optional. Attribute columns you want to export. If you do not specify attribute columns, the export includes all the attribute columns.
fieldName	String	Optional. Name of attribute column to include in the exported file.
excludeParentId	Boolean	Optional. Indicates whether to include the parent ID of value mappings in the CSV file. Value is `true` or `false`. Default is `false`.
pageSize	Number	Optional. Number of records to display on each page. The maximum value is `10000`. Default is `10000`.
page	Number	Optional. Page number to display. Default is `0`.
pit	String	Optional. Date to retrieve the point in time information about the crosswalk. Use the ISO format: yyyy-mm-dd. Default value is the current timestamp in yyyy-mm-dd format.
repeatHeaders	Boolean	Optional. Indicates whether to populate headers in the CSV response. Value is `true` or `false`. Default is `true`.
direction	String	Use `incoming` to export incoming crosswalks.
mappingDelimiter	String	Optional. Delimiter used to separate values from different mappings. Value must be `COMMA`, `SEMICOLON`, `SPACE`, `TAB`, `PIPE`, `COLON`, `SECTION`, `CIRCUMFLEX`, `HYPHEN`, or `ASTERISK`. Default is `COMMA`.

POST response

The response is a CSV file.

POST example

To export the incoming crosswalk value mappings, you might use the following request:

```
POST https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v3/export HTTP/1.1
Content-Type: application/json
Accept: application/octet-stream
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXXX
Content-Length: 517

{
   "delimiter":"COMMA",
   "codepage":"UTF8",
   "decimalSeparator":"COMMA",
```

```
    "thousandSeparator":"DOT",
    "dateFormat":"ISO",
    "filename":"testdata.csv",
    "containerType":"codelist_crosswalk",
    "containerId":"9a34f404a1e836f3a16af3e9",
    "columns":[
       {
          "fieldName":"Name"
       },
       {
          "fieldName":"Code"
       }
    ],
    "excludeParentId":false,
    "pageSize":10000,
    "page":0,
    "pit":"2021-10-21",
    "repeatHeaders":false,
    "direction":"incoming",
    "mappingDelimiter":"PIPE"
}
```

The following sample response shows the exported data in the CSV format:

```
HTTP/1.1 200 OK
Content-Disposition: attachment;filename=testdata.csv
Content-Type: application/octet-stream
Content-Length: 888

CodelistName,IncomingCodelist1Name,IncomingCodelist2Name
codelistCode0,incomingCodelist1Code0|incomingCodelist1Code1,incomingCodelist2Code0
codelistCode1,incomingCodelist1Code1|incomingCodelist1Code2,incomingCodelist2Code1
codelistCode2,incomingCodelist1Code2|incomingCodelist1Code3,incomingCodelist2Code2
codelistCode3,incomingCodelist1Code3|incomingCodelist1Code4,incomingCodelist2Code3
codelistCode4,incomingCodelist1Code4|incomingCodelist1Code5,incomingCodelist2Code4
codelistCode5,incomingCodelist1Code5|incomingCodelist1Code6,incomingCodelist2Code5
codelistCode6,incomingCodelist1Code6|incomingCodelist1Code7,incomingCodelist2Code6
codelistCode7,incomingCodelist1Code7|incomingCodelist1Code8,incomingCodelist2Code7
codelistCode8,incomingCodelist1Code8|incomingCodelist1Code9,incomingCodelist2Code8
codelistCode9,incomingCodelist1Code9|incomingCodelist1Code10,incomingCodelist2Code9
```

Export outgoing crosswalk mappings to CSV format

Exports outgoing crosswalk value mappings to CSV format.

POST request

To export outgoing crosswalk value mappings, submit a POST request with the following URI:

```
/rdm-service/external/v3/export
```

Note: In the request header, you must specify the `Accept` attribute to `application/octet-stream`.

Use the following parameters in the request body to specify the outgoing crosswalk value mappings to export:

Field	Type	Description
Delimiter	String	Optional. Delimiter used to separate values that belong to a mapping. Value must be `COMMA`, `SEMICOLON`, `SPACE`, `TAB`, `PIPE`, `COLON`, `SECTION`, `CIRCUMFLEX`, or `ASTERISK`. Default is `COMMA`.
codepage	String	Code page used for the export file. Value must be `UTF8` or `MS_WINDOWS`. Default is `UTF8`.

Field	Type	Description
decimalSeparator	String	Optional. Decimal separator used for numbers. Value must be `COMMA` or `DOT`. Default is `DOT`.
thousandSeparator	String	Optional. Grouping separator used for numbers. Value must be `COMMA`, `DOT`, `SPACE`, `SINGLEQUOTE` or `NONE`. Default is `NONE`.
dateFormat	String	Format used for dates. Use one of the following formats: - `DE`. For dd.mm.yyyy format. - `ISO`. For yyyy-mm-dd format. - `US`. For mm/dd/yyyy format.
filename	String	File name for the exported file. Value must end with the `.csv` file extension.
containerType	String	Type of asset that contains outgoing value mappings from a codelist. Value must be `codelist_crosswalk`.
containerId	String	ID of the codelist from which the outgoing crosswalks must be exported. **Note:** You can find the ID of assets in Reference 360 or retrieve the IDs by using REST APIs. For more information, see "Asset IDs" on page 117.
columns	Array	Optional. Attribute columns you want to export. If you do not specify attribute columns, the export includes all the attribute columns.
fieldName	String	Optional. Name of attribute column to include in the exported file.
excludeParentId	Boolean	Optional. Indicates whether to include the parent ID of value mappings in the CSV file. Value is `true` or `false`. Default is `false`.
pageSize	Number	Optional. Number of records to display on each page. The maximum value is `10000`. Default is `10000`.
page	Number	Optional. Page number to display. Default is `0`.
pit	String	Optional. Date to retrieve the point in time information about the crosswalk. Use the ISO format: yyyy-mm-dd. Default value is the current timestamp in yyyy-mm-dd format.
repeatHeaders	Boolean	Optional. Indicates whether to populate headers in the CSV response. Value is `true` or `false`. Default is `true`.
direction	String	Use `outgoing` to export outgoing crosswalks.
mappingDelimiter	String	Optional. Delimiter used to separate values from different mappings. Value must be `COMMA`, `SEMICOLON`, `SPACE`, `TAB`, `PIPE`, `COLON`, `SECTION`, `CIRCUMFLEX`, `HYPHEN`, or `ASTERISK`. Default is `COMMA`.

POST response

The response is a CSV file.

POST example

To export the outgoing crosswalk value mappings, you might use the following request:

```
POST https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v3/export HTTP/1.1
Content-Type: application/json
Accept: application/octet-stream
```

```
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXX
Content-Length: 888

{
   "delimiter":"COMMA",
   "codepage":"UTF8",
   "decimalSeparator":"COMMA",
   "thousandSeparator":"DOT",
   "dateFormat":"ISO",
   "filename":"testdata.csv",
   "containerType":"codelist_crosswalk",
   "containerId":"26ab763d543b157a4b12097f",
   "columns":[
      {
         "fieldName":"Name"
      },
      {
         "fieldName":"Code"
      }
   ],
   "excludeParentId":false,
   "pageSize":10000,
   "page":0,
   "pit":"2021-10-13",
   "repeatHeaders":false,
   "direction":"outgoing",
   "mappingDelimiter":"PIPE"
}
```

The following sample response shows the exported data in the CSV format:

```
HTTP/1.1 200 OK
Content-Disposition: attachment;filename=testdata.csv
Content-Type: application/octet-stream
Content-Length: 888

CodelistName,OutgoingCodelist1Name,OutgoingCodelist2Name
codelistCode0,outgoingCodelist1Code0|outgoingCodelist1Code1,outgoingCodelist2Code0
codelistCode1,outgoingCodelist1Code1|outgoingCodelist1Code2,outgoingCodelist2Code1
codelistCode2,outgoingCodelist1Code2|outgoingCodelist1Code3,outgoingCodelist2Code2
codelistCode3,outgoingCodelist1Code3|outgoingCodelist1Code4,outgoingCodelist2Code3
codelistCode4,outgoingCodelist1Code4|outgoingCodelist1Code5,outgoingCodelist2Code4
codelistCode5,outgoingCodelist1Code5|outgoingCodelist1Code6,outgoingCodelist2Code5
codelistCode6,outgoingCodelist1Code6|outgoingCodelist1Code7,outgoingCodelist2Code6
codelistCode7,outgoingCodelist1Code7|outgoingCodelist1Code8,outgoingCodelist2Code7
codelistCode8,outgoingCodelist1Code8|outgoingCodelist1Code9,outgoingCodelist2Code8
codelistCode9,outgoingCodelist1Code9|outgoingCodelist1Code10,outgoingCodelist2Code9
```

Export value mappings to CSV format with display attributes in the header

Exports the value mappings in a crosswalk to CSV format with display attributes instead of codes, in the header.

POST request

To export value mappings in a crosswalk with display attributes in the header, submit a POST request with the following URI:

```
/rdm-service/external/v3/export
```

Note: In the request header, you must specify the `Accept` attribute to `application/octet-stream`.

Use the following parameters in the request body to specify the crosswalk value mappings to export:

Field	Type	Description
Delimiter	String	Optional. Delimiter used to separate values. Value must be `COMMA`, `SEMICOLON`, `SPACE`, `TAB`, `PIPE`, `COLON`, `SECTION`, `CIRCUMFLEX`, or `ASTERISK`. Default is `COMMA`.
codepage	String	Code page used for the export file. Value must be `UTF8` or `MS_WINDOWS`. Default is `UTF8`.
decimalSeparator	String	Optional. Decimal separator used for numbers. Value must be `COMMA` or `DOT`. Default is `DOT`.
thousandSeparator	String	Optional. Grouping separator used for numbers. Value must be `COMMA`, `DOT`, `SPACE`, `SINGLEQUOTE` or `NONE`. Default is `NONE`.
dateFormat	String	Format used for dates. Use one of the following formats: - `DE`. For dd.mm.yyyy format. - `ISO`. For yyyy-mm-dd format. - `US`. For mm/dd/yyyy format.
filename	String	File name for the exported file. Value must end with the `.csv` file extension.
containerType	String	Type of asset that contains value mappings. Value must be `crosswalk`.
containerId	String	ID of the crosswalk. **Note:** You can find the ID of assets in Reference 360 or retrieve the IDs by using REST APIs. For more information, see "Asset IDs" on page 117.
pageSize	Number	Optional. Number of records to display on each page. The maximum value is `10000`. Default is `10000`.
page	Number	Optional. Page number to display. Default is `0`.
pit	String	Optional. Date to retrieve the point in time information about the crosswalk. Use the ISO format: yyyy-mm-dd. Default value is the current timestamp in yyyy-mm-dd format.
repeatHeaders	Boolean	Optional. Indicates whether to populate headers in the CSV response for the requested page. Value is `true` or `false`. Default is `true`.
useDisplayAttributes	Boolean	Indicates to use display attributes in the header of the CSV file. Value must be `true`.
codelistNameAndFieldDelimiter	String	Optional. Delimiter used to separate the codelist name and the display attribute name in the header. Value must be `COMMA`, `SEMICOLON`, `SPACE`, `TAB`, `PIPE`, `COLON`, `SECTION`, `CIRCUMFLEX`, or `ASTERISK`. Default is `SPACE`.

POST response

The response is a CSV file.

POST example

To export value mappings in a crosswalk with display attributes in the header, you might use the following request:

```
POST https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v3/export HTTP/1.1
Content-Type: application/json
Accept: application/octet-stream
Content-Length: 454

{
   "delimiter":"SEMICOLON",
   "codepage":"UTF8",
   "decimalSeparator":"COMMA",
   "thousandSeparator":"DOT",
   "dateFormat":"ISO",
   "filename":"testdata.csv",
   "containerType":"crosswalk",
   "containerId":"a8ef483f4f448a6ec273d547",
   "pageSize":10000,
   "page":0,
   "pit":"2021-11-09",
   "repeatHeaders":false,
   "useDisplayAttributes":true,
   "codelistNameAndFieldDelimiter":"SPACE"
}
```

The following sample response shows the exported data in the CSV format:

```
HTTP/1.1 200 OK
Content-Disposition: attachment;filename=testdata.csv
Content-Type: application/octet-stream
RDM-PAGE: 0
RDM-TOTAL-NUMBER-OF-ELEMENTS: 10
RDM-NUMBER-OF-ELEMENTS: 10
RDM-PAGE-SIZE: 10000
RDM-FIRST-PAGE: true
RDM-LAST-PAGE: true
Content-Length: 560

SourceCodelist Name;SourceCodelist Code;TargetCodelist Name;TargetCodelist Code
sourceName0;sourceCode0;targetName0;targetCode0
sourceName1;sourceCode1;targetName1;targetCode1
sourceName2;sourceCode2;targetName2;targetCode2
sourceName3;sourceCode3;targetName3;targetCode3
sourceName4;sourceCode4;targetName4;targetCode4
sourceName5;sourceCode5;targetName5;targetCode5
sourceName6;sourceCode6;targetName6;targetCode6
sourceName7;sourceCode7;targetName7;targetCode7
sourceName8;sourceCode8;targetName8;targetCode8
sourceName9;sourceCode9;targetName9;targetCode9
```

Export value mappings to CSV format with codelist names in the header

Export value mappings to CSV format with codelist names in the header.

POST request

To export value mappings in a crosswalk with codelist names in the header, submit a POST request with the following URI:

```
/rdm-service/external/v3/export
```

Note: In the request header, you must specify the `Accept` attribute to `application/octet-stream`.

Use the following parameters in the request body to specify the crosswalk value mappings to export:

Field	Type	Description
Delimiter	String	Optional. Delimiter used to separate values. Value must be `COMMA`, `SEMICOLON`, `SPACE`, `TAB`, `PIPE`, `COLON`, `SECTION`, `CIRCUMFLEX`, or `ASTERISK`. Default is `COMMA`.
codepage	String	Code page used for the export file. Value must be `UTF8` or `MS_WINDOWS`. Default is `UTF8`.
decimalSeparator	String	Optional. Decimal separator used for numbers. Value must be `COMMA` or `DOT`. Default is `DOT`.
thousandSeparator	String	Optional. Grouping separator used for numbers. Value must be `COMMA`, `DOT`, `SPACE`, `SINGLEQUOTE` or `NONE`. Default is `NONE`.
dateFormat	String	Format used for dates. Use one of the following formats: - `DE`. For dd.mm.yyyy format. - `ISO`. For yyyy-mm-dd format. - `US`. For mm/dd/yyyy format.
filename	String	File name for the exported file. Value must end with the `.csv` file extension.
containerType	String	Type of asset that contains value mappings. Value must be `crosswalk`.
containerId	String	ID of the crosswalk. **Note:** You can find the ID of assets in Reference 360 or retrieve the IDs by using REST APIs. For more information, see "Asset IDs" on page 117.
pageSize	Number	Optional. Number of records to display on each page. The maximum value is `10000`. Default is `10000`.
page	Number	Optional. Page number to display. Default is `0`.
pit	String	Optional. Date to retrieve the point in time information about the crosswalk. Use the ISO format: yyyy-mm-dd. Default value is the current timestamp in yyyy-mm-dd format.
repeatHeaders	Boolean	Optional. Indicates whether to populate headers in the CSV response for the requested page. Value is `true` or `false`. Default is `true`.
useAssetNameInHeader	Boolean	Indicates to use the AssetName attribute in the header. Value must be `true`.

POST response

The response is a CSV file.

POST example

To export value mappings in a crosswalk with codelist names in the header, you might use the following request:

```
POST https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v3/export HTTP/1.1
Content-Type: application/json
Accept: application/octet-stream
Content-Length: 454

{
   "delimiter":"SEMICOLON",
   "codepage":"UTF8",
```

```
   "decimalSeparator":"COMMA",
   "thousandSeparator":"DOT",
   "dateFormat":"ISO",
   "filename":"testdata.csv",
   "containerType":"crosswalk",
   "containerId":"a8ef483f4f448a6ec273d547",
   "pageSize":10000,
   "page":0,
   "pit":"2021-11-09",
   "repeatHeaders":false,
   "useAssetNameInHeader":true,
}
```

The following sample response shows the exported data in the CSV format:

```
HTTP/1.1 200 OK
Content-Disposition: attachment;filename=testdata.csv
Content-Type: application/octet-stream
RDM-PAGE: 0
RDM-TOTAL-NUMBER-OF-ELEMENTS: 10
RDM-NUMBER-OF-ELEMENTS: 10
RDM-PAGE-SIZE: 10000
RDM-FIRST-PAGE: true
RDM-LAST-PAGE: true
Content-Length: 560

SourceCodelist;TargetCodelist
sourceCode0;targetCode0
sourceCode1;targetCode1
sourceCode2;targetCode2
sourceCode3;targetCode3
sourceCode4;targetCode4
sourceCode5;targetCode5
sourceCode6;targetCode6
sourceCode7;targetCode7
sourceCode8;targetCode8
sourceCode9;targetCode9
```

Filter criteria for export version 3

You can export a filtered set of code values in a code list.

You can only filter values in the code attribute or values in attributes that are configured as display attributes. For example, you might want to filter values with `001` in the code attribute.

Supported filter operators and filter values for different field types

The filter operators depend on the field type of the attribute.

The following table describes the filter operators supported for each field type:

Field Type	Supported Filter Operators	Filter Values
Boolean	_eq (equal to) _ne (not equal to) _or (or) _and (and)	Boolean
Decimal or Integer	_eq (equal to) _ne (not equal to) _gt (greater than) _gte (greater than or equal to) _lt (less than) _lte (less than or equal to) _or (or) _and (and)	Number
String	_eq (equal to) _ne (not equal to) _or (or) _and (and)	Text
Date	_eq (equal to) _ne (not equal to) _or (or) _and (and)	ISO 8601 date or date and time For example, `2019-12-24` or `2019-12-15T14:17:04Z`.
Reference Data	_eq (equal to) _ne (not equal to) _in (in) _or (or) _and (and)	Values in the Code attribute or values in the display attributes for the reference data.

Filter examples

To filter assets with fields that are equal to NY, you can use the following filter operator:

```
{
   "state":{
      "_eq":"NY"
   }
}
```

To filter assets with fields that are not equal to NY, you can use the following filter operator:

```
{
   "state":{
      "_ne":"NY"
   }
}
```

To filter assets with number fields that are greater than 68, you can use the following filter operator:

```
{
   "age":{
      "_gt":68
   }
}
```

To filter assets with number fields that are greater than or equal to 68, you can use the following filter operator:

```
{
   "age":{
      "_gte":68
   }
}
```

To filter assets with number fields that are less than 68, you can use the following filter operator:

```
{
   "age":{
      "_lt":68
   }
}
```

To filter assets with number fields that are less than or equal to 68, you can use the following filter operator:

```
{
   "age":{
      "_lte":68
   }
}
```

To filter assets with fields that match any of the specified values, you can use the following filter operator:

```
{
   "state":{
      "_in":[
         "NY",
         "CA"
      ]
   }
}
```

To filter assets with fields that match all the specified values, you can use the following filter operator:

```
{
   "_and":[
      {
         "state":"CA"
      },
      {
         "age":68
      }
   ]
}
```

To filter assets with fields that match at least one of the specified values, you can use the following filter operator:

```
{
   "_or":[
      {
         "state":"CA"
      },
      {
         "age":68
      }
   ]
}
```

When you use a comma in the code, the comma acts like an `_and` operator. For example, the following code filters the state "NY" and people of age "68":

```
{
   {
      "state":"NY"
   },
   {
      "age":68
```

```
    }
}
```

rds

Use this resource to list all reference data sets, get the details for a reference data set, and list code lists in a reference data set.

List reference data sets

Retrieves a list of all reference data sets.

GET request

To retrieve a list of all reference data sets, submit a GET request with the following URI:

```
/rdm-service/external/v1/rds
```

GET response

The response contains information about each reference data sets.

The following table describes the attributes in the response:

Field	Type	Description
id	String	ID of the asset. Assets include reference data sets, code lists, crosswalks, and hierarchies. For more information, see "Asset IDs" on page 117.
name	String	Name of the asset.
description	String	Optional. Description of asset.
hierarchical	Boolean	Optional. Indicates whether code lists in the reference data set inherit the hierarchical structure definition.
levels	Number	Optional. Number of hierarchical levels supported in the code lists associated with the reference data set. If hierarchical is false or levels are not provided, value is `1`. If levels are unlimited, value is `-1`.
domain	String	Optional. Domain of the asset.
confidentiality	String	Optional. Confidentiality of the asset.
priority	String	Optional. Priority of the asset.
status	String	Optional. Status of the asset.
effectiveDate	String	Optional. Date the asset became effective.
approvedById	String	Optional. ID of the approver of the asset.

Field	Type	Description
approvedByName	String	Optional. Username of the user who approved the asset.
approvedOn	String	Optional. Date the asset was approved.
displayColumns	String	Optional. List of display columns used as labels for code values. Default is `name`.

GET example

To retrieve a list of reference data sets, you might use the following request:

```
GET https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/rds HTTP/1.1
Accept: application/json
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXXX
```

The following sample response shows the reference data sets:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 514

[
  {
    "id":"15d106a7905bd02a580d2b8d",
    "name":"Country",
    "description":"A Business Term named Country",
    "hierarchical":false,
    "levels":1,
    "displayColumns":[
      "Name"
    ]
  },
  {
    "id":"655df89e349a7fc1c0cd5f33",
    "name":"Currency",
    "hierarchical":false,
    "levels":1,
    "domain":"International standards",
    "confidentiality":"private",
    "priority":"Prio1",
    "status":"Draft",
    "effectiveDate":"2017-04-01",
    "approvedOn":"2017-03-01",
    "displayColumns":[
      "Name"
    ]
  }
]
```

List reference data set details

Retrieves the details a reference data set, such as the properties, status, structure definition, and attributes.

GET request

To retrieve the details of a reference data set, submit a GET request with the following URI:

```
/rdm-service/external/v1/rds/<reference data set ID>
```

Note: You can find the ID of assets in Reference 360 or retrieve the IDs by using REST APIs. For more information, see "Asset IDs" on page 117.

GET response

The response contains the summary information and definition of the reference data set.

The following table describes the attributes in the response:

Field	Type	Description
id	String	ID of the asset. Assets include reference data sets, code lists, crosswalks, and hierarchies. For more information, see "Asset IDs" on page 117.
name	String	Name of the asset.
description	String	Optional. Description of asset.
hierarchical	Boolean	Optional. Indicates whether code lists in the reference data set inherit the hierarchical structure definition.
levels	Number	Optional. Number of hierarchical levels supported in the code lists associated with the reference data set. If hierarchical is false or levels are not provided, value is `1`. If levels are unlimited, value is `-1`.
domain	String	Optional. Domain of the asset.
confidentiality	String	Optional. Confidentiality of the asset.
priority	String	Optional. Priority of the asset.
status	String	Optional. Status of the asset.
effectiveDate	String	Optional. Date the asset became effective.
approvedById	String	Optional. ID of the approver of the asset.
approvedByName	String	Optional. Username of the user who approved the asset.
approvedOn	String	Optional. Date the asset was approved.
defaultList	String	ID of the default code list.
displayColumns	String	Optional. List of display columns used as labels for code values. Default is `name`.
codeValueFields	-	Includes the attribute definition for code values in the reference data set.
name	String	Name of the field.
origin	String	Origin of the definition of the field.
datatype	String	Datatype of the field. Values are `String`, `Integer`, `Decimal`, `Boolean`, `Date`, or `Reference`.
mandatory	Boolean	Indicates whether the attribute is required.
relatedTermId	String	Optional. If the attribute datatype is Reference, lists the ID of the reference data set.

Field	Type	Description
displayColumns	Array	Optional. If the attribute datatype is Reference, lists the display columns.
dependencyDef	-	Optional. Includes the definition of the asset specified as the dependency.
termId	String	Optional. ID of the asset specified as the dependency.
displayColumns	Array	Optional. Display columns used as labels for code values associated with the dependent asset.

GET example

To retrieve the details of a reference data set, you might use the following request:

```
GET https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/rds/
15d106a7905bd02a580d2b8d HTTP/1.1
Accept: application/json
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXX
```

The following sample response shows the details of a reference data set:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 1265

{
  "id":"15d106a7905bd02a580d2b8d",
  "name":"Country",
  "description":"A Business Term named Country",
  "hierarchical":false,
  "levels":1,
  "defaultList":"fa2a7f11fea1fc38db8d29aa",
  "domain":"International standards",
  "confidentiality":"private",
  "priority":"Prio1",
  "status":"Draft",
  "effectiveDate":"2017-04-01",
  "approvedOn":"2017-03-01",
  "displayColumns":[
    "Name"
  ],
  "codeValueFields":[
    {
      "name":"Name",
      "origin":"TERM",
      "datatype":"String",
      "mandatory":true
    },
    {
      "name":"Code",
      "origin":"TERM",
      "datatype":"String",
      "mandatory":true
    },
    {
      "name":"Description",
      "origin":"TERM",
      "datatype":"String",
      "mandatory":false
    },
    {
      "name":"Alpha2Code",
      "origin":"TERM",
      "datatype":"String",
      "mandatory":true
    },
```

```
      {
        "name":"Alpha3Code",
        "origin":"TERM",
        "datatype":"String",
        "mandatory":false
      },
      {
        "name":"RefField",
        "origin":"TERM",
        "datatype":"Reference",
        "mandatory":true,
        "relatedTermId":"655df89e349a7fc1c0cd5f33",
        "displayColumns":[
          "column1",
          "column2"
        ]
      }
    ],
    "dependencyDef":{
      "termId":"Continent",
      "displayColumns":[
        "Name"
      ]
    }
}
```

List code lists

Retrieves a list of all code lists in a reference data set.

GET request

To retrieve a list of all code lists in a reference data set, submit a GET request with the following URI:

```
/rdm-service/external/v1/rds/<reference data set ID>/codelists
```

Note: You can find the ID of assets in Reference 360 or retrieve the IDs by using REST APIs. For more information, see "Asset IDs" on page 117.

GET response

The response contains information about the code lists in the specified reference data set.

The following table describes the attributes in the response:

Field	Type	Definition
id	String	ID of the asset. Assets include reference data sets, code lists, crosswalks, and hierarchies. For more information, see "Asset IDs" on page 117.
termId	String	ID of the reference data set to which the code list is associated.
name	String	Name of the asset.
description	String	Optional. Description of asset.
displayColumns	String	Optional. List of display columns used as labels for code values. Default is `name`.
version	String	Optional. Version of the code list.
application	String	Optional. Application that uses the code list.

Field	Type	Definition
hierarchical	Boolean	Optional. Indicates whether code lists in the reference data set inherit the hierarchical structure definition.
levels	Number	Optional. Number of hierarchical levels supported in the code lists associated with the reference data set. If hierarchical is false or levels are not provided, value is `1`. If levels are unlimited, value is `-1`.
domain	String	Optional. Domain of the asset.
confidentiality	String	Optional. Confidentiality of the asset.
priority	String	Optional. Priority of the asset.
status	String	Optional. Status of the asset.
effectiveDate	String	Optional. Date the asset became effective.
approvedById	String	Optional. ID of the approver of the asset.
approvedByName	String	Optional. Username of the user who approved the asset.
approvedOn	String	Optional. Date the asset was approved.

GET example

To retrieve a list of all code lists in a reference data set, you might use the following request:

```
GET https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/rds/
15d106a7905bd02a580d2b8d/codelists HTTP/1.1
Accept: application/json
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXXX
```

The following sample response shows the code lists in a reference data set:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 747

[
  {
    "id":"863efc094436e68bc4299204",
    "termId":"c0af1cd77e7a1a3d658883ad",
    "name":"Units",
    "description":"Code list for units",
    "version":"2.0",
    "application":"UN recommendation 20",
    "hierarchical":false,
    "levels":1,
    "displayColumns":[
      "Name"
    ]
  },
  {
    "id":"17f60eb76ebcc75fec93dadc",
    "termId":"c0af1cd77e7a1a3d658883ad",
    "name":"SAP Units",
    "description":"Code list for SAP units",
    "version":"1.1",
    "application":"SAP",
    "hierarchical":true,
```

```
    "levels":10,
    "domain":"International standards",
    "confidentiality":"private",
    "priority":"Prio1",
    "status":"Draft",
    "effectiveDate":"2017-04-01",
    "approvedOn":"2017-03-01",
    "displayColumns":[
      "Name",
      "Code"
    ]
  }
]
```

codelists

Use this resource to retrieve the details of a code list, the details of a code value, and the crosswalks associated to a code list. You can also unlock code lists locked by other users.

Unlock code lists

Unlocks a code list that is locked by another user.

Note: To use the API, you must be assigned the Informatica Intelligent Cloud Services Reference 360 Administrator role.

PUT request

To unlock a code list, submit a PUT request with the following URI:

```
/rdm-service/external/v1/codelists/<code list ID>/unlock
```

Note: You can find the ID of assets in Reference 360 or retrieve the IDs by using REST APIs. For more information, see "Asset IDs" on page 117.

PUT response

The response contains a success message.

PUT example

To unlock a code list, you might use the following request:

```
PUT https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/codelists/
4fb1356728272974bd46945f/unlock HTTP/1.1
Accept: application/json
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXXX
```

The following sample response shows the success message:

```
HTTP/1.1 200 OK
```

List code list details

Retrieves the details of a code list, such as the properties, status, structure definition, and attributes.

GET request

To retrieve the details of a code list, submit a GET request with the following URI:

```
/rdm-service/external/v1/codelists/<code list ID>
```

Note: You can find the ID of assets in Reference 360 or retrieve the IDs by using REST APIs. For more information, see "Asset IDs" on page 117.

GET response

The response contains the details of the specified code list.

The following table describes the attributes in the response:

Field	Type	Description
id	String	ID of the asset. Assets include reference data sets, code lists, crosswalks, and hierarchies. For more information, see "Asset IDs" on page 117.
termId	String	ID of the reference data set to which the code list is associated.
name	String	Name of the asset.
description	String	Optional. Description of asset.
hierarchical	Boolean	Optional. Indicates whether code lists in the reference data set inherit the hierarchical structure definition.
levels	Number	Optional. Number of hierarchical levels supported in the code lists associated with the reference data set. If hierarchical is false or levels are not provided, value is `1`. If levels are unlimited, value is `-1`.
domain	String	Optional. Domain of the asset.
confidentiality	String	Optional. Confidentiality of the asset.
priority	String	Optional. Priority of the asset.
status	String	Optional. Status of the asset.
effectiveDate	String	Optional. Date the asset became effective.
approvedById	String	Optional. ID of the approver of the asset.
approvedByName	String	Optional. Username of the user who approved the asset.
approvedOn	String	Optional. Date the asset was approved.
defaultList	String	ID of the default code list.
displayColumns	String	Optional. List of display columns used as labels for code values. Default is `name`.

Field	Type	Description
codeValueFields	-	Includes the attribute definition for code values in the reference data set.
name	String	Name of the field.
origin	String	Origin of the definition of the field.
datatype	String	Datatype of the field. Values are `String`, `Integer`, `Decimal`, `Boolean`, `Date`, or `Reference`.
mandatory	Boolean	Indicates whether the attribute is required.
relatedTermId	String	Optional. If the attribute datatype is Reference, lists the ID of the reference data set.
displayColumns	Array	Optional. If the attribute datatype is Reference, lists the display columns.
dependencyDef	-	Optional. Includes the definition of the asset specified as the dependency.
termId	String	Optional. ID of the asset specified as the dependency.
displayColumns	Array	Optional. Display columns used as labels for code values associated with the dependent asset.

GET example

To retrieve the details of a code list, you might use the following request:

```
GET https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/codelists/
4fb1356728272974bd46945f HTTP/1.1
Accept: application/json
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXXX
```

The following sample response shows the details of a code list:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 1284

{
  "id":"4fb1356728272974bd46945f",
  "termId":"Unit Term",
  "name":"Units",
  "description":"Code list for units",
  "version":"2.0",
  "application":"UN recommendation 20",
  "hierarchical":false,
  "levels":1,
  "domain":"International standards",
  "confidentiality":"private",
  "priority":"Prio1",
  "status":"Draft",
  "effectiveDate":"2017-04-01",
  "approvedOn":"2017-03-01",
  "displayColumns":[
    "Name",
    "Code"
  ],
  "codeValueFields":[
    {
      "name":"Name",
      "origin":"TERM",
      "datatype":"String",
      "mandatory":false
```

```
      },
      {
        "name":"Code",
        "origin":"TERM",
        "datatype":"String",
        "mandatory":true
      },
      {
        "name":"Description",
        "origin":"TERM",
        "datatype":"String",
        "mandatory":false
      },
      {
        "name":"achronym",
        "origin":"CODELIST",
        "datatype":"String",
        "mandatory":false
      },
      {
        "name":"refField2",
        "origin":"CODELIST",
        "datatype":"Reference",
        "mandatory":true,
        "relatedTermId":"b02c86d02ac7de3a688353dc",
        "relatedListId":"dc245266d5a61ce4d0535f74",
        "displayColumns":[
          "column5"
        ]
      }
    ],
    "dependencyDef":{
      "termId":"UnitSystem Term",
      "codelistId":"UnitSystems",
      "displayColumns":[
        "Name"
      ]
    }
  }
```

List code value details

Retrieves the details of a code value.

You identify the code value that you want to retrieve the details for by specifying the value in the Code attribute.

Note: You cannot use the + symbol inside the code field value.

GET request

To retrieve the details of a code value, submit a GET request with the following URI and specify the code:

```
/rdm-service/external/v1/codelists/{code list ID}/codevalues?Code={code}
```

GET response

The response contains the details of the code value.

The following table describes the attributes in the response:

Field	Type	Description
codelistId	String	ID of the code list that the code values belong to.
status	String	Optional. Status of the code value. **Note:** To retrieve a list of statuses, use the List enum entries API. For more information, see "List system reference data values" on page 234.
effectiveDate	String	Optional. Date the code value became effective.
codeValueFields	Object	Includes the attribute field values for the code value.

GET example

To retrieve the details of a code value, you might use the following request:

```
GET https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/codelists/
591c302d8af18b0001b1fac2/codevalues?Code=AR HTTP/1.1
Accept: application/json
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXXX
```

The following sample response shows the details of a code value:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 237

[
  {
    "codelistId":"591c302d8af18b0001b1fac2",
    "status":"Draft",
    "effectiveDate":"2019-09-20",
    "codeValueFields":{
      "Name":"Argentina",
      "Code":"AR",
      "description":"The EU country code for Argentina"
    }
  }
]
```

Move a code value

Moves a code value to another node within the same hierarchical code list without locking the code list.

POST request

To move a code value to another node within the same hierarchical code list, submit a POST request with the following URI:

```
/rdm-service/external/v1/codelists/{listIdentifier}/codevalues/move
```

Use the following parameters in the request body to specify the code fields of the code value to move and the target node:

Field	Type	Description
Code	String	The code field of the code value to move.
TargetParentCode	String	The code field of the target node to which you want to move the code value. If you don't specify a code, the code value is moved to the top level.

POST response

The response contains a success message.

POST example

To move the code value to another node within the same hierarchical code list, you might use the following request:

```
POST https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/codelists/
833861786fb48aa467602980/codevalues/move HTTP/1.1
Content-Type: application/json
Content-Length: 48
Host: localhost:8080

{
  "Code" : "DE",
  "TargetParentCode" : "EU"
}
```

The following sample response shows the success message:

```
HTTP/1.1 200 OK
```

Delete code values

Deletes one or multiple code values that you no longer need. When you delete code values using the Delete code values API, you directly delete code values without creating a draft changes or sending your changes for approval.

You cannot delete the following types of code values:

- Code values that are defined as a parent code value in a hierarchical code list
- Code values that are used as a Reference Data attribute for another code list
- Code values that are used as a dependency in another code list
- Code values that are part of a value mapping in a crosswalk
- Code values that are part of a hierarchy asset

DELETE request

To delete a code value, submit a DELETE request with the following URI:

```
/rdm-service/external/v1/codelists/<code list ID>/codevalues
```

The following table describes the attributes in the request:

Field	Type	Description
Code	String	Comma-separated list of code values by the value in the Code attribute.

DELETE response

The response contains the deletion report. If the delete failed, the report provides failure reasons.

The following table describes the attributes in the response:

Field	Type	Description
numberOfRecordsDeleted	Number	Number of code values that were deleted successfully.
numberOfRecordsFailed	Number	Number of code values that were not deleted.
deletedRecords	-	Lists the code values that were deleted successfully.
Code	String	Code attribute value for deleted code value.
Label	String	Display attribute value for deleted code value.
failedRecords	-	Lists the code values that were not deleted and describes the reasons.
Code	String	Code attribute value for the code value.
label	String	Display attribute value for the code value.
errorCode	String	Error code for the error type.
errorSummary	String	Explains why the code value was not deleted.

The display attribute value represents the code value when the code value appears in other assets in Reference 360. For more information, see "Display attributes" on page 26.

DELETE example

To delete code values, you might use the following request:

```
DELETE https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/codelists/
34cea9471fe977f7decef5f5/codevalues HTTP/1.1
Content-Type: application/json
Content-Length: 30
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXXX


{
  "Codes":[
    "DE",
    "EN"
  ]
}
```

The following sample response shows the deletion report:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 345

{
  "deletedRecords":[
    {
      "Code":"DE",
      "label":"DELabel"
    }
  ],
  "failedRecords":[
    {
```

```
        "Code":"EN",
        "label":"EnLabel",
        "errorCauses":[
          {
            "errorCode":"RDM.0010209",
            "errorSummary":"The code value is used in a Hierarchy"
          }
        ]
      }
    ],
    "numberOfRecordsDeleted":1,
    "numberOfRecordsFailed":1
}
```

List crosswalks for a code list

Retrieves the crosswalks associated to a code list and the summary information for each crosswalk.

GET request

To retrieve all crosswalks associated to a code list, submit a GET request with the following URI:

```
/rdm-service/external/v1/codelists/<code list ID>/crosswalks
```

Note: You can find the ID of assets in Reference 360 or retrieve the IDs by using REST APIs. For more information, see "Asset IDs" on page 117.

GET response

The response contains the crosswalks associated to the code list.

The following table describes the attributes in the response:

Field	Type	Description
id	String	ID of the asset. Assets include reference data sets, code lists, crosswalks, and hierarchies. For more information, see "Asset IDs" on page 117.
sourceCodelistId	String	ID of the source code list to which the crosswalk is associated.
targetCodelistId	String	ID of the target code list.
description	String	Optional. Description of asset.
status	String	Optional. Status of the asset.
confidentiality	String	Optional. Confidentiality of the asset.
sourceApplication	String	Optional. Application of the source code list.
targetApplication	String	Optional. Application of the target code list.
effectiveDate	String	Optional. Date the asset became effective.
approvedById	String	Optional. ID of the approver of the asset.
approvedByName	String	Optional. Username of the user who approved the asset.
approvedOn	String	Optional. Date the asset was approved.

GET example

To retrieve the crosswalks associated to a code list, you might use the following request:

```
GET https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/codelists/
eff26036111cbb1e9c03f21f/crosswalks HTTP/1.1
Accept: application/json
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXX
```

The following sample response shows the crosswalks associated to a code list:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 514

[
  {
    "id":"242a0d14fc1149843cf00626",
    "description":"Sample Crosswalk 1",
    "status":"active",
    "sourceCodelistId":"eff26036111cbb1e9c03f21f",
    "targetCodelistId":"6253315d698f95a216e15a5e"
  },
  {
    "id":"cd11cae0400cf0eaaedfd711",
    "description":"Sample Crosswalk 2",
    "status":"inactive",
    "sourceCodelistId":"eff26036111cbb1e9c03f21f",
    "targetCodelistId":"1bf325b44008a5ba4034c23c",
    "confidentiality":"private",
    "effectiveDate":"2007-04-01",
    "approvedOn":"2017-03-01"
  }
]
```

List modified code values by time range

Retrieves the modified code values in a code list for a time range.

By default, the request returns the first 100 records. To retrieve more records or to view the next page of records, use the query parameters. You can retrieve a maximum of 10000 records in a request.

GET request

To retrieve the modified code values in a code list for a time range, submit a GET request with the following URI:

```
/rdm-service/external/v1/codelists/<code list ID>/codevalues/modifications?
from=<from>&to=<to>
```

To specify the page number and page size, submit a GET request with the following query parameters appended to the URI:

```
/rdm-service/external/v1/codelists/<code list ID>/codevalues/modifications?
from=<from>&to=<to>&pageSize=<page size>&page=<page number>
```

GET request query parameters

You can append query parameters to the URI to specify the time range, page number, and page size.

The following table lists the query parameters:

Parameter	Description
from	Start date and time of the time range. The start point is inclusive. Time range must be in the ISO-8601 format: yyyy-MM-dd'T'HH:mm:ss'Z'. For example, you might use `2019-12-12T14:04:04Z`.
to	End date and time of the time range. The end point is exclusive. Time range must be in the ISO-8601 format: yyyy-MM-dd'T'HH:mm:ss'Z'. For example, you might use `2019-12-15T14:04:04Z`.
page	Optional. Page number to display. Default is `0`.
pageSize	Optional. Number of records to display per page. Default value is `100`. Maximum value is `10000`.

GET response

The response contains data about the modified code values.

The following table describes the attributes in the response:

Field	Type	Description
page	Number	Page number displayed.
pageSize	Number	Number of records displayed per page.
totalNumberOfElements	Number	Total number of records found.
numberOfElements	Number	Number of records returned in the current page.
lastPage	Boolean	Indicates whether the current page is the last page in the total result.
firstPage	Boolean	Indicates whether the current page is the first page in the total result.
content	Array	Includes the list of code values.
status	String	Included in the content object. Optional. Status of the code value.
effectiveDate	String	Included in the content object. Optional. Date code value becomes effective.
codeValueFields	Object	Included in the content object. Contains the field values for the code value.
lastUpdateDate	String	Included in the content object. Update date of the last modification.
changeType	Object	Included in the content object. Type of change made to the code value. Values are `MODIFIED` or `DELETED`. **Note:** The MODIFIED change type appears for both new and updated code values.

GET example

To retrieve the first page of modified code values for a time range, you might use the following request:

```
GET https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/codelists/
ce63c85efb9791eb49c4baa3/codevalues/modifications?
from=2019-12-11T13:29:55Z&to=2019-12-12T13:29:55Z&page=0&pageSize=100 HTTP/1.1
Accept: application/json
```

The following sample response shows the first page of modified code values:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 412

{
  "pageSize":100,
  "page":0,
  "totalNumberOfElements":1000,
  "numberOfElements":1,
  "lastPage":false,
  "firstPage":true,
  "content":[
    {
      "status":"Draft",
      "dependency":"f9b48b61fbeb3b491a69bd44",
      "lastUpdateDate":"2019-12-11T13:29:55Z",
      "changeType":"MODIFIED",
      "effectiveDate":"2017-04-01",
      "codeValueFields":{
        "field1":"Some value"
      }
    }
  ]
}
```

Known limitations

- If the code list is in draft state, some code values might appear in the API response even though the code values have not changed in the specified time range. For example, if you delete a code value in a draft code list, the last update date of the code value updates to the date the code value was deleted in the draft. This means that the deleted code value in the draft code list might meet the specified time range criteria now.

List modified code value relationships in hierarchy by time range

Retrieve the modified code value relationships in a hierarchy for a time range. Use the API to retrieve changes to code value relationships, such as code values that moved in the hierarchy and code values added under a parent code value.

By default, the request returns the first 100 records. To retrieve more records or to view the next page of records, use the query parameters.

GET request

To retrieve the modified code values in a hierarchical code list for a time range, submit a GET request with the following URI:

```
/rdm-service/external/v1/codelists/<code list ID>/codevalues/hierarchy/modifications?
from=<from>&to=<to>
```

To specify the page number and page size, submit a GET request with the following query parameters appended to the URI:

```
/rdm-service/external/v1/codelists/<code list ID>/codevalues/hierarchy/modifications?
from=<from>&to=<to>&pageSize=<page size>&page=<page number>
```

GET request query parameters

You can append query parameters to the URI to specify the time range, page number, and page size.

The following table lists the query parameters:

Parameter	Description
from	Start date and time of the time range. The start point is inclusive. Time range must be in the ISO-8601 format: yyyy-MM-dd'T'HH:mm:ss'Z'. For example, you might use `2019-12-12T14:04:04Z`.
to	End date and time of the time range. The end point is exclusive. Time range must be in the ISO-8601 format: yyyy-MM-dd'T'HH:mm:ss'Z'. For example, you might use `2019-12-15T14:04:04Z`.
page	Optional. Page number to display. Default is `0`.
pageSize	Optional. Number of records to display per page. Default value is `100`. Maximum value is `10000`.

GET response

The response contains data about the modified code value relationships for the specified time range.

The following table describes the attributes in the response:

Field	Type	Description
page	Number	Page number displayed.
pageSize	Number	Number of records displayed per page.
totalNumberOfElements	Number	Total number of records found.
numberOfElements	Number	Number of records returned in the current page.
lastPage	Boolean	Indicates whether the current page is the last page in the total result.
firstPage	Boolean	Indicates whether the current page is the first page in the total result.
content	Array	Includes the list of code values.
lastUpdateDate	String	Included in the content object. Update date of the last modification.
parentCode	String	Included in the content object. Code value of the parent node.
childCode	String	Included in the content object. Code value of the child node.

GET example

To retrieve the first page of modified code values relationships for a time range, you might use the following request:

```
GET https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/codelists/123456/
codevalues/hierarchy/modifications?
from=2019-12-11T13:29:55Z&to=2019-12-12T13:29:55Z&page=0&pageSize=100 HTTP/1.1
Accept: application/json
Host: localhost:8080
```

The following sample response shows the first page of modified code value relationships:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 261

{
  "pageSize":100,
  "page":0,
  "totalNumberOfElements":1000,
  "numberOfElements":1,
  "lastPage":false,
  "firstPage":true,
  "content":[
    {
      "lastUpdateDate":"2019-12-11T13:29:55Z",
      "parentCode":"USA",
      "childCode":"NY"
    }
  ]
}
```

Known limitations

- If the code list is in draft state, some code value relationships might appear in the API response even though the code value relationships have not changed in the specified time range. For example, if you move a code value in a draft code list, the last update date of the code value updates to the date the code value was moved in the draft. This means that the moved code value in draft code list might meet the specified time range criteria now.

List modified code lists by time range

Retrieve the modified code lists for a time range. Modified code lists include code lists that users published or code values that users directly imported during the time range.

By default, the request returns the first 100 records. To retrieve more records or to view the next page of records, use the query parameters. You can retrieve a maximum of 10000 records in a request.

GET request

To retrieve the modified code lists for a time range, submit a GET request with the following URI:

```
/rdm-service/external/v1/codelists/modifications?from=<from>&to=<to>
```

To specify the page number and page size, submit a GET request with the following query parameters appended to the URI:

```
/rdm-service/external/v1/codelists/modifications?from=<from>&to=<to>&pageSize=<page
size>&page=<page number>
```

GET request query parameters

You can append query parameters to the URI to specify the time range, page number, and page size.

Parameter	Description
from	Start date and time of the time range. The start point is inclusive. Time range must be in the ISO-8601 format: yyyy-MM-dd'T'HH:mm:ss'Z'. For example, you might use `2019-12-12T14:04:04Z`.
to	End date and time of the time range. The end point is exclusive. Time range must be in the ISO-8601 format: yyyy-MM-dd'T'HH:mm:ss'Z'. For example, you might use `2019-12-15T14:04:04Z`.
page	Optional. Page number to display. Default is `0`.
pageSize	Optional. Number of records to display per page. Default value is `100`. Maximum value is `10000`.

GET response

The response contains data about the modified code lists.

The following table describes the attributes in the response:

Field	Type	Description
page	Number	Page number displayed.
pageSize	Number	Number of records displayed per page.
totalNumberOfElements	Number	Total number of records found.
numberOfElements	Number	Number of records returned in the current page.
lastPage	Boolean	Indicates whether the current page is the last page in the total result.
firstPage	Boolean	Indicates whether the current page is the first page in the total result.
content	Array	Includes the list of code values.
id	String	Included in the content object. ID of the code list.
updateDate	String	Included in the content object. Last updated date of the change.
changeType	String	Included in the content object. Type of change made to the code list. Values are `DIRECT_IMPORT` and `PUBLISH_DRAFT`.

GET example

To retrieve the first page of modified code lists for a time range, you might use the following request:

```
GET https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/codelists/
modifications?from=2019-12-11T13:29:55Z&to=2019-12-12T13:29:55Z&page=0&pageSize=100
HTTP/1.1
Accept: application/json
```

The following sample response shows the first page of modified code lists:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 282

{
  "pageSize":100,
  "page":0,
  "totalNumberOfElements":1000,
  "numberOfElements":1,
  "lastPage":true,
  "firstPage":true,
  "content":[
    {
      "id":"3f23f14a44bae0f6e2bb8aa2",
      "updateDate":"2019-12-11T13:29:55Z",
      "changeType":"DIRECT_IMPORT"
    }
  ]
}
```

crosswalks

Use this resource to retrieve the details of a crosswalk, such as the properties, status, source code list, and target code list. You can also retrieve the value mappings for a code value.

List crosswalk details

Retrieves the details of a crosswalk.

GET request

To retrieve the details of a crosswalk, submit a GET request with the following URI:

```
/rdm-service/external/v1/crosswalks/<crosswalk ID>
```

Note: You can find the ID of assets in Reference 360 or retrieve the IDs by using REST APIs. For more information, see "Asset IDs" on page 117.

GET response

The response contains the details of a crosswalk.

The following table describes the attributes in the response:

Field	Type	Description
id	String	ID of the asset. Assets include reference data sets, code lists, crosswalks, and hierarchies. For more information, see "Asset IDs" on page 117.
sourceCodelistId	String	ID of the source code list to which the crosswalk is associated.
targetCodelistId	String	ID of the target code list.
description	String	Optional. Description of asset.

Field	Type	Description
status	String	Optional. Status of the asset.
confidentiality	String	Optional. Confidentiality of the asset.
sourceApplication	String	Optional. Application of the source code list.
targetApplication	String	Optional. Application of the target code list.
effectiveDate	String	Optional. Date the asset became effective.
approvedById	String	Optional. ID of the approver of the asset.
approvedByName	String	Optional. Username of the user who approved the asset.
approvedOn	String	Optional. Date the asset was approved.

GET example

To retrieve the details of a crosswalk, you might use the following request:

```
GET https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/crosswalks/
20c85dde693051cf8037f1eb HTTP/1.1
Accept: application/json
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXXX
```

The following sample response shows the details of a crosswalk:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 300

{
  "id" : "20c85dde693051cf8037f1eb",
  "description" : "Sample Crosswalk",
  "status" : "active",
  "sourceCodelistId" : "f6821570be0fd451934dff86",
  "targetCodelistId" : "628234c5ba9d033c33ff0284",
  "confidentiality" : "private",
  "effectiveDate" : "2007-04-01",
  "approvedOn" : "2017-03-01"
}
```

List value mappings for a code value

Retrieve the value mappings for a code value.

You identify the code value that you want to retrieve value mappings for by specifying the value in the Code attribute.

Note: You cannot use the + symbol inside the code field value.

GET request

To retrieve the value mappings for a code value, submit a GET request with the following URI and specify the code:

```
/rdm-service/external/v1/crosswalks/{crosswalk ID}/mappings?Code={code}
```

GET response

The response contains the value mappings for the code value.

The following table describes the attributes in the response:

Field	Type	Description
Code	String	Value in the Code attribute of the source code value.
mappings	Array	Includes the values in the Code attribute of the target code values.

GET example

To retrieve the value mappings for the code value, you might use the following request:

```
GET https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/crosswalks/
69d08b8c300e9d1aed32a777/mappings?Code=DE HTTP/1.1
Accept: application/json
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXX
```

The following sample response shows value mappings:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 92

[
  {
    "Code":"DE",
    "mappings":[
      "Code 1",
      "Code 2",
      "Code 3",
      "Code 4",
      "Code 5"
    ]
  }
]
```

Delete value mappings of a crosswalk

To delete value mappings of a crosswalk, you must retrieve the list of all mappings of the crosswalk by crosswalk identifier and then delete value mappings of the crosswalk by mapped source and target code values.

List value mappings of a crosswalk by crosswalk identifier

Retrieves all value mappings of a crosswalk by crosswalk identifier.

GET request

To retrieve all mappings of a crosswalk by crosswalk identifier, submit a GET request with the following URI:

```
/rdm-service/external/v2/crosswalks/{crosswalkIdentifier}/mappings?
page={page}&pageSize={pageSize}
```

Note: This API also supports crosswalks created from code lists belonging to different reference data sets.

GET request query parameters

You can append query parameters to the URI to specify the page number and page size.

The following table lists the query parameters:

Field	Type	Description
page	Number	Page number to display. Default is 0.
pageSize	Number	Number of records to display on each page. Default is 100.

GET response

The response contains the mappings of a specific crosswalk by crosswalk identifier.

The following table describes the attributes in the response:

Field	Type	Description
mappings	Array	List of mappings for the specified crosswalk.
Code	String	Source code value of a crosswalk mapping.
targetCode	String	Target code value of a crosswalk mapping.
page	Number	Page number to display. Default is 0.
pageSize	Number	Number of records to display on each page. Default is 100.
firstPage	Boolean	Indicates whether the current page is the first page of the total results.
lastPage	Boolean	Indicates whether the current page is the last page of the total results.
numberOfElements	Number	Number of records returned in the current page.
totalNumberOfElements	Number	Total number of records found.

GET example

To retrieve the mappings of a specific crosswalk by crosswalk identifier, you might use the following request:

```
GET https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v2/crosswalks/
daf0cc189e530b6979ac77ce/mappings?page=0&pageSize=100 HTTP/1.1
Accept: application/json
```

The following sample response shows the mappings of a specific crosswalk by crosswalk identifier:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 465

{
   "pageSize":100,
   "page":0,
   "totalNumberOfElements":5,
   "numberOfElements":5,
   "lastPage":true,
   "firstPage":true,
   "mappings":[
      {
         "Code":"code1",
         "targetCode":"targetCode1"
      },
      {
         "Code":"code2",
```

```
            "targetCode":"targetCode2"
         },
         {
            "Code":"code3",
            "targetCode":"targetCode3"
         },
         {
            "Code":"code4",
            "targetCode":"targetCode4"
         },
         {
            "Code":"code5",
            "targetCode":"targetCode5"
         }
      ]
}
```

Delete value mappings of a crosswalk by mapped source and target code values

Delete value mappings of a crosswalk by mapped source and target code values.

POST request

To delete value mappings of a crosswalk by mapped source and target code values, submit a POST request with the following URI:

```
/rdm-service/external/v2/crosswalks/{crosswalkIdentifier}/mappings
```

Note:

- You can delete a maximum of 100 value mappings with each request.
- This API also supports crosswalks created from code lists belonging to different reference data sets.

Use the following parameters in the request body to specify the value mappings of a crosswalk to delete:

Field	Type	Description
action	String	Action to perform.
mappings	Array	List of mappings for the specified crosswalk.
Code	String	Source code value of a crosswalk mapping.
targetCode	String	Target code value of a crosswalk mapping.

POST response

A 204 no content response is returned.

POST example

To delete value mappings of a crosswalk by mapped source and target code values, you might use the following request:

```
POST https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v2/crosswalks/
daf0cc189e530b6979ac77ce/mappings HTTP/1.1
Content-Type: application/json
Content-Length: 384

{
   "action":"DELETE",
   "mappings":[
      {
```

```
            "Code":"sourceCode1",
            "targetCode":"targetCode1"
        },
        {
            "Code":"sourceCode2",
            "targetCode":"targetCode2"
        },
        {
            "Code":"sourceCode3",
            "targetCode":"targetCode3"
        },
        {
            "Code":"sourceCode4",
            "targetCode":"targetCode4"
        },
        {
            "Code":"sourceCode5",
            "targetCode":"targetCode5"
        }
    ]
}
```

The following sample response shows the no content response:

```
HTTP/1.1 204 No Content
```

Delete duplicate mappings of a crosswalk

The delete request runs a crosswalk cleanser job that scans all mappings of a crosswalk and removes the duplicate mappings.

DELETE request

To delete duplicate mappings of a crosswalk, submit a DELETE request with the following URI:

```
/rdm-service/external/v1/crosswalks/{crosswalkIdentifier}/mappings/duplicates
```

Note: You must execute this API to run the crosswalk cleanser job only if you find duplicate crosswalk mappings in the exported CSV file. The CSV file must be exported using export v3 API at a point in time.

DELETE response

A 202 accepted response is returned.

DELETE example

To delete duplicate mappings of a crosswalk, you might use the following request:

```
DELETE https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/crosswalks/
12102ba6924d0279d40bf5b1/mappings/duplicates HTTP/1.1
Content-Type: application/json
```

The following sample response shows the accepted response:

```
HTTP/1.1 202 Accepted
```

Get job details of a crosswalk cleanser job

Retrieves job details of a crosswalk cleanser job.

GET request

To get the job details of a crosswalk cleanser job, submit a GET request with the following URI:

```
/rdm-service/external/v1/crosswalks/{crosswalkIdentifier}/mappings/duplicates/status
```

GET response

The response contains the details of the crosswalk cleanser job, such as the status of the crosswalk cleanser job, number of records processed for import, and the error details.

The following table describes the attributes in the response:

Field	Type	Description
crosswalkId	String	Identifier of the crosswalk for which the crosswalk cleanser job was triggered.
createdBy	String	User name of the user who triggered the crosswalk cleanser job.
createdDate	String	Date when the crosswalk cleanser job was triggered.
status	String	Status of the crosswalk cleanser job.
jobDetails	Object	Details of the crosswalk cleanser job.
initialNumberOfMappings	Number	Number of existing crosswalk mappings.
invalidSourcePKeyReport	Object	Details of invalid mappings.
numberOfRecords	Number	Number of invalid records.
numberOfSuccessRecords	Number	Number of invalid records that were successfully deleted.
numberOfFailedRecords	Number	Number of invalid records that were not deleted.
errors	Array	Details of the errors.
errorCode	String	Error code for the error type.
errorSummary	String	Message that explains why the invalid records are not deleted.
errorParameter	Object	Error parameter.

GET example

To get the job details of a crosswalk cleanser job, you might use the following request:

```
GET https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/crosswalks/
12102ba6924d0279d40bf5b1/mappings/duplicates/status HTTP/1.1
Accept: application/json
```

The following sample response shows the job details of a crosswalk cleanser job:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 555

{
   "crosswalkId":"12102ba6924d0279d40bf5b1",
   "createdBy":"12bb7c6b10e23be958e8b270",
   "createdDate":"2021-12-10T11:29:34.174+00:00",
   "status":"RUNNING",
   "jobDetails":{
      "initialNumberOfMappings":10,
      "invalidSourcePKeyReport":{
         "numberOfRecords":5,
```

```
            "numberOfSuccessRecords":4,
            "numberOfFailedRecords":1
         }
      },
      "errors":[
         {
            "errorCode":"RDM.0010000",
            "errorSummary":"Unable to fix the sourcePkey",
            "errorParameter":{
               "sourcePkey":"619cd13b57eb9948fa585d65"
            }
         }
      ]
   }
```


Known limitation

If the crosswalk cleanser job fails to remove duplicate mappings of a crosswalk, the API response returns the `RDM.0010269` error code.

To avoid duplicate mappings in the crosswalk when the crosswalk cleanser job fails, perform the following actions:

1. In Reference 360, open the specific crosswalk.
2. Click **Export mappings** to export the value mappings of the crosswalk.
3. Delete the crosswalk.
4. Create another crosswalk between the same source and target code lists.
5. Open the crosswalk and click **Import Mappings** to import the exported CSV file to the crosswalk.

hierarchies

Use this resource to retrieve hierarchies, hierarchy details, and hierarchy model relationships.

List hierarchies

Retrieves a list of all hierarchies.

GET request

To retrieve a list of all hierarchies, submit a GET request with the following URI:

```
/rdm-service/external/v1/hierarchies
```

GET response

The response contains information about each hierarchy.

The following table describes the attributes in the response:

Field	Type	Definition
id	String	ID of the asset. Assets include reference data sets, code lists, crosswalks, and hierarchies. For more information, see "Asset IDs" on page 117.
name	String	Name of the asset.

Field	Type	Definition
description	String	Optional. Description of asset.
version	String	Optional. Version of the code list.
domain	String	Optional. Domain of the asset.
confidentiality	String	Optional. Confidentiality of the asset.
priority	String	Optional. Priority of the asset.
application	String	Optional. Application that uses the code list.
priority	String	Optional. Priority of the asset.
status	String	Optional. Status of the asset.
effectiveDate	String	Optional. Date the asset became effective.
approvedById	String	Optional. ID of the approver of the asset.
approvedByName	String	Optional. Username of the user who approved the asset.
approvedOn	String	Optional. Date the asset was approved.

GET example

To retrieve the list of all hierarchies, you might use the following request:

```
GET https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/hierarchies
HTTP/1.1
Accept: application/json
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXXX
```

The following sample response shows hierarchies:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 452

[
  {
    "id":"6e7dd28fc13b417c5c19d1fb",
    "name":"Cost centers",
    "description":"Cost center hierarchy",
    "version":"v1",
    "domain":"International standards",
    "confidentiality":"private",
    "priority":"Prio1",
    "status":"Draft",
    "effectiveDate":"2017-04-01",
    "approvedOn":"2017-03-01"
  },
  {
    "id":"e3e57e39ea8623f258013c43",
    "name":"Profit centers",
    "description":"Profit center hierarchy",
    "version":"v1"
  }
]
```

List hierarchy details

Retrieves the details of a hierarchy, such as the properties and status.

GET request

To retrieve the details of a hierarchy, submit a GET request with the following URI:

```
/rdm-service/external/v1/hierarchies/<hierarchy ID>
```

GET response

The response contains the details of the hierarchy.

The following table describes the attributes in the response:

Field	Type	Definition
id	String	ID of the asset. Assets include reference data sets, code lists, crosswalks, and hierarchies. For more information, see "Asset IDs" on page 117.
name	String	Name of the asset.
description	String	Optional. Description of asset.
version	String	Optional. Version of the code list.
domain	String	Optional. Domain of the asset.
confidentiality	String	Optional. Confidentiality of the asset.
priority	String	Optional. Priority of the asset.
application	String	Optional. Application that uses the code list.
priority	String	Optional. Priority of the asset.
status	String	Optional. Status of the asset.
effectiveDate	String	Optional. Date the asset became effective.
approvedById	String	Optional. ID of the approver of the asset.
approvedByName	String	Optional. Username of the user who approved the asset.
approvedOn	String	Optional. Date the asset was approved.

GET example

To retrieve the hierarchy details, you might use the following request:

```
GET https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/hierarchies/
6e7dd28fc13b417c5c19d1fb HTTP/1.1
Accept: application/json
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXXX
```

The following sample response shows hierarchies:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 334
```

```
{
  "id":"6e7dd28fc13b417c5c19d1fb",
  "name":"Cost centers",
  "description":"Cost center hierarchy",
  "version":"v1",
  "application":"App1",
  "domain":"Internal standards",
  "confidentiality":"private",
  "priority":"Prio1",
  "status":"Draft",
  "effectiveDate":"2017-04-01",
  "approvedOn":"2020-03-01"
}
```

List hierarchy model relationships

Retrieves the relationships in a hierarchy model.

GET request

To retrieve the relationships in a hierarchy model, submit a GET request with the following URI:

```
/rdm-service/external/v1/hierarchies/<hierarchy ID>/relations
```

GET response

The response contains the relationships and the code lists in each relationship. The `child.codeListId` attribute contains the top-level node relationship.

The following table describes the attributes in the response:

Field	Type	Definition
relations	-	Lists the code list in the relationship.
child	-	Contains information about the child code list.
parent	-	Contains information about the parent code list.
codeListId	String	ID of the code list.
codeListName	String	Name of the code list.
termId	String	ID of the reference data set to which the code list is associated.
termName	String	Name of the reference data set.

GET example

To retrieve the relationships in a hierarchy model, you might use the following request:

```
GET https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/hierarchies/
6e7dd28fc13b417c5c19d1fb/relations HTTP/1.1
Accept: application/json
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXXX
```

The following sample response shows hierarchies:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 452

{
```

```
  "relations":[
    {
      "child":{
        "codeListId":"8bc955e614df2040968e9d85",
        "codeListName":"Parent Codelist",
        "termId":"28a1320fe7f63cd25b58bef4",
        "termName":"Reference Data Set"
      }
    },
    {
      "parent":{
        "codeListId":"8bc955e614df2040968e9d85",
        "codeListName":"Parent Codelist",
        "termId":"28a1320fe7f63cd25b58bef4",
        "termName":"Reference Data Set"
      },
      "child":{
        "codeListId":"81a714a66863f954a9b60045",
        "codeListName":"First Level Codelist",
        "termId":"28a1320fe7f63cd25b58bef4",
        "termName":"Reference Data Set"
      }
    }
  ]
}
```

The `child.codeListId` attribute contains the top-level node relationship. For example, the first relationship in the example is the top-level node relationship.

enums

Use this resource to list system reference data values and add system reference data values.

You can retrieve system reference data values or add values to the following system reference data:

- Application
- Confidentiality
- Domain
- Priority
- Status

List system reference data values

Retrieves values of the system reference data.

GET request

To retrieve system reference data values, submit a GET request with the following URI:

```
/rdm-service/external/v1/enums
```

GET response

The response contains the values of all the system reference data.

The following table describes attributes in the response:

Field	Type	Description
key	String	ID of the system reference data value. In Reference 360, when you sort assets in the **Explore** panel by priority, the assets are sorted based on the key value. **Note:** When you import code values, you might want to provide additional information about the data. For example, you might want to assign a code value to the Approved status. You use the key value to assign the appropriate system reference data value.
label	String	Label for the system reference data value. The labels appear in Reference 360. In Reference 360, the values appear in alphanumeric order based on the label.

GET example

To retrieve the values of all the system reference data, you might use the following request:

```
GET https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/enums HTTP/1.1
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXX
```

The following sample response shows the values for each system reference data:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 368

{
   "Priority":[
      {
         "key":"0",
         "label":"Low"
      },
      {
         "key":"1",
         "label":"Medium"
      },
      {
         "key":"2",
         "label":"High"
      },
      {
         "key":"9",
         "label":"Critical"
      }
   ],
   "Domain":[
      {
         "key":"0",
         "label":"Finance"
      },
      {
         "key":"1",
         "label":"Geography"
      },
      {
         "key":"2",
         "label":"Social"
      }
   ]
}
```

Add system reference data values

Adds values to the system reference data.

PATCH request

To add values to the system reference data, submit a PATCH request with the following URI:

```
/rdm-service/external/v1/enums
```

Use the following attributes in the request body to specify the new values:

Field	Type	Description
key	String	ID of the system reference data value. In Reference 360, when you sort assets in the **Explore** panel by priority, the assets are sorted based on the key value. **Note:** When you import code values, you might want to provide additional information about the data. For example, you might want to assign a code value to the Approved status. You use the key value to assign the appropriate system reference data value.
label	String	Label for the system reference data value. The labels appear in Reference 360. In Reference 360, the values appear in alphanumeric order based on the label.

Note: You can use the same value for the `key` and `label` parameters.

PATCH response

The response shows the number of values added.

The following table describes the attributes in the response:

Field	Type	Description
newEntries	Number	Number of values added.
existingEntries	Number	Number of existing values that were skipped.

PATCH example

To add values to the system reference data, you might use the following request:

```
PATCH https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/enums HTTP/1.1
Content-Type: application/json
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXXX

{
  "Priority":[
    {
      "key":"0",
      "label":"Low"
    },
    {
      "key":"1",
      "label":"Medium"
    },
    {
      "key":"2",
      "label":"High"
    },
    {
      "key":"9",
```

```
        "label":"Critical"
      }
    ],
    "Domain":[
      {
        "key":"0",
        "label":"Finance"
      },
      {
        "key":"1",
        "label":"Geography"
      },
      {
        "key":"2",
        "label":"Social"
      }
    ]
}
```

The following sample response shows the number of values added:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 47

{
  "newEntries" : 5,
  "existingEntries" : 2
}
```

Update system reference data values

Updates the values of the system reference data.

PATCH request

To update the values of the system reference data, submit a PATCH request with the following URI:

```
/rdm-service/external/v1/enums
```

The following table describes the parameter in the request:

Parameter	Description
Overwrite	Optional. Indicates whether to update the existing system reference data values with same keys. Value is `true` or `false`. Default is `false`.

The following table describes the attributes in the request:

Field	Type	Description
key	String	ID of the system reference data value. In Reference 360, when you sort assets in the **Explore** panel by priority, the assets are sorted based on the key value. **Note:** When you import code values, you can provide additional information about the data. For example, you can assign a code value to the Approved status. You can use the key value to assign the appropriate system reference data value.
label	String	Label for the system reference data value. In Reference 360, the values appear in alphanumeric order based on the label.

PATCH response

The response shows the number of values updated.

The following table describes the attributes in the response:

Field	Type	Description
newEntries	Number	Number of values added.
existingEntries	Number	Number of existing values that are skipped.
updatedEntries	Number	Number of existing values that are updated.

PATCH example

To update values of the system reference data, you can use the following request:

```
PATCH https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/enums?
overwrite=true HTTP/1.1
Content-Type: application/json
Host: localhost:8080
Content-Length: 368

{
   "Priority":[
      {
         "key":"0",
         "label":"Low"
      },
      {
         "key":"1",
         "label":"Medium"
      },
      {
         "key":"2",
         "label":"High"
      },
      {
         "key":"9",
         "label":"Critical"
      }
   ],
   "Domain":[
      {
         "key":"0",
         "label":"Finance"
      },
      {
         "key":"1",
         "label":"Geography"
      },
      {
         "key":"2",
         "label":"Social"
      }
   ]
}
```

The following sample response shows the number of values updated:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 71

{
   "newEntries":1,
```

```
    "existingEntries":4,
    "updatedEntries":2
}
```

Delete system reference data values

Deletes values of the system reference data.

DELETE request

To delete a value of the system reference data, submit a DELETE request with the following URI:

```
/rdm-service/external/v1/enum/{enumType}/{enumKey}
```

Note: You can delete only one system reference data value at a time.

The following table describes the attributes in the request:

Field	Type	Description
enumType	String	Type of system reference data.
enumKey	String	ID of the system reference data value.

DELETE response

A 204 no content response is returned.

DELETE example

To delete a value of the system reference data, you might use the following request:

```
DELETE https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/enums/status/
statusKey HTTP/1.1
Host: localhost:8080
```

The following sample response shows the no content response:

```
HTTP/1.1 204 No Content
```

Using REST APIs to import and export

You can use the Reference 360 REST APIs to import or export code values and value mappings in bulk.

REST APIs for import and export

You can use a set of REST APIs to import or export data, retrieve the status of an import job, or retrieve a failed import job report.

The following table describes the REST APIs for importing and exporting data:

REST API	Description
Import code values	Import code values into a code list.
Import value mappings	Import value mappings into a crosswalk.
Get import job status	Retrieve the status of an import job.
Get failed import job report	Retrieve an error report for a failed import job.
Export code values	Export the code values in a code list.
Export value mappings	Export the value mappings in a crosswalk.

Filter criteria

You can filter the reference data that you want to export. For example, you can export a filtered set of code values in a code list.

When you filter code values, you can only filter values in the Code attribute or values in attributes that are configured as display attributes. For example, you might want a filter to include values with `001` in the Code attribute.

Field types

The filter operators available depends on the field type of the attribute.

The following table describes the filter operators supported for each field type:

Field Type	Supported Filter Operators	Filter Values
Boolean	_equals _notEquals _isEmpty	Boolean
Decimal or Integer	_equals _notEquals _isEmpty _greaterThan _greaterThanEquals _lessThan _lessThanEquals	Number

Field Type	Supported Filter Operators	Filter Values
String	_equals _notEquals _isEmpty _startsWith _endsWith _contains _notContains	Text
Date	_equals _notEquals _isEmpty _from, _to, _range	ISO 8601 date or date and time For example, `2019-12-24` or `1969-12-15T14:17:04Z`.
Reference Data	_equals _notEquals _isEmpty _in	Values in the Code attribute or values in the display attributes for the reference data.

Filter examples

To filter assets with text fields that are empty, you might use the following filter operator:

```
{
   "textField":{
      "_isEmpty":true
   }
}
```

To filter assets with boolean fields that are equal to true, you might use the following filter operator:

```
{
   "booleanField":{
      "_equals":true
   }
}
```

To filter assets with number fields that are greater than 1 and less than 2, you might use the following filter operators:

```
{
   "numberField":{
      "_greaterThan":1,
      "_lessThan":3
   }
}
```

To filter assets with date fields between specified dates, you might use the following filter operators:

```
{
   "dateField":{
      "_from":"2019-01-01",
      "_to":"2019-06-15"
   }
}
```

To filter assets with date fields for a time range based on a reference date or time, you might use the following filter operators:

```
{
   "dateField":{
```

```
      "_range":{
         "_months":6,
         "_reference":"2020-01-01"
      }
   }
}
```

To filter assets with reference data attribute fields, you might use the following filter operators:

```
{
   "referenceDataAttributeField.name":{
      "_in":[
         "EUR",
         "USD"
      ]
   }
}
```

To use multiple field operators, you might use the `_and` or `_or` operators like the following example:

```
{
  "_and": [
    {
      "_or": [
        {
          "name": {
            "_startsWith": "G"
          }
        },
        {
          "name": {
            "_endsWith": "g"
          }
        }
      ]
    },
    {
      "currencyLookup.name": [
        "EUR"
      ]
    },
    {
      "founded": {
        "_to": "1951-08-29"
      }
    },
    {
      "population": {
        "_greaterThan": 8e7
      }
    },
    {
      "monarchic": false
    }
  ]
}
```

When you use a comma to separate operators inside a field, the comma acts like an `_and` operator. For example, the following examples filters for a name that starts with "Ger" and ends with "many", or equals "Japan":

```
{
  "_or": [
    {
      "name": {
        "_startsWith": "Ger","_endsWith": "many"
      }
    },
    {
      "name": {
        "_equals": "Japan"
```

```
        }
      }
    ]
}
```

Related Topics:

- "Exporting filtered code values" on page 247

Importing code values

You can import code values into a code list. After you start an import job, you can check the status of the import job. If the import job fails, you can retrieve an error report.

Before you begin, you must get a session ID and identify the asset ID. The session ID authenticates your requests. The asset ID is the ID of the code list to which you want to import code values. For more information, see "Session IDs" on page 116 and "Asset IDs" on page 117.

1. To import code values into a code list, use the Import code values REST API.

   For more information about the Import code values REST API, see "Import code values" on page 151.

   For example, the following request imports code values:

```
POST https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/import
HTTP/1.1
Content-Type: multipart/form-data; boundary=6o2knFse3p53ty9dmcQvWAIx1zInP11uCfbm
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXXX

--6o2knFse3p53ty9dmcQvWAIx1zInP11uCfbm
Content-Disposition: form-data; name=file; filename=import-code-values.csv


--6o2knFse3p53ty9dmcQvWAIx1zInP11uCfbm
Content-Disposition: form-data; name=importSettings
Content-Type: application/json;charset=UTF-8

{
  "delimiter":"COMMA",
  "textQualifier":"DOUBLE_QUOTE",
  "codepage":"UTF8",
  "dateFormat":"ISO",
  "containerType":"CODELIST",
  "containerId":"9ab3201990a54dcdc86f54cf",
  "startingRow":null
}
--6o2knFse3p53ty9dmcQvWAIx1zInP11uCfbm--
```

   The CSV file might contain the following header rows and data rows:

```
Name,Code
Name,Code
Afghanistan,AFG
Aland Islands,ALA
Albania,ALB
Algeria,DZA
American Samoa,ASM
```

   Note: The `containerId` attribute is the ID of the code list to which you want to import code values.

   For example, the Import code values REST API returns the following job ID and details about the import job:

```
{
  "jobId":"dd1b2018cb47cef99f8d0f42",
  "state":"INPROGRESS",
  "startTime":1561367377428,
  "numOfRecordsProcessed":100,
```

```
  "numOfRecordsFailed":25,
  "numOfRecordsSucceeded":75
}
```

2. To check the status of an import job, use the Get import job status REST API.

   For more information about the Get import job status REST API, see "Get import job status" on page 158.

   For example, the following request retrieves the status of an import job:

```
GET https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/import/job/
dd1b2018cb47cef99f8d0f42 HTTP/1.1
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXX
```

   For example, the Get import job status REST API returns the following status of the import job:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 193
{
  "jobId":"dd1b2018cb47cef99f8d0f42",
  "state":"INPROGRESS",
  "startTime":1561367376330,
  "numOfRecordsProcessed":100,
  "numOfRecordsFailed":25,
  "numOfRecordsSucceeded":75
}
```

3. To retrieve an error report for a failed import job, use the Get failed import job report REST API.

   For more information about the Get failed import job report REST API, see "Get failed import job report" on page 158.

   For example, the following request retrieves the error report for a failed import job:

```
GET https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/import/job/
dd1b2018cb47cef99f8d0f42/errorDetails HTTP/1.1
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXX
```

   For example, the Get failed import job report REST API returns the following details of the failed import job:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 354

{
  "jobId":"dd1b2018cb47cef99f8d0f42",
  "entityType":"Relationship",
  "fileName":"import.csv",
  "entityName":"rdm.crosswalk.rel.21ffd6b5f92d10c744acc27c.fc66c441288cf898c6fe5023",
  "errorDetails":[
    {
      "lineNumber":1,
      "entitySourcePkey":"AF_AFG",
      "reasons":[
        "The requested resource with ID 'AFG' does not exist."
      ]
    }
  ]
}
```

Importing value mappings

You can import value mappings into a crosswalk. After you start an import job, you can check the status of the import job. If the import job fails, you can retrieve an error report.

Before you begin, you must get a session ID and identify the asset ID. The session ID authenticates your requests. The asset ID is the ID of the crosswalk to which you want to import value mappings. For more information, see "Session IDs" on page 116 and "Asset IDs" on page 117.

1. To import value mappings into a crosswalk, use the Import value mappings REST API.

   For more information about the Import value mappings REST API, see "Import value mappings" on page 153.

   For example, the following request imports value mappings:

   ```
   POST https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/import
   HTTP/1.1
   Content-Type: multipart/form-data; boundary=6o2knFse3p53ty9dmcQvWAIx1zInP11uCfbm
   IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXXX

   --6o2knFse3p53ty9dmcQvWAIx1zInP11uCfbm
   Content-Disposition: form-data; name=file; filename=import-value-mappings.csv

   --6o2knFse3p53ty9dmcQvWAIx1zInP11uCfbm
   Content-Disposition: form-data; name=importSettings
   Content-Type: application/json;charset=UTF-8

   {
     "delimiter":"COMMA",
     "textQualifier":"DOUBLE_QUOTE",
     "codepage":"UTF8",
     "dateFormat":"ISO",
     "containerType":"CROSSWALK",
     "containerId":"9ab3201990a54dcdc86f53AB",
     "startingRow":null
   }
   --6o2knFse3p53ty9dmcQvWAIx1zInP11uCfbm--
   ```

   The CSV file might contain the following header rows and data rows:

   ```
   sourcePKey,_from.id.sourcePKey,_to.id.sourcePKey
   sourcePKey,_from.id.sourcePKey,_to.id.sourcePKey
   AF_AFG,AF,AFG
   AL_ALA,AL,ALA
   ALB_ALB,ALB,ALB
   DZ_DZA,DZ,DZA
   AS_ASM,AS,ASM
   ```

   Note: The `containerId` attribute is the ID of the code list to which you want to import value mappings.

   For example, the Import value mappings REST API returns the following job ID and import job information:

   ```
   {
     "jobId":"dd1b2018cb47cef99f8d0f43",
     "state":"INPROGRESS",
     "startTime":1561367377428,
     "numOfRecordsProcessed":100,
     "numOfRecordsFailed":25,
     "numOfRecordsSucceeded":75
   }
   ```

   Note: You use the job ID to check the status of an import job.

2. To check the status of an import job, use the Get import job status REST API.

   For more information about the Get import job status REST API, see "Get import job status" on page 158.

For example, the following request retrieves the status of an import job:

```
GET https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/import/job/
dd1b2018cb47cef99f8d0f43 HTTP/1.1
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXX
```

For example, the Get import job status REST API returns the following status of the import job:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 193
{
  "jobId":"dd1b2018cb47cef99f8d0f43",
  "state":"INPROGRESS",
  "startTime":1561367376330,
  "numOfRecordsProcessed":100,
  "numOfRecordsFailed":25,
  "numOfRecordsSucceeded":75
}
```

3. To retrieve an error report for a failed import job, use the Get error report for failed import job REST API.

   For more information about the Get failed import job report REST API, see "Get failed import job report" on page 158.

   For example, the following request retrieves the error report for a failed import job:

```
GET https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/import/job/
dd1b2018cb47cef99f8d0f43/errorDetails HTTP/1.1
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXX
```

   For example, the Get error report for failed import job REST API returns the following error report:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 354

{
  "jobId":"dd1b2018cb47cef99f8d0f43",
  "entityType":"Relationship",
  "fileName":"import.csv",
  "entityName":"rdm.crosswalk.rel.21ffd6b5f92d10c744acc27c.fc66c441288cf898c6fe5023",
  "errorDetails":[
    {
      "lineNumber":1,
      "entitySourcePkey":"AF_AFG",
      "reasons":[
        "The requested resource with ID 'AFG' does not exist."
      ]
    }
  ]
}
```

Exporting code values

Export code values in a code list.

Before you begin, you must get a session ID and identify the asset ID. The session ID authenticates your requests. The asset ID is the ID of the code list that contains the code values that you want to export. For more information, see "Session IDs" on page 116 and "Asset IDs" on page 117.

- To export code values in a code list, use the Export code values REST API.

  For more information about the Export code values REST API, see "Exporting code values" on page 246.

  For example, the following request exports code values:

```
POST https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/export
HTTP/1.1
Content-Type: application/json
Accept: application/octet-stream
```

```
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXX

{
  "delimiter" : "SEMICOLON",
  "codepage" : "UTF8",
  "decimalSeparator" : "COMMA",
  "thousandSeparator" : "DOT",
  "dateFormat" : "ISO",
  "filename" : "testdata.csv",
  "containerType" : "codelist",
  "containerId" : "1989aae96bdaa4c2b8768fcc"
}
```

Note: The `containerId` attribute is the ID of the code list that contains the code values that you want to export.

For example, the Export code values REST API exports the following CSV file with the data:

```
HTTP/1.1 200 OK
Content-Disposition: attachment;filename=testdata.csv
Content-Type: application/octet-stream
Content-Length: 124

status.key;effectiveDate;approvedOn
status.status.key;effectiveDate;approvedOn
ActiveStatus;myEffectiveDate;myApprovedOn
ActiveStatus;myEffectiveDate;myApprovedOn
```

Exporting filtered code values

Export filtered code values based on filter criteria. You can filter code values based on values in attributes or reference data attributes.

Before you begin, you must get a session ID and identify the asset ID. The session ID authenticates your requests. The asset ID is the ID of the code list that contains the code values that you want to export. For more information, see "Session IDs" on page 116 and "Asset IDs" on page 117.

1. To export code values that contain a status, use the Export code values API with a filter operator for the status field.

   For more information about the Export code values REST API, see "Exporting code values" on page 246.

   For example, the following request exports code values with any status:

```
POST https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/export
HTTP/1.1
Content-Type: application/json
Accept: application/octet-stream
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXX

{
  "delimiter":"SEMICOLON",
  "codepage":"UTF8",
  "decimalSeparator":"COMMA",
  "thousandSeparator":"DOT",
  "dateFormat":"ISO",
  "filename":"testdata.csv",
  "containerType":"codelist",
  "containerId":"1989aae96bdaa4c2b8768fcc",
  "filter":{
    "_and":[
      {
        "Status":{
          "_isEmpty":false
        }
      }
    ]
  }
}
```

Note: The `containerId` attribute is the ID of the code list that contains the code values that you want to export.

For example, the Export code values REST API exports the following CSV file with the data:

```
HTTP/1.1 200 OK
Content-Disposition: attachment;filename=testdata.csv
Content-Type: application/octet-stream
Content-Length: 124

status.key;effectiveDate;Name;Code;Description
status.status.key;effectiveDate;Name;Code;Description
Active;;US;001;United States of America
Active;;CAN;002;Canada
```

2. To export filtered code values based on a display attribute for a reference data attribute and an attribute, use the Export code values API with multiple filter operators.

   For more information about the Export code values REST API, see "Exporting code values" on page 246.

   For example, the following request exports filtered code values that contain `Dollar` in the Name display attribute for the Currency reference data attribute and `00` in the Code attribute:

```
POST https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/export
HTTP/1.1
Content-Type: application/json
Accept: application/octet-stream
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXXX

{
  "delimiter":"SEMICOLON",
  "codepage":"UTF8",
  "decimalSeparator":"COMMA",
  "thousandSeparator":"DOT",
  "dateFormat":"ISO",
  "filename":"testdata.csv",
  "containerType":"codelist",
  "containerId":"1989aae96bdaa4c2b8768fcc",
  "filter":{
    "_and":[
      {
        "Currency.Name":{
          "_contains":"Dollar"
        }
      },
      {
        "Code":{
          "_contains":"00"
        }
      }
    ]
  }
}
```

   Note: The `containerId` attribute is the ID of the code list that contains the code values that you want to export.

   For example, the Export code values REST API exports the following CSV file with the data:

```
HTTP/1.1 200 OK
Content-Disposition: attachment;filename=testdata.csv
Content-Type: application/octet-stream
Content-Length: 124

status.key;effectiveDate;Name;Code;Description;Currency.Code
status.status.key;effectiveDate;Name;Code;Currency.Currency.Code
Active;;US;001;United States of America;USD
Active;;CAN;002;Canada;CAD
```

   Note: When you filter on a specific display attribute for a reference data attribute, the filtered code values appear in the CSV file, but the code value is represented by the Code attribute.

3. To export filtered code values based on a reference data attribute and an attribute, use the Export code values API with multiple filter operators.

   For more information about the Export code values REST API, see "Exporting code values" on page 246.

   For example, the following request exports filtered code values with `EUR` in the Currency reference data attribute and `an` in the Name attribute:

```
POST https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/export
HTTP/1.1
Content-Type: application/json
Accept: application/octet-stream
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXXX

{
  "delimiter":"SEMICOLON",
  "codepage":"UTF8",
  "decimalSeparator":"COMMA",
  "thousandSeparator":"DOT",
  "dateFormat":"ISO",
  "filename":"testdata.csv",
  "containerType":"codelist",
  "containerId":"1989aae96bdaa4c2b8768fcc",
  "filter":{
    "_and":[
      {
        "Currency":"EUR"
      },
      {
        "Name":{
          "_contains":"an"
        }
      }
    ]
  }
}
```

   Note: The `containerId` attribute is the ID of the code list that contains the code values that you want to export.

   For example, the Export code values REST API exports the following CSV file with the data:

```
HTTP/1.1 200 OK
Content-Disposition: attachment;filename=testdata.csv
Content-Type: application/octet-stream
Content-Length: 124

status.key;effectiveDate;Name;Code;Description;Currency.Code
status.status.key;effectiveDate;Name;Code;Description;Currency.Code
;;Netherlands;NLD;Netherlands;;EUR
;;Germany;DEU;Germany;;EUR
;;Ireland;IRL;Ireland;;EUR
;;Finland;FIN;Finland;;EUR
```

   Note: When you filter on values in a reference data attribute without specifying a display attribute to filter on, the filter applies on values in the Code attribute.

Related Topics:

- "Filter criteria" on page 240
- "Display attributes" on page 26
- "Reference data attributes" on page 25

Exporting value mappings

Export value mappings in a crosswalk.

Before you begin, you must get a session ID and identify the asset ID. The session ID authenticates your requests. The asset ID is the ID of the crosswalk that contains the value mappings that you want to export. For more information, see "Session IDs" on page 116 and "Asset IDs" on page 117.

- To export value mappings in a crosswalk, use the Export value mappings REST API.

  For more information about the Import value mappings REST API, see "Export value mappings to a CSV file" on page 172.

  For example, the following request exports value mappings:

```
POST https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/export
HTTP/1.1
Content-Type: application/json
Accept: application/octet-stream
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXXX

{
  "delimiter" : "SEMICOLON",
  "codepage" : "UTF8",
  "decimalSeparator" : "COMMA",
  "thousandSeparator" : "DOT",
  "dateFormat" : "ISO",
  "filename" : "testdata.csv",
  "containerType" : "crosswalk",
  "containerId" : "5d123f6e4077c700010d59e4"
}
```

  Note: The `containerId` attribute is the ID of the crosswalk that contains the value mappings that you want to export.

  For example, the Export value mappings REST API exports the following CSV file with the data:

```
HTTP/1.1 200 OK
Content-Disposition: attachment;filename=testdata.csv
Content-Type: application/octet-stream
Content-Length: 124

status.key;effectiveDate;approvedOn
status.status.key;effectiveDate;approvedOn
ActiveStatus;myEffectiveDate;myApprovedOn
ActiveStatus;myEffectiveDate;myApprovedOn
```

Using REST APIs to manage hierarchies

You can use the Reference 360 REST APIs to manage hierarchies.

REST APIs to manage hierarchies

You can use a set of REST APIs to list hierarchies, list hierarchy details, list hierarchy model relationships, and import hierarchy relationships.

The following table describes the REST APIs for managing hierarchies:

REST API	Description
List hierarchies	Retrieves all hierarchies.
List hierarchy details	Retrieves the details of a hierarchy, such as the properties and status.
List hierarchy model relationships	Retrieves the relationships in a hierarchy model.
List hierarchy relationships	Imports top-level code values and relationships into a hierarchy.

Managing hierarchies

You can create hierarchies to show hierarchical relationships between code values in multiple code lists. A hierarchy consists of two components: the hierarchy model and the hierarchy tree. In the hierarchy model, you define the top-level code list and add relationships to other code lists. Then based on the hierarchy model, you can create the hierarchy tree and define relationships between the code values in the code lists.

For example, you might create a location hierarchy. First, you define the hierarchy model. You define the Region code list as the top-level code list. Then you create a parent-child relationship from the Region code list to the Enterprise Country Codes code list. Based on this hierarchy model, in the hierarchy, you create a hierarchy relationship from the North America code value to the United States code value. You create a hierarchy relationship from the North America code value to the Canada code value.

To create and manage hierarchies, perform the following actions:

1. In Reference 360, create the hierarchy asset and define the hierarchy model. For more information, see "Creating hierarchy models" on page 89.
2. Use the List hierarchies REST API to retrieve all hierarchies.
3. Use the List hierarchy model relationships REST API to retrieve the relationships in a hierarchy model.
4. Use the Import hierarchy relationships REST API to import relationships in a hierarchy tree.

Step 1. List hierarchies

You can retrieve all hierarchies in Reference 360.

Before you begin, you must get a session ID. The session ID authenticates your requests. For more information, see "Session IDs" on page 116.

1. To retrieve all hierarchies, use the List hierarchies REST API.

   For more information about the List hierarchies REST API, see "List hierarchies" on page 230.

   For example, the following request retrieves all hierarchies:

```
GET https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/hierarchies
HTTP/1.1
Accept: application/json
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXXX
```

For example, the List hierarchies REST API returns the following hierarchies:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 452
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXXX

[
  {
    "id":"6e7dd28fc13b417c5c19d1fb",
    "name":"Cost centers",
    "description":"Cost center hierarchy",
    "version":"v1",
    "domain":"International standards",
    "confidentiality":"private",
    "priority":"Prio1",
    "status":"Draft",
    "effectiveDate":"2017-04-01",
    "approvedOn":"2017-03-01"
  },
  {
    "id":"e3e57e39ea8623f258013c43",
    "name":"Profit centers",
    "description":"Profit center hierarchy",
    "version":"v1"
  }
]
```

Note: The `id` attribute is the ID of the hierarchy, which you require for other hierarchy REST APIs.

2. Optionally, to retrieve the details of a hierarchy, use the List hierarchy details REST API.

   For more information about the List hierarchy details REST API, see "List hierarchy details" on page 232.

   For example, the following request retrieves the details of a hierarchy:

```
GET https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/
hierarchies/6e7dd28fc13b417c5c19d1fb HTTP/1.1
Accept: application/json
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXXX
```

   For example, the List hierarchy details REST API returns the following hierarchy details:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 334

{
  "id":"6e7dd28fc13b417c5c19d1fb",
  "name":"Cost centers",
  "description":"Cost center hierarchy",
  "version":"v1",
  "application":"App1",
  "domain":"Internal standards",
  "confidentiality":"private",
  "priority":"Prio1",
  "status":"Draft",
  "effectiveDate":"2017-04-01",
  "approvedOn":"2020-03-01"
}
```

Step 2. List hierarchy model relationships

You can retrieve the top-level code list and the relationships between code lists in a hierarchy model. Then, based on the hierarchy model, you can import hierarchy relationships between code values in the code lists.

- To retrieve the relationships in a hierarchy model, use the List hierarchy model relationships REST API.

  For more information about the List hierarchy model relationships REST API, see "List hierarchy model relationships" on page 233.

For example, the following request retrieves the relationships in a hierarchy model:

```
GET https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/
hierarchies/6e7dd28fc13b417c5c19d1fb/relations HTTP/1.1
Accept: application/json
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXXX
```

For example, the List hierarchy model relationship REST API returns the following relationships:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 452

{
  "relations":[
    {
      "child":{
        "codeListId":"8bc955e614df2040968e9d85",
        "codeListName":"Parent Codelist",
        "termId":"28a1320fe7f63cd25b58bef4",
        "termName":"Reference Data Set"
      }
    },
    {
      "parent":{
        "codeListId":"8bc955e614df2040968e9d85",
        "codeListName":"Parent Codelist",
        "termId":"28a1320fe7f63cd25b58bef4",
        "termName":"Reference Data Set"
      },
      "child":{
        "codeListId":"81a714a66863f954a9b60045",
        "codeListName":"First Level Codelist",
        "termId":"28a1320fe7f63cd25b58bef4",
        "termName":"Reference Data Set"
      }
    }
  ]
}
```

The `child.codeListId` attribute contains the top-level node relationship. For example, the first relationship in the example is the top-level node relationship.

Step 3. Import hierarchy relationships

You can import top-level code values and parent-child relationships into a hierarchy. For example, in a locations hierarchy, you might define the North America code value as a top-level code value. Then you define a relationship from the North America code value to the United States code value. You might also define a relationship from the North America code value to the Canada code value.

1. To import top-level code values into a hierarchy, use the Import hierarchy relationships REST API to specify the CSV file that contains the code values.

   For more information about the Import hierarchy relationships REST API, see "Step 3. Import hierarchy relationships" on page 253.

   For example, the following request imports top-level code values:

```
POST https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/import/
hierarchy HTTP/1.1
Content-Type: multipart/form-data; boundary=6o2knFse3p53ty9dmcQvWAIx1zInP11uCfbm
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXXX

--6o2knFse3p53ty9dmcQvWAIx1zInP11uCfbm
Content-Disposition: form-data; name=file; filename=import.csv
Content-Type: text/plain

--6o2knFse3p53ty9dmcQvWAIx1zInP11uCfbm
Content-Disposition: form-data; name=importSettings
```

```
Content-Type: application/json;charset=UTF-8

{
  "delimiter":"COMMA",
  "textQualifier":"DOUBLE_QUOTE",
  "startingRow":0,
  "codepage":"UTF8",
  "hierarchyId":"c79ab91c19b13b11d8d43770",
  "childCodeListId":"96f06071e4aaea81ff203abe"
}
--6o2knFse3p53ty9dmcQvWAIx1zInP11uCfbm--
```

The CSV file might contain the following header row and data rows:

```
Code
P1
P2
P3
```

For example, the Import hierarchy relationships REST API returns the following response:

```
{
  "jobId":"73580d323feb170be5ec0fd5",
  "state":"INPROGRESS",
  "startTime":1603092643055,
  "numOfRecordsProcessed":100,
  "numOfRecordsFailed":25,
  "numOfRecordsSucceeded":75
}
```

2. To import relationships, use the Import hierarchy relationships REST API to specify the CSV file that contains the relationships.

   For more information about the Import hierarchy relationships REST API, see "Step 3. Import hierarchy relationships" on page 253.

   For example, the following request imports relationships:

```
POST https://use4-mdm.dm-us.informaticacloud.com/rdm-service/external/v1/import/
hierarchy HTTP/1.1
Content-Type: multipart/form-data; boundary=6o2knFse3p53ty9dmcQvWAIx1zInP11uCfbm
IDS-SESSION-ID: XXXXXXXXXXXXXXXXXXXXX

--6o2knFse3p53ty9dmcQvWAIx1zInP11uCfbm
Content-Disposition: form-data; name=file; filename=import.csv
Content-Type: text/plain

--6o2knFse3p53ty9dmcQvWAIx1zInP11uCfbm
Content-Disposition: form-data; name=importSettings
Content-Type: application/json;charset=UTF-8

{
  "delimiter":"COMMA",
  "textQualifier":"DOUBLE_QUOTE",
  "startingRow":0,
  "codepage":"UTF8",
  "hierarchyId":"c79ab91c19b13b11d8d43770",
  "childCodeListId":"96f06071e4aaea81ff203abe",
  "parentCodeListId":"9b300f793470882ca23e6091"
}
--6o2knFse3p53ty9dmcQvWAIx1zInP11uCfbm--
```

   The CSV file might contain the following header row and data rows:

```
Code,ParentCode
C1,P1
C2,P2
C3,P3
```

   For example, the Import hierarchy relationships REST API returns the following response:

```
{
  "jobId":"73580d323feb170be5ec0fd5",
```

```
  "state":"INPROGRESS",
  "startTime":1603092643055,
  "numOfRecordsProcessed":100,
  "numOfRecordsFailed":25,
  "numOfRecordsSucceeded":75
}
```

CHAPTER 14

Glossary

attribute

A component of a code value. Code values have two or more attributes.

code list

A grouping of reference data that comes from the same application, industry standard list, or internal list. You organize code lists into reference data sets.

code value

A unique value, such as a business term. A code value is the lowest representation of reference data. You create code values in code lists.

crosswalk

A visual representation of a one-way relationship between code values in a pair of code lists. Crosswalks exist between a pair of code lists in the same reference data set.

dependent code list

A code list that contains code values that depend on code values from a code list in a different reference data set.

dependent reference data set

A reference data set that depends on code values in another reference data set. Code lists inherit the dependency from the reference data set.

hierarchical code lists

A code list that supports hierarchical data structures. You can arrange its code values into levels to create hierarchies.

hierarchical reference data set

A reference data set that supports hierarchical data structures. Code lists inherit the hierarchical structure from the reference data set. You can arrange code values in a hierarchical reference data set into levels to create hierarchies.

hierarchy

An arrangement of code values from multiple code lists below, above, or at the same level as other code values. The code values that you can arrange depend on the hierarchy model.

For example, you might have a hierarchy model with the Region code list defined as the top level node and a relationship from the Region code list to the Country code list. Then in the hierarchy, you can create a hierarchy relationship from the North America code value to the United States code value. You might also create a hierarchy relationship from the North America code value to the Canada code value.

hierarchy model

A model in which nodes are organized into a tree-like structure. A hierarchy model contains a top-level node, child nodes, a relationship from the top-level node to a child node, and relationships from child nodes to other child nodes. Each node in the model corresponds to a code list.

For example, you might create a location hierarchy. In the hierarchy model, you define the Region code list as the top-level node. You add the Country code list as a child node and then create a hierarchy model relationship from the Region node to the Country node.

reference data set

A logical grouping of reference data. A reference data set represents a category of reference data and acts as a template and container for code lists.

value mapping

The process of creating a one-way relationship between code values in the source code list and code values in the target code list. Value mappings provide a way to translate code values in the source code list to code values in the target code list.

INDEX

S

T

U

V

W