Canto Infinite Platform

Developer Guide

It is assumed the reader is familiar with the concept of web service software technologies and in particular RESTful web services. You should be familiar with programming concepts related to the HTTP protocol, and data exchange mechanisms such as JSON or XML.

The Canto Integration Platform (CIP) is a web service that runs inside a web application container such as Apache Tomcat. The service is providing functionality via HTTP requests of type GET and POST. Unlike other RESTful web services CIP does not use the HTTP requests DELETE, LIST to make it easier to get started by just using a web browser.

# Pre-Requisites

Before going any further, please ensure you have the following setup.

1. You must have a running a Cumulus 8 server. CIP works with both Workgroup and Enterprise editions of the Cumulus server. CIP does not work with Cumulus Single User, or versions of Cumulus earlier than Cumulus 8.
2. Your Cumulus Server needs an optional serial number for letting the CIP connect to it. This serial number needs to be activated.
3. Make sure your Cumulus server has a client license free to support write activities. CIP will reserve one or more licenses on startup (depending on your configuration).
4. Install CIP on some computer. It does not have to be the same as the one running the Cumulus Server.
5. For development you should have an up to date version of the Firefox browser with the “JSONView” plug-in installed. This will allow you to better see the results of some RESTful requests directly in the browser.
6. For developing your client-side code you can use whatever environment you feel comfortable with. CIP comes with a set of sample source files in different languages to be included in your project.

# Secure Communication (Optional)

You can configure your web application server (e.g. Apache Tomcat) to use SSL for communicating with any client-side software. This ensures that all data is encrypted and cannot be decoded by any attacker. The actual configuration depends on the web application server software used. The instructions for Apache Tomcat can be found here: <http://tomcat.apache.org/tomcat-6.0-doc/ssl-howto.html>

# Authentication (Optional)

When accessing the Cumulus DAM system CIP needs to connect using a valid Cumulus user name and password. There are several options available in CIP to be used (in the given order, if available):

1. Provide the user name and password in the request URL using the named parameters “user” and “password.”
2. Provide the user name and password in the request using HTTP Basic Access Authentication (<http://en.wikipedia.org/wiki/Basic_access_authentication>).
3. Provide a session id for a valid session in the request. The session had to be opened with valid credentials.
4. In the URL refer to a catalog alias definition in the configuration file that includes user name and password.
5. Without any of the above CIP will reply to the client according to the HTTP Basic Access Authentication protocol (<http://en.wikipedia.org/wiki/Basic_access_authentication>). In case the client software is a web browser it will ask the user to enter name and password for the realm “CIP.”

# Session Handling (Optional)

Besides the typical HTTP Request-Response protocol CIP supports keeping a session for answering requests. The session is identified by a session id and can store credentials and collections of items to be used across several requests. This way you only need to provide the credentials when opening the session and subsequent requests do not need the credentials anymore.

To start a session you call the “open” operation of the “session” service. This will create a new session at the CIP and will return a “jsessionid” cookie as well as the “jsessionid” value in the response data. The value of “jsessionid” is an arbitrary string that your client software needs to keep.

Your client software then provides the “jsessionid” with every subsequent request. You can use either the HTTP cookie mechanism or URL rewriting to provide the “jsessionid” value to CIP.

When you are done with the session you can call the “close” operation of the “session” service to close the session at the server (also providing the “jsessionid”). Every attempt to issue another request for the same session will fail after the session is closed.

# Reference Documentation

The operations (functions) that CIP provides to clients are grouped into different “services.” Each service offers a set of operations that conceptually belong together. The URL for an operation always starts with the following:

<http://cip-server/cip/service/operation/path-parameters?named-parameters>

The “cip-server” and “port” are the address and port number of the CIP web application server.

The “cip” specifies the web application “CIP”

The “service” is the name of the CIP server (always in lower case)

The “operation” is the name of operation inside the service (always in lower case)

The “path-parameters” are the optional path elements that the operation uses. Typically only parameters that need to occur exactly once for the operation are given as path-parameters. If a parameter is optional or can occur more than once it is typically provided using named parameters.

The “named-parameters” are optional parameters for the operation. Typically parameters that are optional or can occur more than once are passed this way. As usual in URLs named parameters are passed using “name=value” notation using a “&” to separate them.

Here is an overview of the services that CIP provides:

session

Creating and closing a session

metadata

Operations for searching, reading, and writing asset metadata

preview

Operations for generating images of assets in different sizes

asset

Operations for getting or uploading assets

developer

Operations for helping developers (e.g. generating client source code)

# Session Service

This service provides operations to create and close server-side sessions. A session can store the credentials for later use in any subsequent request for the same session. Any credentials provided explicitly with a request take precedence over session credentials.

## /session/open

Description

Open a new session at the server. You can provide user name and password to store them in the session for subsequent requests.

The credentials can be provided in two different ways:

1. As named URL parameters “user” and “password”
2. In the HTTP request header following the HTTP “Basic Access Authentication” standard

The first option is more flexible as it can also handle user names and passwords that contain colon (“:”) characters.

If you want to hide the credentials and encrypt all communication you should configure your web application server to use SSL. Then each URL you call needs to start with “https” instead of “http.”

Named Parameters

user

Optional: the Cumulus user name to be used for later catalog access

password

Optional: the Cumulus user’s password to be used for later catalog access

Result

The result contains the session id of the newly created session. The session id is returned as a HTTP cookie “jsessionid” as well as in the response text.

Example Request

<http://cip-server/cip/session/open?user=myself&password=secret>

Example Response

{

jsessionid: "F045F54785152CE916F8ABA6916D22F4"

}

## /session/close

Description

Close an existing session at the server. Any subsequent request for this session will fail after this operation is executed. You need to provide a session id (see section about session handling above) for this operation.

Result

None.

Example Request

<http://cip-server:port/cip/session/close>

# Metadata Service

The main purpose of the metadata service is to provide operations for searching, and retrieving and modifying metadata.

When searching you have the following options to keep the result:

1. Immediately return all IDs of the resulting items.
2. Immediately return metadata fields for each of the items of the result.
3. Store the resulting item IDs in a collection by optionally combining this search result with the previous contents of the collection. You then use the getfieldvalues operation to retrieve metadata for items in the collection. This way you can implement “paging” through a long list of items without returning metadata for all items found in a single operation.

###### Collections

The search operations allow you to optionally store the result in a named collection which is stored in the current session (see session handling above). You give the session a name which has to be unique within the current session. The collection is bound to a specific catalog and table with the catalog and only contains a list of item IDs.

After storing a search result in a collection you can then either retrieve the metadata for items in the collection using a getfieldvalues operation or combine the collection with the result of a subsequent search operation. Each search always returns the total number of items found. The getfieldvalues operation allow you to specify a starting offset in the item IDs and a maximum number of items to return. This way you can implement client-side “paging” through a long list of resulting items.

When you are done with a collection you can close it to release the memory it occupies in the CIP.

## /metadata/search

Description

Perform a search using a query defined in the CIP configuration file. You need to specify one of the parameters quicksearchstring, queryname, or querystring to perform a search.

A query configuration can be defined in three different ways:

1. A “QuickSearch” performed using a simple search string that searches across multiple metadata fields depending on the DAM system (use parameter quicksearchstring).
2. A preset query stored in the DAM system (use parameter queryname).  
   This is best suited for searches using a fixed query that is also used by other users of the DAM system.
3. A query string that may contain placeholders to be replaced with URL named parameters at runtime (use parameter queryname).  
   This is best suited for searches using a query with some variable parts that are specified in the request
4. A query string explicitly provided with the request (use parameter querystring).  
   This is best suited for any query that is created at runtime

Path Parameters

<http://cip-server:port/cip/metadata/search/catalog/view?...>

1. catalog

The name of a catalog alias definition from the configuration file. See the configuration section for details on how to define catalog aliases.

1. view

Optional: The name of view definition from the configuration file. If no view is specified then the resulting list of items is just an array of item IDs. See the configuration section on details on how to define views.

Named Parameters

quicksearchtext

(Optional) The text to perform a quicksearch with. The result depends on the DAM system and its configuration.

queryname

(Optional) The name of a query defined in the CIP configuration file. The defined query is either one that is preset in the DAM system or is configured using a query string with placeholders. See the configuration file section for details on how to define queries.

querystring

(Optional) The complete query string as expected by the DAM system. You need to make sure all special characters are correctly escaped if you want to pass this parameter in an HTTP GET request in the URL. It is recommended to use the HTTP POST request type to pass the parameter in the body of the request instead.

locale

(Optional) The two-letter language code (ISO 639-1) to be used for the metadata field values for the result. This parameter affects the way language-dependent metadata values are returned. For example you can specify “fr” to return all values suitable for French users. The default is the default locale the CIP server is running in (may be controlled using the “user.language” Java VM parameter when starting the web application server).

sortby

(Optional) A list of field names or field IDs separated by comma to specify the fields to be used for sorting the result. The default is that the result is not sorted by any field.

sortdirection

(Optional) The direction in which to sort the search result. Possible values are:

ascending (default)

descending

collection

(Optional) The name of a collection to save the resulting list of IDs. If you leave the value empty then CIP will create a unique collection name for you and will return this name in the result. This can be used for temporary collection to make sure the name is unique in the session.

When also using the parameter combine you can combine the existing collection contents with the result if this search operation.

combine

(Optional) This parameter is only used when using stored collection (see collection parameter above). It specifies how the result of this search operation is combined with the contents of the collection specified. If not specified the collection is created from this search operation’s result.

Possible values are

new

Do not use the previous contents of the specified collection but store the result of this search operation in the collection.

narrow

Only keep items in the collection that have been in the result of this search operation.

broaden

Add all items of the result of this search operation to the collection (if not contained already).

Result

The result is returned in JSON format and consists of the total number of items returned and a list of items with the field values defined in the given view. If no view and no collection are specified then the list of items is just an array of item IDs. If no view but a collection is specified the result just returns the total count and the name of the collection. The item field values or IDs can then be retrieved using the getfieldvalues operation.

Example Request

Perform a QuickSearch using the text “sample” and return the field values defined in the view “myfields” for all items found

<http://cip-server:port/cip/metadata/search/mycatalog/myfields?quicksearchstring=sample>

Example Configuration

<view name=’myfields’>

<field>Status</field>

<field>Record Name</field>

<field>Rating</field>

</view>

Example Response

{

"items": [

{

"Status": {

"displayString": "in progress",

"id": 0

},

"Record Name": "IMG\_3145.tif",

"Rating": 3

},

{

"Status": {

"displayString": "approved",

"id": 1

},

"Record Name": "IMG\_6345.tif",

"Rating": 5

}

],

"totalCount": 2

}

Example Request

Just return the IDs of the items found.

<http://cip-server:port/cip/metadata/search/mycatalog?quicksearchstring=sample>

Example Response

{

"items": [ 1745, 1867 ],

"totalCount": 2

}

Example Request

Store the result in a collection named “mycoll” instead of returning any items in the result.

<http://cip-server:port/cip/metadata/search/mycatalog?queryname=myquery&collection=mycoll>

Example Response

{

"collection": "mycoll",

"totalCount": 173493

}

Example Request

Sort the result by “Record Modification Date” descending:

<http://cip-server:port/cip/metadata/search/mycatalog?queryname=myquery&sortby=Record+Modification+Date&sortdirection=descending>

Example Request

Perform a search with the named query “myquery” which is defined in the CIP configuration file and return the metadata fields defined by the view “myfields.”

[http://cip-server:port/cip/metadata/search/mycatalog?queryname=fixedquery](http://cip-server:port/cip/metadata/search/mycatalog/myfields?queryname=fixedquery)

Example Configuration

<query name=’fixedquery’>

<serverqueryname>Fixed Query</serverqueryname>

</query>

Example Request

Perform a search with the named query “myquery” which is defined using a placeholder in the CIP configuration file and return the metadata fields defined by the view “myfields.”

[http://cip-server:port/cip/metadata/search/mycatalog?queryname=myquery&s=approved](http://cip-server:port/cip/metadata/search/mycatalog/myfields?queryname=myquery&s=approved)

Example Configuration

<query name=’myquery’>

<querystring>Status == ${s}</querystring>

</query>

## /metadata/getfieldvalues

Description

Retrieve the metadata fields or IDs of items in a stored collection. You can specify an offset to start at and a maximum number of items returned.

You specify the items either by setting the named parameter “collection” or by specifying the catalog and item ID using path parameters 1 and 2 respectively.

Collection-based request:

<http://cip-server:port/cip/metadata/getfieldvalues/view?collection=mycoll>

Catalog-based request:

<http://cip-server:port/cip/metadata/getfieldvalues/catalog/id/view>

Path Parameters

Collection-based request:

1. view

(Optional) The name of view definition from the configuration file. If no view is specified then the resulting list of items is just an array of item IDs if the result is not stored in a collection.

Catalog-based request:

1. catalog

The catalog alias for the catalog to work with.

1. id

The ID of the item in the catalog to return field values for.

1. view

(Optional) The name of view definition from the configuration file. If no view is specified then the resulting list of items is just an array of item IDs if the result is not stored in a collection.

Named Parameters

collection

(Optional) The name of an existing collection in the current session.

table

(Optional) If you do not specify a collection you may want to specify the table to return field values for. The default is “AssetRecords.”

startindex

(Optional) The index (zero-based) to start returning the items. Using this parameter you can page through the result list by starting with 0 and then incrementing by a given number. The default is 0 which returns the items starting with the first one.

maxreturned

(Optional) The maximum number of items returned by this operation. You may use this parameter to limit the size of the resulting JSON data. When used together with the startindex parameter you can implement paging through the result list. The default is to return all items starting at the one specified by the startindex parameter. Due to changes being encountered n the catalog this operation may return less than the specified number of items if items in the given range have been deleted from the catalog. However, to do proper paging you should start the next getfieldvalues operation at the index you calculate from the given parameters startindex and maxreturned.

locale

(Optional) The two-letter language code (ISO 639-1) to be used for the metadata field values for the result. This parameter affects the way language-dependent metadata values are returned. For example you can specify “fr” to return all values suitable for French users. The default is the default locale the CIP server is running in (may be controlled using the “user.language” Java VM parameter when starting the web application server).

Result

The result is returned in JSON format and consists of the total number of items returned and an optional list of items with IDs or field values defined in the given view. If no view is specified then the list of items is just an array of item IDs.

Example Request

Return the first 500 items of the collection “mycoll” with field values taken from view “myfields.”

<http://cip-server:port/cip/metadata/getfieldvalues/myfields?collection=mycoll&maxreturned=500>

Example Response

{

"items": [

{

"Status": {

"displayString": "in progress",

"id": 0

},

"Record Name": "IMG\_3145.tif",

"Rating": 3

},

{

"Status": {

"displayString": "approved",

"id": 1

},

"Record Name": "IMG\_6345.tif",

"Rating": 5

}

...

],

"totalCount": 173493

}

Example Request

Return all IDs of all items of the collection “mycoll.”

<http://cip-server:port/cip/metadata/getfieldvalues?collection=mycoll>

Example Response

{

"items": [ 1745, 1867 ],

"totalCount": 2

}

## /metadata/setfieldvalues

Description

Set the metadata fields of catalog items. The field values are specified using a JSON structure transferred in the request body of an HTTP POST request. The JSON data always contains the ID of the item to be modified.

Path Parameters

<http://cip-server:port/cip/metadata/setfieldvalues/catalog?...>

1. catalog

(Optional) The catalog alias for the catalog for the item to be modified.

Named Parameters

table

(Optional) The name of a table for the items to be modified. The default is “AssetRecords.”

locale

(Optional) The two-letter language code (ISO 639-1) to be used for the metadata field values. This parameter affects the way language-dependent metadata values are parsed. For example you can specify “fr” to specify all values suitable for French users. The default is the default locale the CIP server is running in (may be controlled using the “user.language” Java VM parameter when starting the web application server).

Result

The result does not have any contents.

Example Request

Set the “Status” and “Rating” field values of two asset records of ID 1537 and 638 to some values. In this example the “Rating” field for item 638 is cleared.

<http://cip-server:port/cip/metadata/setfieldvalues/mycatalog>

Example Request Body

{

"items": [

{

"ID": 1537,

"Status": 2,

"Rating": 3

},

{

"ID": 638,

"Status": 1,

"Rating": null

}

...

]

}

## /metadata/getcategories

Description

Get the sub-categories for a given root category. You can either get the direct sub-categories of the root only or the whole sub-tree.

Three options allow specifying the root category:

1. Specify the category by the complete path in the (use parameter path).
2. Specify the category by its ID (use parameter categoryid).
3. If neither the path nor categoryid parameter is specified the operation will return the top-level categories.

Path Parameters

<http://cip-server:port/cip/metadata/getcategories/catalog/view?...>

1. catalog

The name of a catalog alias definition from the configuration file. See the configuration section for details on how to define catalog aliases.

1. view

Optional: The name of view definition from the configuration file. If no view is specified then the resulting list of items is just an array of item IDs. See the configuration section on details on how to define views.

Named Parameters

path

(Optional) The complete path of the parent category in the tree starting at the top-level category name. The category names for each level are separated by colon. Use double-colon to escape a colon appearing in a category name.

categoryid

(Optional) The ID of the parent category.

subtree

(Optional) This parameter specifies whether you want the result to contain not just the direct sub-categories of the given parent but the whole sub-tree including all categories down to the bottom level.

Possible values are:

false

(Default) Return the direct sub-categories of the given root category only.

true

Return all the categories underneath the root category down to the bottom level. They are returned in “depth-first”. The result nests sub-categories inside their parent category item so that the tree structure can be reconstructed. If you specify a collection to store the result the collection will contain the category IDs as a flat list.

locale

(Optional) The two-letter language code (ISO 639-1) to be used for the metadata field values for the result. This parameter affects the way language-dependent metadata values are returned. For example you can specify “fr” to return all values suitable for French users. The default is the default locale the CIP server is running in (may be controlled using the “user.language” Java VM parameter when starting the web application server).

collection

(Optional) The name of a collection to save the resulting list of IDs. If you leave the value empty then CIP will create a unique collection name for you and will return this name in the result. This can be used for temporary collection to make sure the name is unique in the session.

When also using the parameter combine you can combine the existing collection contents with the result if this search operation.

Result

The result is returned in JSON format and consists of the total number of category items returned and a list of items with the field values defined in the given view. The sub-categories are returned as a list with the name subcategories. Empty sub-category arrays are suppressed in the output.

If no view and no collection are specified then the format of the result depends on the value of the parameter subtree. If only direct sub-categories are returned then the result is just an array of category IDs. If you wanted to get a whole sub-tree then the result is the same as if you would have specified the field “ID” as the only field of the view. If no view but a collection is specified the result just returns the total count and the name of the collection. The item field values or IDs can then be retrieved using the getfieldvalues operation.

Example Request

Get the all top-level categories and return the field values defined in the view “myfields” for all items found

<http://cip-server:port/cip/metadata/getcategories/mycatalog/myfields>

Example Configuration

<view name=’myfields’>

<field>Category Name</field>

<field>ID</field>

</view>

Example Response

{

"ID": 0,

"subcategories": [

{

"Category Name": "America",

"ID": 4

},

{

"Category Name": "Asia",

"ID": 5

},

{

"Category Name": "Europe",

"ID": 3

}

],

"totalCount": 3

}

Example Request

Just return the IDs of the top-level categories.

<http://cip-server:port/cip/metadata/getcategories/mycatalog>

Example Response

{

"items": [4, 5, 3 ],

"totalCount": 3

}

Example Request

Store the result in a collection named “topcats” instead of returning any items in the result.

<http://cip-server:port/cip/metadata/getcategories/mycatalog?collection=topcats>

Example Response

{

"collection": "topcats",

"totalCount": 3

}

Example Request

Return the sub-categories of a specific category (“Europe” above) as a tree.

<http://cip-server:port/cip/metadata/getcategories/mycatalog/myfields?categoryid=3>

Example Configuration

<view name=’myfields’>

<field>Category Name</field>

<field>ID</field>

</view>

Example Response

{

"ID": 3,

"subcategories": [

{

"Category Name": "Belgium",

"ID": 31

},

{

"Category Name": "France",

"ID": 37,

"subcategories": [

{

"Category Name": "Lyon",

"ID": 371

},

{

"Category Name": "Paris",

"ID": 576

}

]

},

{

"Category Name": "Italy",

"ID": 32,

"subcategories": [

{

"Category Name": "Rome",

"ID": 532

},

{

"Category Name": "Venice",

"ID": 936

}

]

}

],

"totalCount": 7

}

Example Request

Return the sub-category IDs for a specific root category (“Europe” above) as a tree.

<http://cip-server:port/cip/metadata/getcategories/mycatalog?categoryid=3>

Example Response

{

"ID": 3,

"subcategories": [

{

"ID": 31

},

{

"ID": 37,

"subcategories": [

{

"ID": 371

},

{

"ID": 576

}

]

},

{

"ID": 32,

"subcategories": [

{

"ID": 532

},

{

"ID": 936

}

]

}

],

"totalCount": 7

}

Example Request

Return the sub-categories of a specific category (“France” above).

<http://cip-server:port/cip/metadata/getcategories/mycatalog/myfields?path=Europe:France>

Example Configuration

<view name=’myfields’>

<field>Category Name</field>

<field>ID</field>

</view>

Example Response

{

"ID": 37,

"subcategories": [

{

"Category Name": "Lyon",

"ID": 371,

},

{

"Category Name": "Paris",

"ID": 576,

}

],

"totalCount": 2

}

## /metadata/createcategory

Description

Create a new category as a sub-category of a given other category.

Three options allow specifying the parent category:

1. Specify the parent category by the complete path in the (use parameter path).
2. Specify the parent category by its ID (use parameter categoryid).
3. If neither the path nor categoryid parameter is specified the operation will create a top-level category.

Path Parameters

<http://cip-server:port/cip/metadata/createcategory/catalog?...>

1. catalog

The name of a catalog alias definition from the configuration file. See the configuration section for details on how to define catalog aliases.

Named Parameters

path

(Optional) The complete path of the parent category in the tree starting at the top-level category name. The category names for each level are separated by colon. Use double-colon to escape a colon appearing in a category name.

categoryid

(Optional) The ID of the parent category.

name

The name for the newly created category.

Result

The result is returned in JSON format and consists of the ID of the newly created category.

Example Request

Create a new top-level category named “Countries”:

<http://cip-server:port/cip/metadata/createcategory/mycatalog?name=Countries>

Example Response

{

"ID": 534

}

Example Request

Create a new category underneath “Countries” named “Sweden”:

<http://cip-server:port/cip/metadata/createcategory/mycatalog?path=Countries&name=Sweden>

Example Response

{

"ID": 535

}

Example Request

Create a new category underneath “Sweden” called “Stockholm”:

<http://cip-server:port/cip/metadata/createcategory/mycatalog?path=Countries:Sweden&name=Stockholm>

Example Response

{

"ID": 536

}

## /metadata/deletecategory

Description

Delete a given category and all of its sub-categories.

Two options allow specifying the category to delete:

1. Specify the parent category by the complete path in the (use parameter path).
2. Specify the parent category by its ID (use parameter categoryid).

Path Parameters

<http://cip-server:port/cip/metadata/deletecategory/catalog?...>

1. catalog

The name of a catalog alias definition from the configuration file. See the configuration section for details on how to define catalog aliases.

Named Parameters

path

(Optional) The complete path of the category in the tree starting at the top-level category name. The category names for each level are separated by colon. Use double-colon to escape a colon appearing in a category name.

categoryid

(Optional) The ID of the category to delete.

Result

The result does not have any contents.

Example Request

Delete the category with the ID 536:

<http://cip-server:port/cip/metadata/deletecategory/mycatalog?categoryid=536>

Example Request

Delete the category “Sweden” underneath “Countries”:

<http://cip-server:port/cip/metadata/createcategory/mycatalog?path=Countries:Sweden>

## /metadata/gettables

Description

Return a list of all table names of a catalog.

Path Parameters

<http://cip-server:port/cip/metadata/gettables/catalog>

1. catalog

The name of a catalog alias definition from the configuration file. See the configuration section for details on how to define catalog aliases.

Named Parameters

None

Result

The result is the list of table names.

Example Request

Get the names of all tables of the catalog with the alias “mycatalog”:

<http://cip-server:port/cip/metadata/gettables/mycatalog>

Example Response

{

"tables": [

"AssetRecords",

"Categories"

]

}

## /metadata/getlayout

Description

Return a description of all the fields of a given table.

Path Parameters

<http://cip-server:port/cip/metadata/getlayout/catalog?...>

1. catalog

The name of a catalog alias definition from the configuration file. See the configuration section for details on how to define catalog aliases.

Named Parameters

table

(Optional) You may want to specify the table to return the layout for. The default is “AssetRecords.”

Result

The result is the list of field definitions of the given table.

Example Request

Get the asset records layout for the catalog with the alias “mycatalog”:

<http://cip-server:port/cip/metadata/getlayout/mycatalog>

Example Response

{

"fields": [

{

"name": "Record Name",

"type": "String"

},

…

{

"name": "Status",

"type": "String List"

},

]

}

## /metadata/assigntocategory

## /metadata/detachfromcategory

# Preview Service

## /preview/image

Description

Return the pixel preview of an asset. Several options allow specifying the parameters for generating the preview image. The options are applied in the following order:

1. Cropping (use optional parameters left, top, width, height).
2. Scale down the image (use optional parameter maxsize or size).
3. Rotate the image in 90 degree steps (use optional the parameter rotate).
4. Output file format (use optional parameter format and quality).

To improve the performance of returning preview images to the caller the CIP server caches the generated preview images.

Path Parameters

<http://cip-server:port/cip/preview/image/catalog/id?...>

1. catalog

The name of a catalog alias definition from the configuration file. See the configuration section for details on how to define catalog aliases.

1. id

The ID of the asset in the catalog specified by the first path parameter.

1. previewname

(Optional) The name of a preview configured in the CIP server. By using named previews you can keep the actual preview parameters separate from the CIP client code. Every parameter that you specify explicitly in the URL replaces a corresponding value from the configured preview.

Named Parameters

Cropping

left

(Optional) Specify the number of pixels to crop off the left side in original image space. The default is 0.

top

(Optional) Specify the number of pixels to crop off the top side in original image space. The default is 0.

width

(Optional) Specify the width of the cropping area in original image space. The default is the width of the original image.

height

(Optional) Specify the height of the cropping area in original image space. The default is the height of the original image.

top

left

height

width

Scaling

maxsize

(Optional) Scale the image down so that the longest side fits into the given number of pixels preserving the aspect ratio of the image.

maxsize

maxsize

size

(Optional) Scale down the image so that the shortest side fits into the square whose size is given in pixels. This option centers the image inside the square and crops away parts of the longer dimension. The resulting image is always square.

maxsize

maxsize

Rotating

rotate

(Optional) Rotate the image clockwise by the given degrees. Possible values are 0, 90, 180, and 270. Default is 0.

Output Format

format

(Optional) The output file format to be used. Possible values are "jpeg" and "png" with the default being "jpeg"

quality

(Optional) when using a file format that supports a lossy compression method (e.g. "jpeg") you can specify the quality level that you want to be preserved. The value ranges from "1" (least quality, smallest resulting data size) to "10" (best quality, largest resulting data size).

Cache Control

usecache

(Optional) Specifies how to use the preview cache. Possible values are

true

(Default) Return a cached preview if it exists and matches the asset modification date. Otherwise return a newly generated preview and store it in the cache.

only

Only generate the preview to store it in cache. No preview is returned to the caller.

false

Do not use the cache at all, return a newly generated preview.

Result

Unless you specify “usecache=only” the response body contains the image data you requested.

Example Request

Delete the category with the ID 536:

<http://cip-server:port/cip/metadata/deletecategory/mycatalog?categoryid=536>

## /preview/purgecache

Description

Purge specific cached preview files from the cache. You can purge all previews generated for a specific asset as well as previews generated with a specific parameter set.

Path Parameters

<http://cip-server:port/cip/preview/purgecache/catalog/id?...>

1. catalog

The name of a catalog alias definition from the configuration file. See the configuration section for details on how to define catalog aliases.

1. id

(Optional) The ID of the asset in the catalog specified by the first path parameter for which to purge all cache files. If this is not specified the operation purges all cache files for the preview parameters that are specified.

Named Parameters

This operation uses the same named parameters as the preview/image operation but it uses them only to determine the files in the cache that are to be purged.

Result

The result does not have any contents.

Example Request

Purge all cached previews for asset ID 724:

<http://cip-server:port/cip/preview/purgecache/mycatalog/724>

Example Request

Purge all cached previews generated with the parameters “size=100”:

<http://cip-server:port/cip/preview/purgecache/mycatalog?size=100>

## /preview/thumbnail

Description

Return the thumbnail image of an asset. Typically the thumbnail image is a small representation of the asset stored in the catalog itself so retrieving the thumbnail.

Path Parameters

<http://cip-server:port/cip/preview/thumbnail/catalog/id?...>

1. catalog

The name of a catalog alias definition from the configuration file. See the configuration section for details on how to define catalog aliases.

1. id

The ID of the asset in the catalog specified by the first path parameter.

Named Parameters

Scaling

maxsize

(Optional) Scale the image down so that the longest side fits into the given number of pixels preserving the aspect ratio of the image.

maxsize

maxsize

Rotating

rotate

(Optional) Rotate the image clockwise by the given degrees. Possible values are 0, 90, 180, and 270. Default is 0.

Output Format

format

(Optional) The output file format to be used. Possible values are "jpeg" and "png" with the default being "jpeg"

quality

(Optional) when using a file format that supports a lossy compression method (e.g. "jpeg") you can specify the quality level that you want to be preserved. The value ranges from "1" (least quality, smallest resulting data size) to "10" (best quality, largest resulting data size).

Result

The response body contains the image data you requested.

Example Request

Get the thumbnail of asset ID 724:

<http://cip-server:port/cip/preview/thumbnail/mycatalog/724>

Example Response

The asset contents (e.g. JPEG data stream).

# Asset Service

## /asset/import

Description

You can imports an asset either by referencing an existing asset accessible to the CIP server or by uploading the asset along with the request. Optionally you can also set metadata field values when importing the asset.

There are three ways of specifying the asset to be imported:

1. Specify an existing asset using a URL (specified in the location parameter).
2. Specify an existing asset based on a configured location and a relative path (specified in the location parameter).
3. Upload the asset with the HTTP request. This is done by putting the asset contents into the request body and using the HTTP POST method. If no metadata fields are to be set with this request the body only contains the asset contents and you specify the file name to be used in the location parameter. If you want to be able to set metadata fields along with the importing you need to post the request using the MIME type

If you also want to set metadata fields for the asset you specify the field values in JSON format in the request body and use the HTTP POST method.

If you want to both upload the asset with the request and set field values the request body needs to have a MIME type of multipart/form-data. This is compatible with the way web browsers upload files to a web server. The asset contents are then embedded in the request body as a “file” contents. If you also want to set metadata field value you also include a parameter item called “fields” with the JSON structure that contains the field values similar to the way the “metadata” “setfieldvalues” operation accepts them.

Path Parameters

<http://cip-server:port/cip/asset/import/catalog?...>

1. catalog

The name of a catalog alias definition from the configuration file. See the configuration section for details on how to define catalog aliases.

Named Parameters

location

(Optional) the location path for an existing asset. The path is either a complete URL that starts with a URL protocol like “ftp” or is based on a location defined in the configuration file. By using a configured location and a relative path you can hide details such as FTP passwords from the user of the service. When executed the location name is replaced with the configured location string.

options

(Optional) The name of a set of asset handling options defined in the configuration file. The exact options are Dam system dependent.

Result

The result is returned in JSON format and consists of the ID of the imported asset.

Example Request

Import an asset by uploading it with the request. The request needs to use POST method with the asset contents in the request body. The MIME type of the request content can be application/octet-stream.

<http://cip-server:port/cip/asset/import/mycatalog>

Example Request Body

------------8cd9fcb9f7e5fab

Content-Disposition: form-data; name="fields"

{

"Status": 0,

"Rating": 3,

"Notes": "Some notes on the newly added asset go here"

}

------------8cd9fcb9f7e5fab

Content-Disposition: form-data; name="file"; filename="MyImage.jpg"

Content-Type: application/octet-stream

... binary contents of the asset ...

------------8cd9fcb9f7e5fab--

Example Response

{

"ID": 2374

}

Example Request

Import a file named “image.jpg” directly from a preconfigured FTP server location “ftpuploads.”

<http://cip-server:port/cip/asset/import/mycatalog?location=ftpuploads/image.jpg>

Example Configuration

<location name=’ftpuploads’>

<ftp://myname:mypassword@ftp.mycompany.com/uploads>

</location>

Example Response

{

"ID": 2375

}

## /asset/download

Description

You can download an asset using the “download” operation of the “asset” service. If the asset is available in several versions you can optionally specify the version that you want to download.

Optionally you can convert the asset prior to downloading. This conversion

An optional location allows you to store the result in the local file system of the CIP server or on an FTP server instead of downloading the result to the client.

Path Parameters

<http://cip-server:port/cip/asset/download/catalog/id?...>

1. catalog

The name of a catalog alias definition from the configuration file. See the configuration section for details on how to define catalog aliases.

1. id

The ID of the asset in the catalog specified by the first path parameter.

Named Parameters

version

(Optional) the version of the asset to download. The default is to download the latest version.

location

(Optional) the location path of a folder to store the result. The path is either a complete URL that starts with a URL protocol like “ftp” or is based on a location defined in the configuration file. By using a configured location and a relative path you can hide details such as FTP passwords from the user of the service. When executed the location name is replaced with the configured location string.

conversion

(Optional) The name of a configured conversion to be applied before downloading. The configuration contains the parameters for the actual conversion.

Result

If you specified a location to store the resulting asset then the response body contains JSON data specifying the exact location.

If you did not specify a location then the response body contains the resulting asset contents.

Example Request

Download the original asset for the asset with the ID 1234 in the catalog whose alias is “mycatalog.”

<http://cip-server:port/cip/asset/download/mycatalog/1234>

Example Response

The asset contents (e.g. JPEG data stream for a JPEG image).

Example Request

Download the asset with the ID 1234 of the catalog whose alias is “mycatalog” after converting it to a PDF-. The example is based on a Cumulus asset action of name “Convert to PDF” to be configured.

<http://cip-server:port/cip/asset/download/mycatalog/1234?conversion=convert2pdf>

Example Configuration

<conversion name=’convert2pdf’>

<parameter name="CumulusAssetAction">Convert to PDF</parameter>

</conversion>

Example Response

The PDF that is the result of converting the given asset.

Example Request

Store the asset with the ID 1234 of the catalog whose alias is “mycatalog” to configured location “ftpdownloads.”

<http://cip-server:port/cip/asset/download/mycatalog/1234?location=ftpdownloads>

Example Configuration

<location name=’ftpdownloads’>

ftp://myname:mypassword@ftp.mycompany.com/downloads

</location>

Example Response

{

"location": "ftpdownloads/myimage.jpg"

}

## /asset/checkout

Description

The “checkout” operation allows you to lock the asset for further modifications and also to download the latest version of the asset or to store it at a given location.

An optional location allows you to store the current asset in the local file system of the CIP server or on an FTP server instead of downloading it to the client.

Path Parameters

<http://cip-server:port/cip/asset/checkout/catalog/id?...>

1. catalog

The name of a catalog alias definition from the configuration file. See the configuration section for details on how to define catalog aliases.

1. id

The ID of the asset in the catalog specified by the first path parameter.

Named Parameters

location

(Optional) the location path for an existing asset. The path is either a complete URL that starts with a URL protocol like “ftp” or is based on a location defined in the configuration file. By using a configured location and a relative path you can hide details such as FTP passwords from the user of the service. When executed the location name is replaced with the configured location string.

Result

If you specified a location to store the resulting asset then the response body contains JSON data specifying the exact location.

If you did not specify a location then the response body contains the current asset contents.

Example Request

Check out the asset with the ID 1234 in the catalog whose alias is “mycatalog” and download the current asset contents.

<http://cip-server:port/cip/asset/checkout/mycatalog/1234>

Example Request

Store the asset with the ID 1234 of the catalog whose alias is “mycatalog” to configured location “ftpdownloads.”

<http://cip-server:port/cip/asset/checkout/mycatalog/1234?location=ftpcheckouts>

Example Configuration

<location name=’ftpcheckouts’>

ftp://myname:mypassword@ftp.mycompany.com/checkouts

</location>

Example Response

{

"location": "ftpcheckouts/myimage.jpg"

}

## /asset/checkin

Description

After you have checked out an asset using the “checkout” operation you can check in a new version of the asset using this “checkin” operation. The operation has parameters similar to importing an asset but instead of adding a new asset to the catalog it adds a new version to an existing asset.

There are three ways of specifying the asset to be checked in:

1. Specify an existing asset using a URL (specified in the location parameter).
2. Specify an existing asset based on a configured location and a relative path (specified in the location parameter).
3. Upload the asset with the HTTP request. This is done by putting the asset contents into the request body and using the HTTP POST method. This request the body only contains the asset contents.

Path Parameters

<http://cip-server:port/cip/asset/import/catalog?...>

1. catalog

The name of a catalog alias definition from the configuration file. See the configuration section for details on how to define catalog aliases.

1. id

The ID of the asset in the catalog specified by the first path parameter.

Named Parameters

location

(Optional) the location path for an existing asset. The path is either a complete URL that starts with a URL protocol like “ftp” or is based on a location defined in the configuration file. By using a configured location and a relative path you can hide details such as FTP passwords from the user of the service. When executed the location name is replaced with the configured location string.

options

(Optional) The name of a set of asset handling options defined in the configuration file. The exact options are Dam system dependent.

comment

(Optional) the comment for this new version of the asset. This comment is available as version-specific metadata.

Result

The result does not have any contents.

Example Request

Check in a new version of the asset with the ID 1234 by uploading it with the request. The request needs to use POST method with the asset contents in the request body. The MIME type of the request content can be application/octet-stream.

<http://cip-server:port/cip/asset/checkin/mycatalog/1234>

Example Request Body

The binary contents of the new version of the asset

Example Request

Check in a new version of the asset with the ID 1234 directly from the preconfigured FTP server location “ftpcheckouts.” This location may have been used before to check out the asset.

<http://cip-server:port/cip/asset/checkin/mycatalog/1234?location=ftpcheckouts/image.jpg>

Example Configuration

<location name=’ftpcheckouts’>

ftp://myname:mypassword@ftp.mycompany.com/checkouts

</location>

## /asset/undocheckout

Description

If you have checked out an asset using the /asset/checkout operation you can undo this action by calling this operation.

Path Parameters

<http://cip-server:port/cip/asset/undocheckout/catalog/id?...>

1. catalog

The name of a catalog alias definition from the configuration file. See the configuration section for details on how to define catalog aliases.

1. id

The ID of the asset in the catalog specified by the first path parameter.

Named Parameters

None

Result

The result does not have any contents.

Example Request

Undo the checkout operation performed for the asset with the ID 1234 in the catalog whose alias is “mycatalog.”

<http://cip-server:port/cip/asset/undocheckout/mycatalog/1234>

# Developer Service