documentation for

reynaers product service

concerning

usage guide

Contact informatie

Ordina werkt mee aan dit project in opdracht van ARTOOS.

Alle communicatie verloopt via ARTOOS met onderstaande contact gegevens:

|  |  |
| --- | --- |
| Contact details for price quote | |
| Company name | Artoos |
| Contact | Yoeri Conickx |
| Function | Business Development Manager Marcom Factory |
| Address | Oudestraat 19 |
| Postcode and town: | 910 Kampenhout |
| Telephone | +32 (0)16 618 343 |
| E-mail | y.conickx@artoos.be |

**Intellectual Property Rights and Copyright Statement**

Copyright © Ordina Belgium, 2012. All Rights Reserved.

This document contains materials, the copyright and other intellectual property rights in which are vested in ORDINA or which appear with the consent of the copyright owner. These materials are made available for Reynaers employees to review and to copy for companies own purposes. All other commercial use is prohibited. It is not permitted to view, download, modify, copy, distribute, transmit, store, reproduce or otherwise use, publish, licence, transfer, sell or create derivative works (in whatever format) from this document or any information obtained from this document otherwise than for personal academic or other non-commercial purposes. All copyright and other proprietary notices contained in the original material must be retained on any copy that you make. All other rights of the copyright owner not expressly dealt with above are reserved.

Contents

[1. Introduction 6](#_Toc452387549)

[1.1. General 6](#_Toc452387550)

[1.2. Target Audience 6](#_Toc452387551)

[1.3. Security 6](#_Toc452387552)

[2. Authentication 7](#_Toc452387553)

[2.1. OAuth 7](#_Toc452387554)

[2.2. Requesting an Access Token 7](#_Toc452387555)

[2.2.1. Request 7](#_Toc452387556)

[2.2.2. Response 8](#_Toc452387557)

[2.3. Refreshing an Access Token 9](#_Toc452387558)

[3. OData 10](#_Toc452387559)

[3.1. Model 10](#_Toc452387560)

[3.1.1. Asset 10](#_Toc452387561)

[3.1.2. AssetCategory 10](#_Toc452387562)

[3.1.3. MetaField 11](#_Toc452387563)

[3.1.4. AssetFile 11](#_Toc452387564)

[3.1.5. AssetLink 11](#_Toc452387565)

[3.2. Query Options 11](#_Toc452387566)

[4. Consuming the Service 12](#_Toc452387567)

[4.1. Authorization 12](#_Toc452387568)

[4.2. Categories 12](#_Toc452387569)

[4.2.1. Querying Categories 12](#_Toc452387570)

[4.2.2. Category Previews 13](#_Toc452387571)

[4.3. Assets 13](#_Toc452387572)

[4.3.1. Querying Assets 13](#_Toc452387573)

[4.3.2. Expanding assets 13](#_Toc452387574)

[4.3.3. Asset Previews 14](#_Toc452387575)

[4.4. Sorting and Paging 14](#_Toc452387576)

[4.5. Files 15](#_Toc452387577)

[4.6. Fields 15](#_Toc452387578)

[4.6.1. Field Types 15](#_Toc452387579)

[4.7. Languages 15](#_Toc452387580)

[4.7.1. Get Supported Languages 15](#_Toc452387581)

[4.7.2. Request translated data 15](#_Toc452387582)

[4.8. Synchronization 16](#_Toc452387583)

[4.8.1. Retrieving Changed Assets 16](#_Toc452387584)

[4.8.2. Retrieving Changed Categories 16](#_Toc452387585)

[4.8.3. Retrieving Changed Files 16](#_Toc452387586)

[4.8.4. Determining Removed Resources 16](#_Toc452387587)

# Introduction

## General

This document is intended to guide a developer through the usage of the Reynaers Product Service. We will discuss how to connect, authenticate and submit specific request to the service.

Your actions might differ in detail from the ones described in this document, dependent on the programming language and approach. However, the general idea should be the same.

All URL’s mentioned in this guide are relative URL’s and the document does not contain credentials.

## Target Audience

This document is written for IT professionals who need to consume the Reynaers Product Service.

## Security

The Distributor Information Service requires authentication, please make sure you have a valid app id and app secret and make sure you have a method of acquiring username and password information before moving any further in this guide.

# Authentication

## OAuth

The web service implements OAuth RFC 6749. The standard is extensively documented by the Internet Engineering Task Force (IETF) and their documentation can be found online via: <https://tools.ietf.org/html/rfc6749>

One of OAuth’s strong suits is that it does not require storing credentials. Instead, a token can be requested by submitting the username / password credentials. After the tokens have been acquired, they can be used to authenticate new sessions and the username / password credentials can be discarded.

Access tokens are valid for 1 hour, refresh tokens are valid for 2 weeks. A refresh token can only be used once, after using it, you will receive a new refresh token.

When the refresh token expires a username and password driven authentication is required to attain new tokens.

## Requesting an Access Token

An authentication request is created by submitting a post message to the authentication controller. The controller is located at: /oauth/token

### Request

Requesting access tokens is documented in [section 4.3.2 of the RFC 6749 specification](https://tools.ietf.org/html/rfc6749#section-4.3.2). We will repeat the essentials here. To obtain an access token the following request must be created:

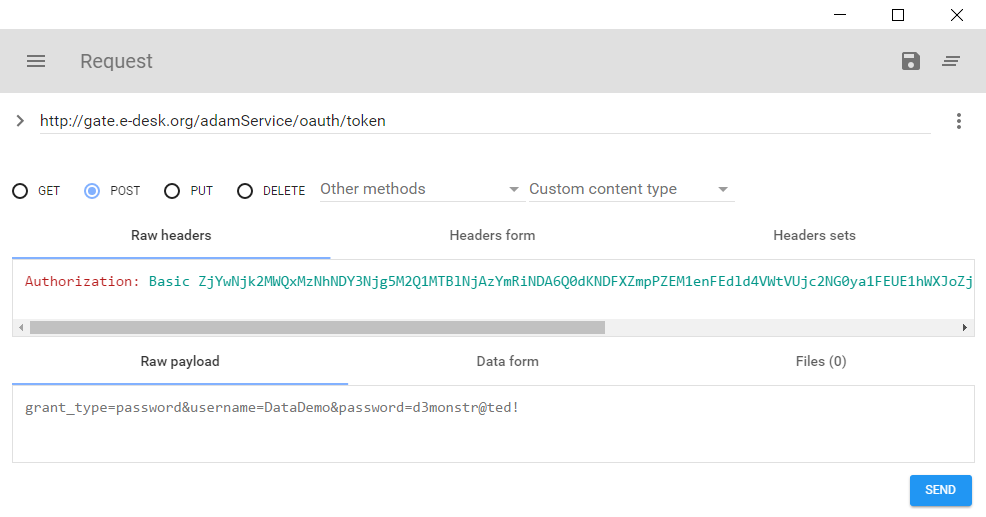
POST /token HTTP/1.1

Host: /oauth/token

Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW

Content-Type: application/x-www-form-urlencoded

grant\_type=password&username=johndoe&password=A3ddj3w



**A closer look at this request:**

POST /token HTTP/1.1

POST is the verb that should be used to create the request.

Host: /oauth/token

The URL too which should be posted.

Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW

The basic authorization header. It contains a base64 encoded concatenation of the app id and app secret, separated by a colon ( <app id>:<app secret> ).

Content-Type: application/x-www-form-urlencoded

Content type header for this request.

grant\_type=password&username=johndoe&password=A3ddj3w

Authentication form data which is posted:

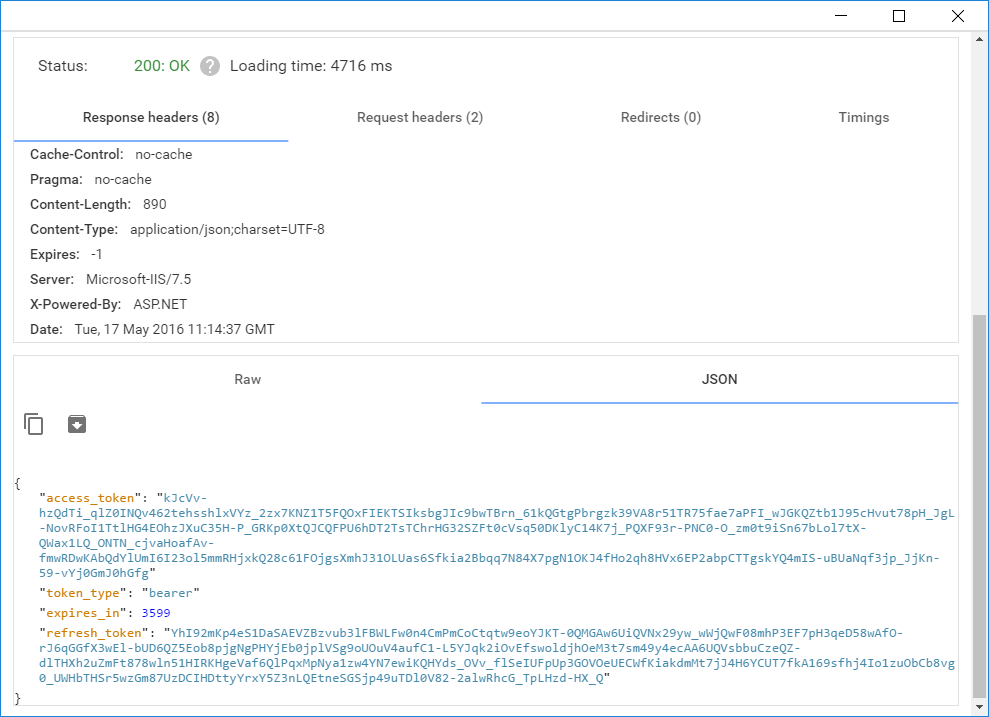
* Grant\_type: set to ‘password’ for access token requests.
* Username / password: user credentials.

### Response

The access token response is documented in [section 4.3.3 of the RFC 6749 specification](https://tools.ietf.org/html/rfc6749#section-4.3.3). We will repeat the essentials here.

A successful response is indicated by status code 200 and contains the following parameters:

* access\_token
* token\_type
* expires\_in
* refresh\_token



## Refreshing an Access Token

Refreshing access tokens is documented in [section 6 of the RFC 6749 specification](https://tools.ietf.org/html/rfc6749#section-6). We will repeat the essentials here. To obtain an access token the following request must be created:

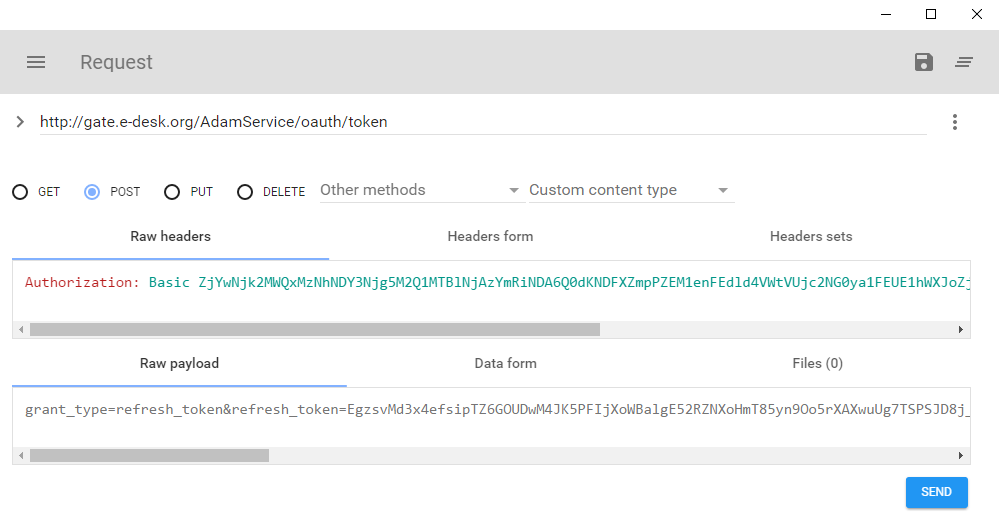
POST /token HTTP/1.1

Host: /oauth/token

Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW

Content-Type: application/x-www-form-urlencoded

grant\_type=refresh\_token&refresh\_token=tGzv3JOkF0XG5Qx2TlKWIA



The response is identical to when requesting an access token. After using the refresh token it expires and the newly provided refresh token must be stored for the next use.

# OData

The service implements OData V4, which is well documented on the [official OData web site](http://www.odata.org/). This document does not repeat the information which can be found in the official documentation (with the exception of a few essentials), but is targeted towards how OData was implemented for Reynaers.

## Model

### Asset

An ‘Asset’ is a dynamic resource which is used to represent any Reynaers product or project. It contains the following properties:

* Id (string): Unique identifier. String representation of a Guid.
* Label (string): Single line, textual representation of the asset in the current language.
* ModifiedOn (DateTime): Date and time in UTC on which the asset was last modified. This value is language dependent.
* CreatedOn (DateTime): Date and time in UTC on which the asset was created.
* Fields (Collection of MetaField): Metadata fields of the asset in the current language.
* Categories (Collection of AssetCategory): Categories in which the asset is classified.
* Files (Collection of AssetFile): Files that are linked to the asset in the current language.
* Links (Collection of AssetLink): Links to other assets in the current language.

### AssetCategory

AssetCategories (also referred to as ‘Categories’) are used to structure assets. These categories give context to an asset. E.G. a window asset will be stored in the ‘Windows’ category.

* Id (string): Unique identifier. String representation of a Guid.
* Label (string): Single line, textual representation of the category in the current language.
* ModifiedOn (DateTime): Date and time in UTC on which the category was last modified.
* CreatedOn (DateTime): Date and time in UTC on which the category was created.
* Parent (string): Unique identifier of the parent category. An empty guid for categories under the root.
* Descendant (string): Only applicable to search, explained in section ‘[Querying Categories](#_Querying_Categories)’ of this document.
* Ancestor (string): Only applicable to search, explained in section ‘[Querying Categories](#_Querying_Categories)’ of this document.

### MetaField

Metafields describe metadata properties for an asset. They are based on field definitions which have a defined data type, label and in some cases pre-defined values.

* Id (string): Unique identifier of the field definition. String representation of a Guid.
* Label (string): Label of the field definition in the current language.
* Value (string): Value of the metafield for the current asset in the current language.

### AssetFile

AssetFile (also referred to as ‘Files’) give a visual representation to an Asset. Each asset can contain multiple files and they are language dependent. Files can be used by multiple assets.

* Id (string): Unique identifier of the file. String representation of a Guid.
* FileName (string): File name and extension.
* ModifiedOn (DateTime): Date and time in UTC on which the file was last modified.
* CreatedOn (DateTime): Date and time in UTC on which the file was created.
* Format (FileType): The content type for this file, possibilities are:
  + Image
  + Video
  + Document
  + Audio
  + Other
* PreviewFormats (Collection of string): Supported preview formats for this file, possibilities are:
  + P: Image preview
  + T: Image thumbnail
  + V: Video preview
  + D: Document preview
  + A: Audio preview

### AssetLink

Asset links describe links to other assets. They are based on link definitions.

* Id (string): Unique identifier of the link definition. String representation of a Guid.
* Label (string): Label of the link definition in the current language.
* Items (Collection of string): Collection of asset identifiers to which the current asset is linked in the current language.

## Query Options

OData supports several query options, more info on query options can be found in the [official documentation](http://docs.oasis-open.org/odata/odata/v4.0/errata02/os/complete/part2-url-conventions/odata-v4.0-errata02-os-part2-url-conventions-complete.html#_Toc406398092). We will repeat the essentials here. A query option is always preceded by ‘$’ and included in lowercase.

* $filter: Queries the collection.
* $expand: Expands a sub collection of the result.
* $top: Returns the top X results.
* $skip: Skips the number of requested results.
* $orderby: Orders the collection by a given property.

# Consuming the Service

This section shows how to use the service. To improve readability we did not encode any of the example URL’s.

## Authorization

After obtaining an access token, the token must be included in every request and is used to determine authorization. To include the token simply add an authorization header containing the access token and token type:

Authorization: <token\_type> <access\_token>

E.G.: Authorization: Bearer kxdQP64fX5V-kfLqLjf3audFzLAJ35Mv8UDZpSj8s83==

## Categories

Categories are an important aspect of our model. They structure assets into a hierarchy and allow us to identify specific object types. It is important to know that a category name is only unique on its current level, which means there can be multiple categories with the same name.

### Querying Categories

This section lists queries that are commonly used when retrieving categories.

Browse Categories (by parent)

Navigating through the structure is facilitated by the ‘parent’ property, which contains the id of the parent category. As mentioned before the root is represented by an empty Guid. The following query retrieves all categories under the root category.

**/odata/categories?$filter=Parent eq '00000000-0000-0000-0000-000000000000'**

Find Modified Categories

Finding modified categories can be achieved via the ‘ModifiedOn’ property. It contains the UTC date on which the category was last modified.

**/odata/categories?$filter=ModifiedOn gt 2016-05-12T11:28:14.077Z**

**Find all child categories (recursive)**

Finding a categories’ children and all children for every result can be achieved via the ‘Ancestor’ property.

**/odata/categories?$filter=Ancestor eq '00000000-0000-0000-0000-000000000000'**

**Find all parent categories (recursive)**

Finding a categories’ parent and all parents for every result can be achieved via the ‘Descendant’ property.

**/odata/categories?$filter=Descendant eq 'ed382dd1-cfeb-4e12-871e-a20700bf896d'**

**Get category detail**

The Id of a category can be used to get its details. This URL requests details for a category with id 'ed382dd1-cfeb-4e12-871e-a20700bf896d'

**/odata/categories('ed382dd1-cfeb-4e12-871e-a20700bf896d')**

**Sort category search result**

Sorting can be easily achieved by adding the OData sort parameter. The following query results in all assets modified after 2016/05/12 and the result are sorted by modification date (newest first).

**/odata/categories?$filter=ModifiedOn gt 2016-05-12T11:28:14.077Z&$orderby=ModifiedOn desc**

### Category Previews

Every category has a JPG preview which can be requested via its detail URL, followed by /media

**/odata/categories('ed382dd1-cfeb-4e12-871e-a20700bf896d')/media**

## Assets

Assets are the corner stone of the model, they are used to represent any Reynaers asset.

### Querying Assets

This section lists queries that are commonly used when retrieving assets.

Find modified assets

Finding modified assets can be achieved via the ‘ModifiedOn’ property. It contains the UTC date on which the asset was last modified in the current language.

**/odata/assets?$filter=ModifiedOn gt 2016-05-12T11:28:14.077Z**

**Get asset detail**

The Id of an asset can be used to get its details. This URL requests details for an asset with id 'ed382dd1-cfeb-4e12-871e-a20700bf896d'

**/odata/assets('ed382dd1-cfeb-4e12-871e-a20700bf896d')**

**Get asset detail (multiple)**

Multiple assets can be requested by using their id. This URL requests details for assets with id '33819104-8b8b-45c6-b551-a1ba00e9b5a5' and 'f2a4e79e-962d-4298-b061-a1ba00e8dce0'. This extension also supports expanding.

**/odata/assets/extensions.getById(ids=['33819104-8b8b-45c6-b551-a1ba00e9b5a5','f2a4e79e-962d-4298-b061-a1ba00e8dce0'])**

**Get assets in category (recursive)**

This query is recursive, which means that it does not only retrieve the assets in the given category, but also in all categories below.

The following query retrieves all assets in a category with id 'ed382dd1-cfeb-4e12-871e-a20700bf896d'. It is always more performant to search by Id and as category names are not unique, they cannot be reliably used for queries.

**/odata/assets?$filter=Categories/any(c:c/Id eq 'ed382dd1-cfeb-4e12-871e-a20700bf896d')**

### Expanding assets

By default an asset will not load any relational data, this must be explicitly requested by using the OData ‘expand’ keyword. Property names are case sensitive.

**Expand the fields of an asset**

**/odata/assets('ed382dd1-cfeb-4e12-871e-a20700bf896d')?$expand=Fields**

**Expand the files of an asset**

**/odata/assets('ed382dd1-cfeb-4e12-871e-a20700bf896d')?$expand=Files**

**Expand files and fields of an asset**

**/odata/assets('ed382dd1-cfeb-4e12-871e-a20700bf896d')?$expand=Fields,Files**

### Asset Previews

An asset’s preview capabilities are dependable on the files it contains. Assets can contain multiple files and every file can support multiple preview formats. An asset also has a default preview.

Comparable to categories, an asset’s preview can be requested via its detail URL, followed by /media. Additionally, specific previews can be requested by adding the ‘q’ parameter, followed by one of the supported previews (as described in ‘[AssetFile](#_AssetFile)’). E.G. /media?q=<PreviewFormat>

Main asset thumbnail

Every asset has a main image thumbnail, regardless of the number of files. The following URL requests that thumbnail for an asset with Id '9be27add-579d-4dac-ad1e-a2d800b1e22a’

**/odata/assets('9be27add-579d-4dac-ad1e-a2d800b1e22a')/media**

Main asset preview

Next to a default thumbnail, every asset also has a default preview image.

**/odata/assets('9be27add-579d-4dac-ad1e-a2d800b1e22a')/media?q=p**

File thumbnail

Most preview requests will be for specific files linked to an asset. Specific files are requested by adding the ‘f’ parameter followed by the file id. The following URL requests the thumbnail for a file with id ‘c072ad8b-1e26-403d-923d-a2d800e0b210’, which is linked to an asset with Id '9be27add-579d-4dac-ad1e-a2d800b1e22a’.

**/odata/assets('9be27add-579d-4dac-ad1e-a2d800b1e22a')/media?f=c072ad8b-1e26-403d-923d-a2d800e0b210**

Specific Preview Format

Specific previews can be requested by adding the ‘q’ parameter, followed by one of the supported previews (as described in ‘[AssetFile](#_AssetFile)’). The following URL requests a video preview for a specific file.

**/odata/assets('9be27add-579d-4dac-ad1e-a2d800b1e22a')/media?f=c072ad8b-1e26-403d-923d-a2d800e0b210&q=v**

## Sorting and Paging

Sorting and paging is default to OData. The service does not enforce server side paging, so the client is free to choose the page size. After the last page has been hit, the service will respond with 0 results. The principles explained here work for every resource type (Assets, Categories, Languages, Files, Fiels)

Sort by modified on date

The following query retrieves all assets in pages of 10 items, requests the third page and is sorted on modification date.

**/odata/assets?$top=10&$skip=20&$orderby=ModifiedOn desc**

## Files

The file controller mainly exists to help determine when a file’s content has been changed, as described in section ‘[Retrieving Changed Files](#_Retrieving_Changed_Files)’ of this document.

Find modified files

Finding modified files can be achieved via the ‘ModifiedOn’ property. It contains the UTC date on which the file was last modified.

**/odata/files?$filter=ModifiedOn gt 2014-05-12T11:28:14.077Z**

**Get file detail**

The Id of a file can be used to get its details. This URL requests details for a file with id 'c072ad8b-1e26-403d-923d-a2d800e0b210'

**/odata/files('c072ad8b-1e26-403d-923d-a2d800e0b210')**

## Fields

All metafields that describe an asset are based on a field definition. This definition determines the Id, label and type of the field. The field resource allows these definitions to be queried. The most commonly use will be to request all fields, which can be achieved with the following query:

**/odata/fields**

### Field Types

Under review

## Languages

The service supports multiple languages. It is important to know that new languages can be created (or even removed) at any time. However, whist this is technically possible, it is expected to be accompanied by a functional change.

By default the service will return data in the language that is configured for the user which was used to authenticate.

### Get Supported Languages

Every language has a unique identifier, a name and a culture code. The following query retrieves an overview of all supported languages

**/odata/languages**

### Request translated data

Any requested data can be translated to a specific language by adding the language header to a request, the value of this parameter is the Id of the language.

**Language header:** X-App-Language

## Synchronization

When the service is used for synchronization we recommend using the ‘ModifiedOn’ property to retrieve changed resources since the last sync action. The asset controller treats the ‘ModifiedOn’ property as a language dependent field. This implies that the result of a ModifiedOn query only retrieves assets that have been changed in the current language.

### Retrieving Changed Assets

We recommend searching directly on the ModifiedOn property, whilst expanding the required relations, but within a limited result set by applying paging. As explained, this query is language dependent. Changing the language of a request is explained in section ‘[Request translated data](#_Request_translated_data)’ of this document.

The following query demonstrates how to request assets as described above:

**/odata/assets?$filter=ModifiedOn gt 2016-05-12T11:28:14.077Z&$top=10&$skip=20&$expand=Fields,Files**

### Retrieving Changed Categories

Retrieving changed categories is very straight forward. There are no relations and the ModifiedOn property is not language dependent. We recommend to use paging because there is no boundary to the number of categories.

The following query demonstrates how to request categories as described above:

**/odata/categories?$filter=ModifiedOn gt 2016-05-12T11:28:14.077Z&$top=10&$skip=20**

### Retrieving Changed Files

When the content of a file changes it is not reflected in the modification date of an asset to which it’s linked. Changing the content of a file has no impact on the asset links in which it is contained, which is why it is not reflected on the modification date of a linked asset. When the collection changes (e.g. when a file is added, removed, or the order has changed), the modification date of the linked asset will change. As always, we recommend to use paging for performance reasons.

The following query demonstrates how to request files as described above:

**/odata/files?$filter=ModifiedOn gt 2014-05-12T11:28:14.077Z&$top=10&$skip=20**

### Determining Removed Resources

The service only returns information on assets and categories within its configured scope. This means that information on resources that were removed from the scope (either by changing meta-data or by deleting the source) cannot be retrieved. This does not impact files; when a file is removed it is also unlinked from any assets, which triggers a modification date change of those assets.

To help subscribers determine which assets and categories have been moved out of scope, the service exposes two extensions which return all known Id’s for the requested resource.

Get all asset ids

**/odata/assets/extensions.getAllIds()**

Get all category ids

**/odata/categories/extensions.getAllIds()**