Search API Documentation

The Search API provides a way to **search for metadata records and media** on the Europeana repository. For example, you would use the Search API to get a response to the query *give me all the results for the word "Vermeer"*. Additionaly, it provides an alternative method using the OpenSearch.RSS protocol for easier integration with external services.

The Search API is the easiest API to use and understand. It interacts with Europeana's data in much the same way as the Europeana website does. You can search for keywords, and the API will return all records that match that keyword. You can refine your search with more advanced filters and advanced query syntax. You can choose to only return objects with certain copyright statements, or you can choose to return the results in a language of your choice. This means that with the Search API, you can get a response to the query: 'Give me all objects by Vermeer that are openly licensed and have high-resolution images.'

Before starting to use this API, we recommend reading the overview of the Europeana Data Model, registering for an API key, and reading the Terms of Use. If you want to get started with this API, go to the Getting Started section or try some calls using our Swagger Console.

Get your API Key here

Get Started

Go to the Console

Need help?

Getting Started

Request

Every call to the Search API is an HTTPS request using the following base URL:

> *https://api.europeana.eu/record/v2/search.json*

On top of this base URL, you need two required parameters to make a successful Search API request: a *query* and an *API key*. to input these required parameters, use “query=” and “wskey=” attached to that URL, using a question mark “?” to separate the parameters from the base URL and an ampersand “&” to separate parameters from each other

```
api.europeana.eu/record/v2/search.json?
query=Vermeer&wskey=yourapikey
```

Below you'll find a table with the other standard parameters you can use in your API Search request:

Search API Request Parameters

Parameter	Datatype	Description
query	String	The search term(s). See Query Syntax for information on forming complex queries and examples.
qf	String	Query Refinement. This parameter can be defined more than once. See Query Syntax page for more information.
reusability	String	Filter by copyright status. Possible values are open, restricted or permission.
media	Boolean	Filter by records where an URL to the full media file is present in the edm:isShownBy or edm:hasView metadata and is resolvable.
thumbnail	Boolean	Filter by records where a thumbnail image has been generated for any of the WebResource media resources (thumbnail available in the edmPreview field).
landingpage	Boolean	Filter by records where the link to the original object on

		the providers website (edm:isShownAt) is present and verified to be working.
colourpalette	String	Filter by images where one of the colours of an image matches the provided colour code. You can provide this parameter multiple times, the search will then do an ‘AND' search on all the provided colours. See: “Colour Palette”
theme	String	Restrict the query over one of the Europeana Thematic Collections. The possible values are: archaeology, art, fashion, industrial, manuscript, map, migration, music, nature, newspaper, photography, sport, ww1.
sort	String	Sorting records in ascending or descending order of search fields. The following fields are supported: score (relevancy of the search result), timestamp_created, timestamp_update, europeana_id, COMPLETENESS, is_fulltext, has_thumbnails, and has_media. Sorting on more than one field is possible by using comma-separated values. It is also possible to randomly order items by using the keyword "random" instead of a field name. You

		can also request for a fixed random order by indicating a seed "random_SEED" which is useful when paginating along the same randomized order. **Use:** field_name+sort_order. **Examples:** sort=timestamp_update+desc sort=random+asc sort=random_12345+asc
profile	String	A profile parameter which controls the format and richness of the response.
rows	Number	The number of records to return. Maximum is 100. Defaults to 12. See pagination.
start	Number	The item in the search results to start with when using cursor-based pagination. First item is 1. Defaults to 1.
cursor	String	A cursor mark from where to start the search result set when using deep pagination. Set to * to start cursor-based pagination.
callback	String	Name of a client side callback function, see JSONP.

Example: Search for all openly licensed records with a direct link to the full media file:

Request:

> *https://api.europeana.eu/record/v2/search.json?query=Paris&reusability=open&media=true*

Response

A response from the Search API is always formatted in JSON and will contain fields that present information about the handling of the request, while the concrete information about the record is presented in the "items" field (see Metadata Sets).

Search API Response Parameters

Field	Datatype	Description
apikey	String	the authentication parameter sent out by the client (the wskey parameter)
action	String	the name of the API method that was called
success	Boolean	a boolean (true/false) flag denoting the successful execution of the call
statsDuration	Number	the time (in milliseconds) taken to serve the request
requestNumber	Number	a positive number denoting the number of request by this API key within the last 24 hours
params	Object	The original request parameters. If an invalid request parameter was submitted, this response parameter will contain the default value (see individual calls for the default values). Shown up only if the profile parameter contains "params".

itemsCount	Number	The number of retrieved records
totalResults	Number	The total number of results
nextCursor	String	Encoded string to pass along to the cursor to navigate to the next page in the search result set. See Pagination.
items	Array (Item)	This is a collection of search results. Each item is represented by a summary of the metadata record. The actual content is dependent of the profile parameter.
facets	Array (Facet)	A collection of facets that describe the resultant dataset.
breadcrumbs	Array (Breadcrumb)	A collection of search queries that were applied in this call.

Error Responses

An error occurring during processing of an API method is reported by (1) a relevant HTTP status code, (2) a value of the success field and (3) a meaningful error message in the error field. The following table shows the fields appearing within an error response:

List of error responses

Field	Datatype	Description
apikey	String	The authentication parameter sent out by the client (the wskey parameter)
success	Boolean	A boolean (true/false) flag denoting the successful execution of the call

statsDuration	Number	The time (in milliseconds) taken to serve the request
error	String	If the call was not successful, this fields will contain a detailed text message.

The following kinds of error codes can be returned by the Record API:

List of Error codes the Record API can throw

HTTP Status Code	Description
200	The request was executed successfully.
401	Authentication credentials were missing or authentication failed.
429	The request could be served because the application has reached its usage limit.
500	An error has occorred in the server which has not been properly handled. If you receive this error it means that something has gone really wrong, so please report them to us!

Example: Request to the Search API supplying an invalid (unknown) API key

Request:

> *https://api.europeana.eu/record/v2/search.json?query=*&wskey=test*

Response:

```
{
    "apikey": "test",
    "success": false,
    "error": "Invalid API key"
}
```

Query, Filter, and Faceting Fields

Search Fields outside EDM

In addition to the fields defined in EDM, a handful of other administrative fields can also be used to search.

Search Field	Datatype	Result Field	Description
europeana_id	String	id	The Europeana ID of the record.
timestamp	Date		
timestamp_created	Date	timestamp_created timestamp_created_epoch	The date when record was created (formatted as ISO 8601)
timestamp_update	Date	timestamp_update timestamp_update_epoch	The date when record was last updated (formatted as ISO 8601)
europeana_completeness	Number	europeanaCompleteness	An internal Europeana measure of the completeness of the metadata of the record, based on the availability of mandatory and optional schema fields. It is measured as a number from 1 to 10 and serves as indicator of the metadata quality.
COMPLETENESS	String	completeness	

Language-specific Search Fields

In EDM, most of the properties that accept a Literal may be language tagged, meaning the field has a tag that describes the language of the text using the ISO 639-2 standard. To allow for a language-specific search on such properties, the Search API defines a field for each of the language variations that appear in our repository while keeping the base field with all the values in all language variations. As opposed to the base field which typically has datatype Text (some fields may also be defined as String), the language-specific fields are always of type String to allow for faceting with the complete value (with no tokenization), see “datatypes for search fields” below for more details. If a language-specific field is part of a metadata set, it can also be output in the response (see “Language-Specific Result Fields” under the “Metadata Sets” Heading).

The following table shows the base and language-specific search fields for the dc:creator property:

Search Field	Search Datatype	Result Field
proxy_dc_creator	Text	dcCreator
proxy_dc_creator.*	String	dcCreatorLangAware

Search Fields defined in EDM

EDM defines an extensive list of classes and properties. In the Search API only a subset of these, corresponding to the ones found to be the most commonly used, can be used to search in the repository. These fields are listed in this section.

The ML (ie. multilingual) column of the table below marks the fields that have multilingual variations. To learn more about the type of information that these fields should hold, please refer to the EDM Definition.

Search Field	Search Datatype	Result Field	ML
ore:Proxy			
proxy_dc_contributor	Text	dcContributor	✓
CONTRIBUTOR	String	dcContributor	
proxy_dc_coverage	String		✓
proxy_dc_creator	Text	dcCreator dcCreatorLangAware	✓

proxy_dc_date	String		✓
proxy_dc_description	Text	dcDescription dcDescriptionLangAware	✓
proxy_dc_format	Text		✓
proxy_dc_identifier	String		✓
LANGUAGE	String	dcLanguage	✓
proxy_dc_publisher	Text		✓
proxy_dc_rights	String		✓
proxy_dc_source	String		✓
proxy_dc_subject	Text		✓
proxy_dc_title	Text	dcTitleLangAware	✓
proxy_dc_type	String		✓
proxy_dc_type_search	Text		✓
proxy_dcterms_alternative	String		✓
proxy_dcterms_created	String		✓
proxy_dcterms_hasPart	String	dctermsHasPart	✓
proxy_dcterms_isPartOf	String	dctermsIsPartOf	✓
proxy_dcterms_issued	String		✓
proxy_dcterms_medium	Text		✓
proxy_dcterms_provenance	String		✓
proxy_dcterms_spatial	String	dctermsSpatial	✓
proxy_dcterms_temporal	String		✓
proxy_edm_currentLocation	String		✓
proxy_edm_hasMet	String		✓
proxy_edm_isRelatedTo	String		✓

TYPE	String	type	
YEAR	String	year	✓
ore:Aggregation			
DATA_PROVIDER	String	edmDataProvider	✓
provider_aggregation_edm_hasView	String		
provider_aggregation_edm_intermediateProvider	String		✓
provider_aggregation_edm_isShownAt	String	edmIsShownAt	
provider_aggregation_edm_isShownBy	String	edmIsShownBy	
provider_aggregation_edm_object	String	edmObject	
PROVIDER	String	provider	✓
provider_aggregation_dc_rights	String		✓
RIGHTS	String	rights	✓
UGC	Boolean	ugc	
edm_previewNoDistribute	Boolean	previewNoDistribute	
edm:EuropeanaAggregation			
europeana_collectionName[1]	String	europeanaCollectionName	
edm_datasetName	String	edmDatasetName	
COUNTRY	String	country	✓
europeana_aggregation_edm_language	String	language	✓
edm:WebResource			
edm_webResource	String		

wr_dc_rights	String		✓
wr_dcterms_isReferencedBy	String		✓
wr_edm_isNextInSequence	String		
wr_edm_rights	String		✓
wr_svcs_hasservice	String		✓
cc:License			
wr_cc_license	String		
provider_aggregation_cc_license	String		
provider_aggregation_odrl_inherited_from	String		
wr_cc_odrl_inherited_from	String		
wr_cc_deprecated_on	Date		
provider_aggregation_cc_deprecated_on	Date		
svcs:Service			
svcs_service	String		
sv_dcterms_conformsTo	String		
edm:Agent			
edm_agent	String	edmAgent	
ag_skos_prefLabel	Text	edmAgentLabel	✓
ag_skos_altLabel	Text		✓
ag_foaf_name	String		✓
ag_rdagr2_dateOfBirth	String		✓
ag_rdagr2_dateOfDeath	String		✓
ag_rdagr2_placeOfBirth	Text		✓

ag_rdagr2_placeOfDeath	Text		✓
ag_rdagr2_professionOrOccupation	String		✓
skos:Concept			
skos_concept	String	edmConceptTerm	
cc_skos_prefLabel	String	edmConceptPrefLabel	✓
cc_skos_altLabel	String		✓
edm:Place			
edm_place	String	edmPlace	
pl_wgs84_pos_lat	String	edmPlaceLatitude	
pl_wgs84_pos_long	String	edmPlaceLongitude	
pl_wgs84_pos_alt	String		
pl_skos_prefLabel	Text	edmPlaceLabel	✓
pl_skos_altLabel	Text	edmPlaceAltLabel	✓
edm:TimeSpan			
edm_timespan	String	edmTimespan	
ts_skos_prefLabel	String	edmTimespanLabel	✓
ts_skos_altLabel	String		✓

Notes:

[1] This field has been deprecated with edmDatasetName. This change followed the change in EDM to rename to edm:collectionName to edm:datasetName. We will keep support for edmCollectionName for a grace period, but on January 2018, we will return only edmDatasetName so please update your API client.

Aggregated Fields

Europeana aggregates its data from cultural institutions that can use diverse, fine-grained systems and methodologies. As a result, a link between for example an object and a person may be stored in different specialized fields. To provide simpler views on this data, Europeana has introduced several general Aggregated Fields, such as: title, who, what, when, and where. In these fields, we gather together information from different record fields to make the discovery of objects easier. Title, for example, aggregates data from the dc:title and dcterms:alternative fields which are part of Dublin Core, a popular general standard for describing different types of resources.

List of Aggregated Fields

Field Name	Search Datatype	Fields being Aggregated
title	Text	proxy_dc_title, proxy_dcterms_alternative
subject	Text	proxy_dc_coverage, proxy_dc_subject, proxy_dcterms_spatial, proxy_dcterms_temporal
what	Text	proxy_dc_format, proxy_dc_type, proxy_dc_subject, proxy_dcterms_medium, cc_skos_prefLabel, cc_skos_altLabel
when	Text	proxy_dcterms_created, proxy_dcterms_temporal, proxy_dc_date, ts_skos_prefLabel, ts_skos_altLabel, proxy_edm_year, proxy_dcterms_issued
where	Text	proxy_dcterms_spatial, pl_skos_prefLabel, pl_skos_altLabel
who	Text	proxy_dc_contributor, proxy_dc_creator, ag_skos_prefLabel,

		ag_skos_altLabel, ag_foaf_name
text	Text	provider_aggregation_edm_dataProvider, provider_aggregation_edm_intermediateProvider, provider_aggregation_edm_provider, proxy_dc_contributor, proxy_dc_coverage, proxy_dc_creator, proxy_dc_date, proxy_dc_description, proxy_dc_format, proxy_dc_language, proxy_dc_publisher, proxy_dc_source, proxy_dc_subject, proxy_dc_title, proxy_dc_type, proxy_dcterms_alternative, proxy_dcterms_created, proxy_dcterms_issued, proxy_dcterms_medium, proxy_dcterms_provenance, proxy_dcterms_spatial, proxy_dcterms_temporal, proxy_edm_currentLocation, proxy_edm_type, ag_skos_altLabel, ag_skos_prefLabel, ag_foaf_name, ts_skos_altLabel, ts_skos_prefLabel, pl_skos_altLabel, pl_skos_prefLabel, cc_skos_altLabel, cc_skos_prefLabel, proxy_dc_type_search

Media Search

The Search API allows not only to search on and retrieve metadata added by curators but also offers powerful features based on technical metadata. Technical metadata is metadata which is extracted from media files such as images and videos which are associated with records, such as the width and height of an image. This allows you to search for and filter Europeana records by media information, for instance to only search for records which have extra large images, high-quality audio files, or images that match a particular colour. Besides searching and filtering, faceting is also possible using technical metadata and is part of the default facets provided by the facet profile.

A Europeana metadata record can contain a reference to zero, one or more media files, this means that when a search is made on a technical metadata property or facet (such as image size), a record is returned if one of the media files present in the record match the search query. The following table lists the fields that relate to the metadata extracted from the media resources:

List of fields related to metadata extracted from media resources

Facet Name	Datatype	Media Type	Description
MEDIA	Boolean		To indicate whether an URL to the full media file is present in the edm:isShownBy or edm:hasView metadata and is resolvable.
MIME_TYPE	String		Mime-type of the file, e.g. image/jpeg
IMAGE_SIZE	String	Image	Size in megapixels of an image, values: small (< 0.5MP), medium (0.5-1MP), large (1-4MP) and extra_large (> 4MP)

IMAGE_COLOUR	Boolean	Image	Lists 'true' for colour images. An alias to this facet is IMAGE_COLOR, note that for non-colour images you cannot provide the 'false' value. Use the greyscale-facet instead.
IMAGE_GREYSCALE	Boolean	Image	Lists 'true' for greyscale images. An alias to this facet is IMAGE_GRAYSCALE, note that for colour images you cannot provide the 'false' value. Use the colour-facet instead.
COLOURPALETTE	String	Image	The most dominant colours present in images, expressed in HEX-colour codes. See colour palette.
IMAGE_ASPECTRATIO	String	Image	Portrait or landscape.
VIDEO_HD	Boolean	Video	Lists 'true' for videos that have a resolution higher than 576p.
VIDEO_DURATION	String	Video	Duration of the video, values: short (< 4 minutes), medium (4-20 minutes) and long (> 20 minutes).

SOUND_HQ	Boolean	Sound	Lists 'true' for sound files where the bit depth is 16 or higher or if the file format is a lossless file type (ALAC, FLAC, APE, SHN, WAV, WMA, AIFF & DSD). Note that 'false' does not work for this facet.
SOUND_DURATION	String	Sound	Duration of the sound file, values: very_short (< 30 seconds), short (30 seconds - 3 minutes), medium (3-6 minutes) and long (> 6 minutes).
TEXT_FULLTEXT	Boolean	Text	Lists 'true' for text media types which are searchable, e.g. a PDF with text.

Colour palette

From all records with images, the six most prominent colours are extracted. These colours are then mapped to one of the 120 colours that can be found in the listing here. To search for records where one of the images matches a particular colour you can use the colour palette parameter, you can provide it multiple times. You need to provide a Hex rgb code as value, such as #8A2BE2 or #FFE4C4.

Datatypes for Search Fields

The following datatypes are defined for the search fields used for querying, filtering and faceting.

List of Datatypes that a search field might hold

Datatype	Description

Boolean	A true or false value.
Number	A numeric value, typically with integer precision.
Date	A point in time with millisecond precision. See Section X to learn more on how to query date fields.
String	Values are preserved as they are present in the data, with no additional NLP processing. This datatype is typically more usefull for faceting.
Text	A word tokenized with punctuation filtering and case sensitive value, with additional stemming of words. This datatype is typically usefull for querying and filtering.

Reusability

The possible values of the reusability parameter are shown in the following Table:

Table of Reusability parameter values

Value		Description
open	The records are freely reusable.	
	PDM	Creative Commons - Public Domain Mark http://creativecommons.org/publicdomain/mark/1.0/
	CC0	Creative Commons - Public Domain Dedication
	CC BY	Creative Commons - Attribution
	CC BY-SA	Creative Commons - Attribution ShareAlike
restricted	The records are reusable, but with restrictions.	
	CC BY-NC	Creative Commons - Attribution NonCommercial

	CC BY-NC-SA	Creative Commons - Attribution NonCommercial ShareAlike
	CC BY-NC-ND	Creative Commons - Attribution NoDerivs NonCommercial
	CC BY-ND	Creative Commons - Attribution NoDerivs
	~~OOC-NC~~	Out of Copyright - Non Commercial re-use (RETIRED)
	InC-EDU	Rights Statements - In Copyright - Educational Use Permitted
	NoC-NC	Rights Statements - No Copyright - Non-Commercial Use Only
	NoC-OKLR	Rights Statements - No Copyright - Other Known Legal Restrictions
permission	You can reuse the records only with explicit permission.	
	~~RR-F~~	Rights Reserved - Free Access (RETIRED)
	~~RR-P~~	Rights Reserved - Paid Access (RETIRED) http://www.europeana.eu/rights/rr-p/
	~~RR-R~~	Rights Reserved - Restricted Access (RETIRED) http://www.europeana.eu/rights/rr-r/
	~~Unknown~~	Unknown Copyright Status (RETIRED)
	RS InC	Rights Statements - In Copyright
	RS InC-OW-EU	Rights Statements - In Copyright - EU Orphan Work
	RS CNE	Rights Statements - Copyright not Evaluated

Example: Search only for freely reusable records:

Request:

> *https://api.europeana.eu/record/v2/search.json?query=Paris&reusability=open*

Profiles

A profile typically determines how extensive the response will be, by either dictating the metadata fields that will be present (ie. minimal, standard and rich) or appending additional data elements such as facets or breadcrumbs. Most facets can be combined with the exception of the metadata facets or combined facets such as rich. The following table lists the profiles supported by the API:

List of possible Profile parameter values

Profile	Description
minimal	Returns minimal set of metadata.
standard	Returns a broader set of metadata.
rich	Returns the broadest set of metadata.
facets	Information about Facets is added. For the records the Standard profile is used.
breadcrumbs	information about the query is added in the form of breadcrumbs. Facets are added as well; for the records the Standard profile is used.
params	The header of the response will contain a params key, which lists the requested and default parameters of the API call.
portal	*standard*, *facets*, and *breadcrumb* combined, plus additional fields.

Breadcrumbs

A collection of search queries that were applied to your call.

List of Breadcrumb fields

Field	Datatype	Description
display	String	Human-readable description of the search

param	String	The search parameter name (**query** or **qf**)
value	String	The search parameter value
href	String	The search part of the URL which can be reused in further calls
last	Boolean	Boolean value indicating whether the current breadcrumb is the last one

Metadata Sets

Each item in a search result is represented by a subset of the fields from the corresponding metadata record. The extent of the fields that are present is determined by the Profile chosen.

Result Fields outside EDM

In addition to the fields defined in EDM, a handful of other fields were defined for administrative reasons that are output in the response.

List of Result fields outside EDM

Result Field	Description
guid	A link to the object page on the Europeana portal to be used by client applications.
link	A link to the API object call. This link should be used to retrieve the full metadata object.
title	The main and alternative titles of the item.
score	The relevancy score calculated by the search engine. Depends of the query.
timestamp	?
timestamp_created_epoch	UNIX timestamp of the date when record were created

timestamp_update_epoch	UNIX timestamp of the date when record were last updated
timestamp_created	ISO 8601 format of the date when record were created
timestamp_update	ISO 8601 format of the date when record were last updated

Language-specific Result Fields

The same way as there are separate language-specific fields for searching, there is also a way to distinguish language-specific values for the response. Such fields always end with the suffix "LangAware" and are represented as LangMap. In order to preserve backwards compatibility we have not changed the original fields. This means that fields such as title, description and creator now appear twice in the search response, one with their original field name (dcTitle) and one as a multilingual labelled list (dcTitleLangAware). In the future, we will replace the single-value fields with the correct multilingual ones.

The following table shows the base and language-specific result fields for the dc:creator property:

Result Field	Result Datatype	Search Field
dcCreator	Array (String)	proxy_dc_creator
dcCreatorLangAware	LangMap	proxy_dc_creator.*

Result Fields

The table below lists all the fields that are output by the search divided per profile (metadata set).

Table of all Response Result fields

Result Field	JSON Datatype	Search Field
Minimal Profile		
id	String	europeana_id
link	String	
guid	String	

edmPreview	Array (String)	
edmIsShownBy	Array (String)	provider_aggregation_edm_isShownBy
edmIsShownAt	Array (String)	provider_aggregation_edm_isShownAt
title	Array (String)	title
dcTitleLangAware	LangMap	proxy_dc_title.*
dcDescription	Array (String)	proxy_dc_description
dcDescriptionLangAware	LangMap	proxy_dc_description
dcCreator	Array (String)	proxy_dc_creator
dcCreatorLangAware	LangMap	proxy_dc_creator.*
edmPlaceLatitude	Array (String)	pl_wgs84_pos_lat
edmPlaceLongitude	Array (String)	pl_wgs84_pos_long
type	String	TYPE
year	Array (String)	YEAR
provider	Array (String)	PROVIDER
dataProvider	Array (String)	provider_aggregation_edm_dataProvider
rights	Array (String)	RIGHTS
europeanaCompleteness	Number	COMPLETENESS
score	Number	score
Standard Profile		
previewNoDistribute	Boolean	edm_previewNoDistribute
edmConceptTerm	Array (String)	skos_concept
edmConceptPrefLabel	Array (LangMap)	cc_skos_prefLabel

edmConceptPrefLabelLangAware	LangMap	cc_skos_prefLabel.*
edmConceptBroaderTerm	Array (String)	cc_skos_broader
edmConceptBroaderLabel	Array (LangMap)	cc_skos_broader
edmTimespanLabel	Array (LangMap)	ts_skos_prefLabel
edmTimespanLabelLangAware	LangMap	ts_skos_prefLabel.*
ugc	Array (Boolean)	UGC
completeness	Number	COMPLETENESS
country	Array (String)	COUNTRY
europeanaCollectionName[1]	Array (String)	europeana_collectionName
edmDatasetName	Array (String)	edm_datasetName
edmPlaceAltLabel	???	pl_skos_altLabel
edmPlaceAltLabelLangAware	LangMap	pl_skos_altLabel.*
dcLanguage	Array (String)	proxy_dc_language
dctermsIsPartOf	Array (String)	proxy_dcterms_isPartOf
timestamp	Number	timestamp
timestampCreated	String	timestamp_created
timestampUpdate	String	timestamp_update
language	Array (String)	LANGUAGE
Portal Profile		
dctermsSpatial	Array (String)	proxy_dcterms_spatial
edmPlace	Array (String)	edm_place
edmTimespan	Array (String)	edm_timespan
edmAgent	Array (String)	edm_agent
edmAgentLabel	Array (LangMap)	ag_skos_prefLabel

dcContributor	Array (String)	proxy_dc_contributor
Rich Profile		
edmLandingPage		europeana_aggregation_edm_landingPage

[1] This field has been deprecated with edmDatasetName. This change followed the change in EDM to rename to edm:collectionName to edm:datasetName. Starting from January 2018, we will return only edmDatasetName . Please update your API client.

JSON Datatypes

The JSON output of this API uses the following datatypes:

List of data types that are used in the JSON Output

Datatype	Description
Boolean	true or false
Number	integer or double precision floating-point number
String	double-quoted Unicode, with backslash escaping
Array	an ordered sequence of values, comma-separated and enclosed in square brackets; the values do not need to be of the same type
Object	an unordered collection of key:value pairs with the ':' character separating the key and the value, comma-separated and enclosed in curly braces; the keys must be strings and should be distinct from each other
LangMap	A special datatype to provide values in various languages. It is an associative array where the keys are ISO language codes or "def" (where the language is not given), and the value is an array of strings. For example: `"dcTitle": {"por":`

	`["Paris"]}`. Here the datatype of dcTitle is a LanguageMap: the language code is "por" (stands for Portuguese), and the value is a list with only one element: "Paris". For those familiar with Java notations: is it the JSON equivalent of `Map<String,List<String>>`

Example: Include the broadest set of metadata in the search response:

Request:

> https://api.europeana.eu/record/v2/search.json?query=Paris&profile=rich

Faceting

The number of records that Europeana contains is very big and growing. Therefore we need efficient ways to allow our users to discover what they need easily. One such technique is a faceted indexing system that classifies each record along multiple dimensions. The facets, seen on the side of europeana.eu, can be useful for filtering search results and can also be used by API clients. If you conduct a search for the keyword "paris" and have a look at the TYPE facet, this facet would tell how many items exist within your search result grouped by TYPE (such as IMAGE, VIDEO etc.). All search fields can also be faceted on.

When you search within your result set for a specific facet, the other items in your facet would still exist (if you search for TYPE:IMAGE, then you can still see how many results there are for TYPE:VIDEO etc.). This last functionality, called multi-facets, is not supported for the Technical Metadata fields.

Requesting Facets

Facets can be requested by either setting the facets or the portal profiles with the profile parameter. By default, a predefined set of facets is returned corresponding to the facets seen on the side of the europeana.eu, which correspond to the following search fields:

- TYPE, LANGUAGE, COMPLETENESS, CONTRIBUTOR, COUNTRY, DATA_PROVIDER, LANGUAGE, PROVIDER, RIGHTS, UGC, YEAR, COLOURPALETTE, MIME_TYPE, REUSABILITY, IMAGE_SIZE, SOUND_DURATION, VIDEO_DURATION, TEXT_FULLTEXT, LANDINGPAGE, MEDIA, THUMBNAIL, IMAGE_ASPECTRATIO, IMAGE_COLOUR, VIDEO_HD, SOUND_HQ

Facet objects in the Response

When requested, facets appear on the response within the facets field as an Array of Facet objects, which are composed by the following fields:

List of Response fields when using the facet profile

Result Field	JSON Datatype	Description
Facet Object		
name	String	The name of the field being facetted, e.g. COUNTRY
fields	Array (Facet Value)	A collection of values for the given facet.
Facet Value		
label	String	The value that was found within the field of one or more objects.
count	Number	The number of objects for which the value was found within that field.

Example: Requesting default facets for all Europeana records

Request:

> [*https://api.europeana.eu/record/v2/search.json?query=*&rows=0&profile=facets*](https://api.europeana.eu/record/v2/search.json?query=*&rows=0&profile=facets)

Response:

```
{
  "apikey": "YOURAPIKEY",
  "success": true,
  "requestNumber": 999,
  "totalResults": 62029238,
  "items": [],
  "facets": [
    {
      "name": "RIGHTS",
      "fields": [
        {
          "label": "http://rightsstatements.org/vocab/InC/1.0/",
          "count": 21135772
        },
        ...
        {
          "label": "http://creativecommons.org/licenses/by-nc-nd/3.0/de/",
          "count": 2732
        }
      ]
    }
    ...
  ]
}

```

Individual Facets

It is also possible to select which facets to retrieve beyond (or instead of) the default facet set, via the `facet` parameter.

Parameter	Datatype	Description
facet	String	A name of an individual field or a comma separated list of fields

The value of the parameter could be "DEFAULT" (which is a shortcut for the default facet set) or any search field. A remainder that search fields with datatype Text are indexed as tokenized terms which imply that facet values and counts will reflect such terms as opposed to the whole value (ie. phrase) like in the remaining datatypes. This is the reason why the language-specific search fields were added with type string so that faceting could be done on the complete values. These are the fields actually used by the Europeana Collections Portal to display the facet values on the side.

We have aligned the logic for faceting across all fields in the API output to be consistent. Previously, faceting on the 'default' facets (such as TYPE, or RIGHTS) would use a different logic than faceting on custom fields (such as proxy_dc_creator). The difference is that now all other values in a list of facet values are returned (multi-facet).

Example: Requesting an individual facet

Request:

> *https://api.europeana.eu/record/v2/search.json?query=*&facet=LANGUAGE&profile=facets&rows=0*

Example: Requesting the default plus an additional individual facet

Request:

> https://api.europeana.eu/record/v2/search.json?query=*&facet=DEFAULT+proxy_dc_rights+proxy_dcterms_medium&profile=facets&rows=0

Multiple Individual Facets

A client can request one or more facets in a single query. This can be done by either duplicating the facet parameter or by combining all the fields needed for faceting as a comma-separated String.

Example: requesting multiple facets by duplicating the facet parameter.

Request:

> *https://api.europeana.eu/record/v2/search.json?query=*&facet=skos_concept&facet=proxy_dcterms_medium&profile=facets&rows=0*

Example: requesting multiple facets using a comma-separated list.

Request:

> *https://api.europeana.eu/record/v2/search.json?query=*&facet=skos_concept,proxy_dcterms_medium&profile=facets&rows=0*

Offset and limit for Facets

A client can request how many facet values to retrieve, and which should be the first one. These parameters can be used to page over all facet values without requesting too many facet values at a time. The table below explains these two parameters. The FACET_NAME constant stands for the field for which the limit applies.

facet offset and limit parameters

Parameter	Datatype	Description
f.[FACET_NAME].facet.limit	Number	Number of values an individual facet should contain. Set a limit of "0" to

		not return anything for that facet. By default, the limit of values of an individual facet is 50. This can be overriden by setting a custom limit e.g. via `&f.DEFAULT.facet.limit=100`.
f.[FACET_NAME].facet.offset	Number	The offset of the first value in an individual facet. The default offset value is "0", starting from the first item in the list while value "1" offsets the list by one, so the first item to return is the second and so on.

Example: Requesting for faceting on the PROVIDER field using offset and limit.

Request:

> *https://api.europeana.eu/record/v2/search.json?query=paris&profile=facets&facet=PROVIDER&f.PROVIDER.facet.offset=10&f.PROVIDER.facet.limit=30&rows=0*

Pagination

The Search API offers two ways of paginating through the result set: basic and cursor-based pagination. The basic pagination is suitable for smaller or user-facing browsing applications which allows for the iteration over the first 1000 results using the start parameter. For larger and/or harvesting applications, the API offers the capability to use cursor-based pagination which allows for a quick iteration over the entire result set.

Pagination	Capabilities	Implementation
Basic	Allows to go to a specific offset/page (start=X).	Use the start parameter to set the search result offset, default value is 1.

	Limited to the first 1000 results (start + rows).	
Cursor-based	Quickly iterate over the entire result set. Does not allow you to go to a specific offset. Cannot be used in conjunction with the start parameter. Based on [Solr Cursor Pagination]().	Set the cursor parameter to * to start cursor-based pagination at page 1. Take the nextCursor value from the response and pass it to the cursor parameter to paginate to the next page (you will need to urlescape the key). When the nextCursor value is not returned anymore, you have reached the end of the search result set.

Query Syntax

Europeana uses the Apache Solr platform to index its data and therefore Apache Lucene Query Syntax is inherently supported by the Search API, although the Solr eDismax query parser is the one currently used by default in the search engine. Advanced users are encouraged to use Lucene and Apache SOLR guides to get the most out of the Europeana repository. For others, we supply a basic guide for querying Europeana.

Basic and phrase search

To look for records that contain a search term in one of the data fields, provide the term as a **query** parameter:

Syntax: "Mona Lisa"

> *[https://api.europeana.eu/record/v2/search.json?query="Mona Lisa"]()*

Note that like in many other search applications omitting the quotes will result in searching for records that contain the term *Mona* and the term *Lisa* but not necessarily both of them together or in that order. We can allow the existence of a number of other words in between by adding that number after the quotes. For example, searching by “Peter Rubens”~1 will return objects about Peter Rubens but also about Peter Paul Rubens.

Search by fields

If you want to limit your search to a specific data field you should provide the name of the field using the following syntax. Use parentheses ( ) to group the keywords to search for in that field. For example, to look for objects whose creator is *Leonardo da Vinci*:

Syntax: who:("Leonardo da Vinci")

> *https://api.europeana.eu/record/v2/search.json?query=who:("Leonardo da Vinci")*

Boolean Search

To combine several terms in one search one can use boolean operators AND, OR, and NOT (note the case-sensitivity). Use parentheses to group logical conditions. Note that two consecutive terms without any boolean operator in between default to the AND operator.

Syntax: mona AND lisa

> *https://api.europeana.eu/record/v2/search.json?query=mona+AND+lisa*

Boolean operators can also be combined with the search by fields. The following example searches for objects whose location is in *Paris* or in *London*:

Syntax: where:(Paris OR London)

> *https://api.europeana.eu/record/v2/search.json?query=where:(Paris+OR+London)*

The boolean NOT operator excludes results that contain the specified word/s after it. For example, looking for objects which contain the term *Lisa* but do not contain the term *Mona* is done by the following:

Syntax: lisa NOT mona

> *https://api.europeana.eu/record/v2/search.json?query=lisa+NOT+mona*

Wildcard search

If you are not sure of the spelling of the search terms, you can use wildcards such as * or ? These will work on all words, but not in the first letter of the word.

- Wildcard - * - will find words with any number of letters in the place of the asterisk, for example ca* will find cat, cap, cane, cable, and canary.
- Wildcard - ? - a single letter wildcard, for example ca?e will find cane, care, case etc.
- You can use the tilde symbol - ~ - to find results with a similar spelling. For example, searching Nicolas~ will also include words Nicholaus, Nicolaas, Nikolaus, Nicola, Nicolai

Syntax: Nicolas~

> *https://api.europeana.eu/record/v2/search.json?query=Nicolas~*

Range search

To execute range queries, the range operator should be used. This example will search for objects whose field values fall between **a** and **z**:

Syntax: [a TO z]

> *https://api.europeana.eu/record/v2/search.json?query=[a TO z]*

As well as for textual fields it can also be used for numeric values, date ranges, or geographical areas, as shown below. Make sure you URLEncode these queries before putting them in a browser, since the square brackets cannot be part of a URL without being encoded first!

Geographical Bounding Box Search

To search for objects by their geographic location you should specify the bounding box of the area. You need to use the range operator and the **pl_wgs84_pos_lat** (latitude position) and **pl_wgs84_pos_long** (longitude position) field. The following example will bring all the objects found between the latitude of 45° and 47° and between the longitude of 7° and 8°:

Syntax: pl_wgs84_pos_lat:[45 TO 47] AND pl_wgs84_pos_long:[7 TO 8]

> *https://api.europeana.eu/record/v2/search.json?query=pl_wgs84_pos_lat:[45 TO 47] AND pl_wgs84_pos_long:[7 TO 8]*

Timestamp Search

One can also search objects by date. Currently, full-fledge date search is supported only for the fields storing the creation (timestamp_created) and update (timestamp_update) dates of the objects in our database, which are available in two formats: the UNIX epoch timestamp and the ISO 8601 formatted date. To search for objects created or updated on a given date, use the following query:

Syntax: timestamp_created:"2013-03-16T20:26:27.168Z"

> *https://api.europeana.eu/record/v2/search.json?query=timestamp_created:"2013-03-16T20:26:27.168Z"*

Syntax: timestamp_update:"2013-03-16T20:26:27.168Z"

> *https://api.europeana.eu/record/v2/search.json?query=timestamp_update:"2013-03-16T20:26:27.168Z"*

Searching for date range (as [date1 TO date2]):

Syntax: timestamp_created:[2013-11-01T00:00:0.000Z TO 2013-12-01T00:00:00.000Z]

> *https://api.europeana.eu/record/v2/search.json?query=timestamp_created:[2013-11-01T00:00:0.000Z TO 2013-12-01T00:00:00.000Z]*

Syntax: timestamp_update:[2013-11-01T00:00:0.000Z TO 2013-12-01T00:00:00.000Z]

> *https://api.europeana.eu/record/v2/search.json?query=timestamp_update:[2013-11-01T00:00:0.000Z TO 2013-12-01T00:00:00.000Z]*

Date mathematics

With date mathematics you can formulate questions such as "in the last two months" or "in the previous week". The basic operations and their symbols are addition (+), substraction (-) and rounding (/). Some examples:

- now = NOW
- tomorrow: NOW+1DAY
- one week before now: NOW-1WEEK
- the start of current hour: /HOUR
- the start of current year: /YEAR

The date units are: YEAR, YEARS, MONTH, MONTHS, DAY, DAYS, DATE, HOUR, HOURS, MINUTE, MINUTES, SECOND, SECONDS, MILLI, MILLIS, MILLISECOND, MILLISECONDS (the plural, singular, and abbreviated forms refer to the same unit).

Let's see how to apply it in Europeana's context.

From xxx up until now

Syntax: timestamp_created:[xxx TO NOW]

> *[https://api.europeana.eu/record/v2/search.json?query=timestamp_created:[2014-05-01T00:00:00.000Z TO NOW]](https://api.europeana.eu/record/v2/search.json?query=timestamp_created:[2014-05-01T00:00:00.000Z TO NOW])*

From xxx up until yesterday

Syntax: timestamp_created:[xxx TO NOW-1DAY]

> *[https://api.europeana.eu/record/v2/search.json?query=timestamp_created:[2014-05-01T00:00:00.000Z TO NOW-1DAY]](https://api.europeana.eu/record/v2/search.json?query=timestamp_created:[2014-05-01T00:00:00.000Z TO NOW-1DAY])*

Changes in the last two months

Syntax: [NOW-2MONTH/DAY TO NOW/DAY]

> *[https://api.europeana.eu/record/v2/search.json?query=timestamp_created:[NOW-2MONTH/DAY TO NOW/DAY]](https://api.europeana.eu/record/v2/search.json?query=timestamp_created:[NOW-2MONTH/DAY TO NOW/DAY])*

You can find more about date mathematics at Solr's API documentation

Query Refinements

So far we have dealt with examples where there was only one query parameter. Sometimes it is useful to split a query into a variable and a constant part. For instance, for an application that accesses only objects located in London, it is possible to have the constant part of the query pre-selecting London-based objects and the variable part selecting objects within this pre-selection.

This can be done using the refinement parameter **qf** which is appended to the request, besides the **query** parameter. This example looks for objects which contain the term *Westminster* and their location is in *London*:

Syntax: query=Westminster & qf=where:London

> *https://api.europeana.eu/record/v2/search.json?query=Westminster&qf=where:London*

Currently, we can also filter the results by distance using the function *distance* in the parameter *qf.* This example will look for objects with the words *world war* that are located (the object itself or the spatial topic of the resource) in a distance of 200 km to the point with latitude 47 and longitude 12.

Syntax: query=world+war & qf=distance(location,47,12,200)

> *https://api.europeana.eu/record/v2/search.json?query=world+war&qf=distance(location,47,12,200)*

We can also use more specific fields instead of location: currentLocation (with coordinates from edm:currentLocation), and coverageLocation (with coordinates from dcterms:spatial and dc:coverage). For example, *qf=distance(currentLocation,47,12,200)* will filter the results to those actually located within 200 km of the coordinates indicated.

Sorting

The search results are, by default, ranked by relevance according to their similarity with the contents of the query parameter. It is possible however to use the parameter ***sort*** to arrange them according to one or more fields, in ascending or descending order. This example looks for objects containing the words *mona* and *lisa*, but sort them according to the field YEAR in ascending order:

Syntax: query=mona+lisa & sort=YEAR+asc

> *https://api.europeana.eu/record/v2/search.json?query=mona+lisa&sort=YEAR+asc*

When we refine by distance (i.e., qf=distance(...)), we can also include *distance+asc* or *distance+desc* in the sorting parameter in order to rank the results by the distance to the coordinates.

Syntax: query=world+war & qf=distance(location,47,12,200) & sort=distance+asc

> *https://api.europeana.eu/record/v2/search.json?query=world+war&qf=distance(location,47,12,200)&sort=distance+asc*

Refinement and sorting parameters can be concatenated. Each such parameter and the mandatory query parameter contributes a breadcrumb object if breadcrumbs are specified in the search profile.

Open Search

Basic search function following the OpenSearch specification, returning the results in XML (RSS) format. This method does not support facet search or profiles. The names of parameters are different from other API call methods, because they match the OpenSearch standard. The OpenSearch response elements can be used by search engines to augment existing XML formats with search-related metadata. The signature of the method is as follows:

> *https://api.europeana.eu/record/opensearch.rss?searchTerms=TERMS&count=COUNT&startIndex=START*

The following parameters are supported by this method:

List of OpenSearch parameters

Parameter	Datatype	Description
searchTerms	String	The search terms used to query the Europeana repository, similar to the query parameter in the search method.
count	Number	The number of search results to return; possible values can be any integer up to 100 [default = 12].
startIndex	Number	The first object in the search result set to start with (first item = 1), e.g., if a result set is made up of 100 objects, you can set the first returned object to the specific object in the set [default = 1].

For the response, see OpenSearch specification.

Libraries and Plugins

Apart from the console, there is a multitude of other ways you can interact with the API. On the libraries and plugins page, you can find libraries that allow you to develop applications with the API in your programming language of choice. Plugins make it easy to integrate the Europeana API into existing applications, such as Wordpress or Google Docs.

Libraries and Plugins - Europeana Knowledge Base - Confluence

Interacting with the Europeana REST API is possible in a multitude of ways, the easiest of which is probably a Console. If you want to use your coding language of choice to interact with an API, there are a few different code libraries that can help you access and use the API in exactly the way you want. Below is a list of libraries and plugins for the Europeana API, sorted by...

pro.europeana.eu

Deprecation Information

The following will be deprecated per the given date, ensure that your API clients are updated accordingly:

Date	Deprecation Details
January 2018	As the API supports HTTPS now for a while, we will start to redirect all non-HTTPS traffic for the API to HTTPS. Ensure your applications follow redirects if needed or adjust the hostname to use HTTPS.

Roadmap and Changelog

We deploy new versions of this API quite regularly, but not all new versions result in changes in the interface. To see the changes made for this version and also all previous releases, see the API changelog in the project GitHub.