Technical Report, No. 402734, May 2019
Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo"

The OpenAIRE Research Graph: Third-party Publishing APIs

Claudio Atzori, Miriam Baglioni, Alessia Bardi, Paolo Manghi, Sandro La Bruzzo, Michele De Bonis, Andrea Dell'Amico, Michele Artini, Andrea Mannocci, Enrico Ottonello

`name.surname@isti.cnr.it`

Abstract

This work describes the specification of the OpenAIRE publishing APIs that support third-party services at publishing metadata about interlinked and packaged research products into the OpenAIRE Research Graph, in respect of the OpenAIRE interoperability guidelines (https://guidelines.openaire.eu). Research products generated by researchers using services of research infrastructures are today manually published by researchers in a repository external to their research infrastructure. This phase is often considered an extra burden, because researchers have to fill in metadata forms with information that is already available in the scope of the services they used. By using the OpenAIRE publishing APIs, services of research infrastructures can implement an on-demand publishing workflow for any type of research products to support their researchers at improving the FAIRness of their research products and relief them from the tedious step of finding a suitable repository and manually depositing the products in it.

Keywords: OpenAIRE, APIs, publishing, Open Science

1. Introduction

The OpenAIRE publishing APIs supports the concept of "continuous publishing" in digital research settings where researchers conduct their activities in digital laboratories using ICT tools and services for processing and analyzing research data. By using the OpenAIRE publishing APIs, a service/tool can automatically publish metadata on behalf of the researchers.
The service/tool and its underlying infrastructure is responsible for keeping persistent identifiers, preserving the payload of the objects and the metadata.
If the service does not want to keep the above responsibilities, then using the Zenodo API is preferred. By depositing an object into Zenodo, the object will get a DOI and its metadata and payload will be preserved according to the policies available at http://about.zenodo.org/policies/ .
Metadata will also be automatically pushed into the OpenAIRE Research Graph.

When a metadata record is pushed into the OpenAIRE Research Graph, via the Zenodo API or the OpenAIRE Publishing API:

- the metadata record is immediately visible via the OpenAIRE search portal and HTTP APIs;
- the metadata record will be cleaned and de-duplicated in a second stage according to the OpenAIRE content provision workflow described at https://www.openaire.eu/aggregation-and-content-provision-workflows and http://doi.org/10.5281/zenodo.996006 .

Researchers benefit from a service that uses the OpenAIRE publishing APIs in several ways:

- The service will support the generation of metadata to improve the FAIRness of the relative research products;
- Researchers are relieved of the burden of depositing the products they want to publish in a repository external to their digital laboratory;
- Researchers can choose to publish research products at any step of their research activity.

Outline Section 2 describes the architectural specification of the APIs.
Section 3 describes the formats of records, based on the OpenAIRE guidelines, that can be submitted via the API.

2. Architecture

The high-level architecture of the APIs and how they interact with the OpenAIRE infrastructure is summarised in figure 1.
The external service calls the APIs to push one metadata record into the OpenAIRE Research Graph. The API component will store the metadata in a dedicated metadata store and call the already existing Direct Indexing API in order to make the metadata available in the OpenAIRE portal and the Search API.
Records in the metadata store will be daily processed by the OpenAIRE Aggregation system to perform cleaning and harmonization to the OpenAIRE internal format, so that they can be de-duplicated and enriched thanks to the OpenAIRE mining algorithms.

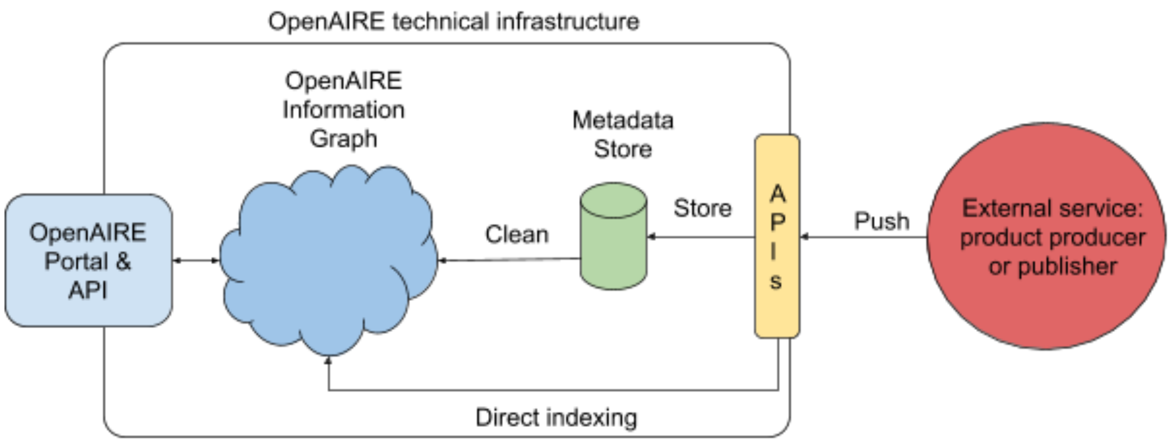
OpenAIRE technical infrastructure
OpenAIRE Information Graph
Metadata Store
OpenAIRE Portal & API
Clean
Store
A P I s
Push
External service: product producer or publisher
Direct indexing

Figure 1. High-level architecture of the OpenAIRE publishing APIs

Details on the API components are shown in figure 2. The APIs provide two main entry points:

- The API key deliverer: a GUI for requesting the API key needed to call the publishing endpoint;
- The publishing endpoint that clients, provided an API key, use to publish metadata into OpenAIRE.

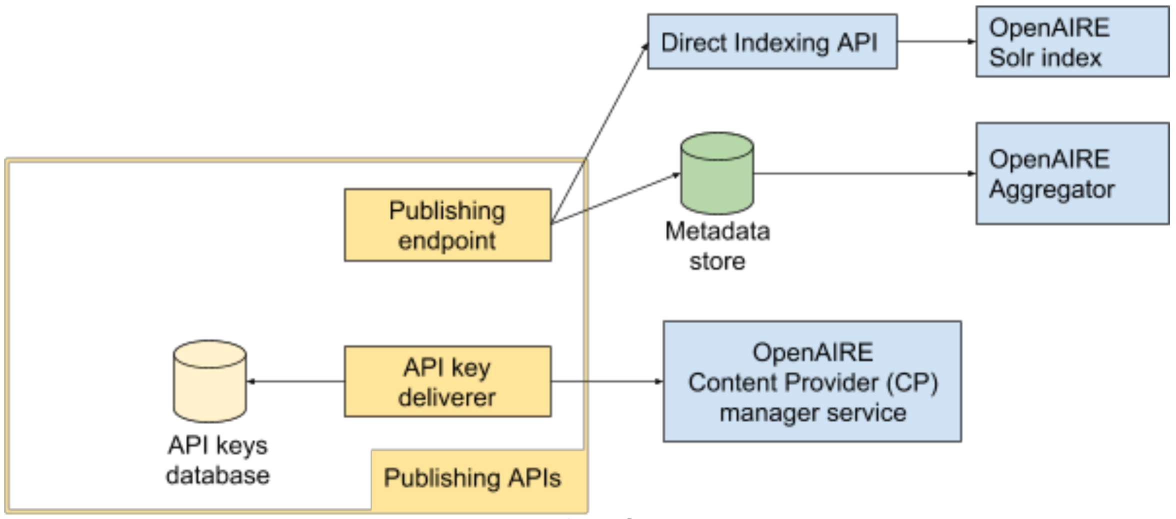

Figure 2. High-level architecture of the OpenAIRE publishing APIs

Figure 3 illustrates the procedure to obtain an API key. The manager of the external service willing to use the OpenAIRE publishing endpoint goes to the Web GUI for token requests and fills in the request form. The form asks the service manager the following information:

- Contact email (mandatory): it will be used by the OpenAIRE Technical team to communicate with the service manager
- Service name (mandatory): it will be used to generate a new OpenAIRE Content Provider. This is needed to track provenance information of the metadata published via the APIs.

- Organization (mandatory): the organization that is responsible for running the service. The service manager will be able to select one of the organization already in OpenAIRE or to add a new one. In case a new organization must be added, the service manager must provide the following information:
  - official/legal name (mandatory)
  - english name (optional)
  - jurisdiction/country (optional, EU will be used by default)
  - web site URL (mandatory)

The API key deliverer component interacts with the Content Provider Manager Service of OpenAIRE to create a new content provider and associate it to the selected (or just created) organisation. An API key associated to the content provider is generated and returned to the service manager.

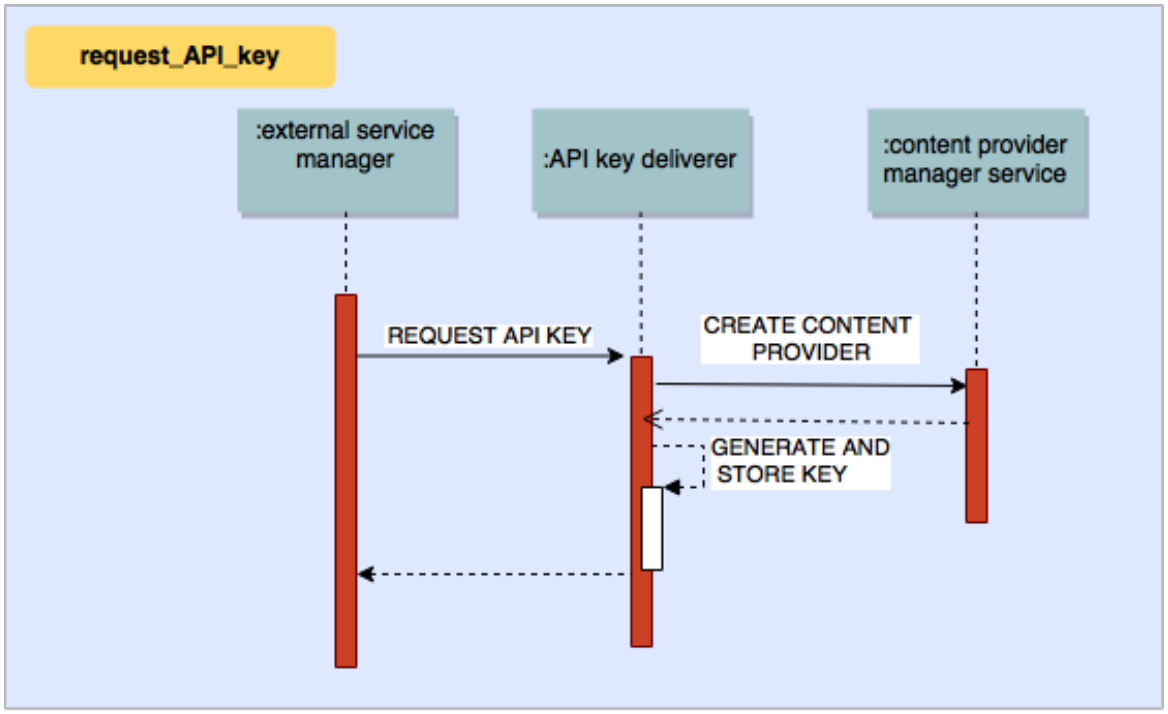

Figure 3 Process to request an API key for the OpenAIRE publishing APIs

Once the API key is obtained, the service manager can configure its service to perform calls to the publishing endpoint providing the API key in the HTTP header.

The publishing endpoint exposes methods to push metadata records in XML format, complaint to the OpenAIRE guidelines. If a record is not compliant to the guidelines, the request will be rejected with an HTTP 4xx Client Error status code.

The sequence diagram in figure 4 describes how the various OpenAIRE components collaborate in case of a successful push request.

The publishing endpoint receives a push request from an authorised service. The request is forwarded to the Publisher component, which validates the input XML record via the XML validator.

If the validation succeeds, as in the case depicted in figure 4, the record is stored in the metadata store and transformed into the format expected by the OpenAIRE Direct Indexing API. The Direct Indexing API transforms again the record into a Solr document and sends it to the OpenAIRE Solr server serving the OpenAIRE production portal and the public search API.

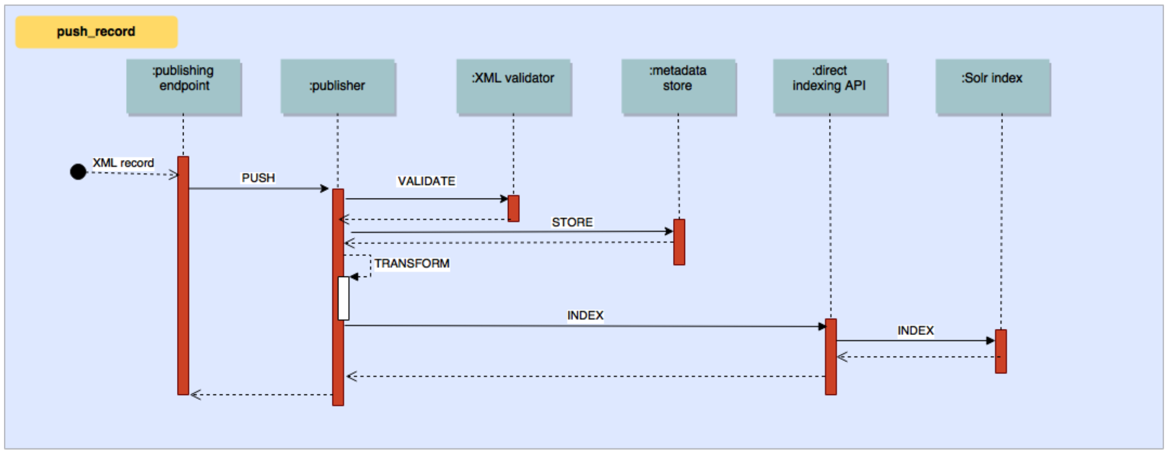

Figure 4. Push request to the OpenAIRE publishing endpoint

2.1 Existing OpenAIRE services used by the publishing APIs

As described in the previous section, the OpenAIRE publishing APIs use existing OpenAIRE internal services to accomplish their task: the Content Provider Manager Service, the Direct Indexing API and the Metadata Store Service. In the following, a brief description of the mentioned services is provided. Note that all these services are not for public use and are accessible only by other OpenAIRE services.

Content Provider Manager Service The Content Provider Manager Service is a service for the management of data sources providing content that is aggregated into the OpenAIRE Research Graph by the OpenAIRE Aggregation infrastructure.
The service supports typical CRUD (Create, Retrieve, Update, Delete) operations via a RESTful interface on the data sources integrated by the OpenAIRE infrastructure. The service is currently used by OpenAIRE Aggregation Infrastructure and the Content Provider Dashboard.

Direct Indexing API The Direct Indexing API has been developed to enable selected components of the OpenAIRE infrastructure (namely Zenodo and the Claim functionality of the OpenAIRE portal) to push metadata records into the Solr index used by the OpenAIRE portal and Search API without waiting for the normal aggregation and content provision workflows to take place. The normal workflow execution that generates updated content for the Solr index takes 1 or 2 weeks and includes several steps of harmonization and enrichment, as explained in details in https://www.openaire.eu/aggregation-and-content-provision-workflows and http://doi.org/10.5281/zenodo.996006. Using the direct indexing API, the harmonization and enrichment steps are not executed, but end-users can view the metadata record on the OpenAIRE portal and the search API just some seconds after their action on Zenodo or on the portal. Records added via the Direct Indexing API that have not been harmonised and enriched are marked as "Records in preview" in the OpenAIRE portal.

Metadata Store A metadata store (mdstore) is a storage unit capable of storing metadata objects in XML format. Metadata stores are managed by the D-Net MDStore Service and are typically fed via metadata aggregation workflows. The OpenAIRE publishing APIs will feed a dedicated mdstore with the validated XML records received from external services. Records will be subject to the OpenAIRE harmonization and enrichment processes like any record collected from "traditional" data sources.

3. Publishing research products into the OpenAIRE Research Graph

Requests to the OpenAIRE publishing API must include the XML metadata record to publish. Its schema varies based on the type of the described resource, according to the OpenAIRE guidelines (http://guidelines.openaire.eu).
The OpenAIRE guidelines are an integrated suite of guidelines for data sources, whose purpose is to set a common and interoperable approach for exporting metadata about research products.

In the following, the guidelines for each type of research product whose metadata can be published via OpenAIRE are briefly described and the following support material is linked to support adopters of the OpenAIRE publishing APIs:

- Guideline documentation;
- XML Schema used by the XML validator component;
- Examples of valid metadata records.

3.1 Publishing literature products¶

Metadata about literature products (e.g. scientific articles, technical reports) can be pushed into the OpenAIRE Research Graph provided they are compliant to the application profile specified by the OpenAIRE guidelines for Literature Repository Managers v4 (https://guidelines.openaire.eu/en/latest/literature_v4/index.html).

The application profile is based on the Dublin Core and DataCite Metadata Schema v4 and COAR vocabularies for access rights (http://vocabularies.coar-repositories.org/documentation/access_rights/) and resource types (http://vocabularies.coar-repositories.org/documentation/resource_types/).

Table 1 lists the fields that are mandatory. Examples of complete and minimal metadata records compliant to the OpenAIRE guidelines for Literature Repository Managers v4 (https://guidelines.openaire.eu/en/latest/literature_v4/index.html) are available at https://github.com/openaire/guidelines-literature-repositories/tree/master/samples.
The XML schema adopted by the XML Validator component is available at https://www.openaire.eu/schema/repo-lit/4.0/. If the submitted record cannot be validated via the schema, the request will be rejected.

Table 1. Mandatory fields in the application profile for literature products

OpenAIRE field	Metadata Element	Note
Title	dc:title	

Creator	datacite:creator	Possibly with identifier, like ORCID
FundingReference	datacite:fundingReference	Mandatory if the literature product is funded by a project
Embargo Period Date	datacite:date (with dateType="Available")	Mandatory if the embargo period hasn't ended yet
Publisher	dc:publisher	Mandatory if available.
Publication date	dc:date	
Publication type	dc:type	Values from the COAR vocabulary of resource types
Description	dc:description	
Resource identifier	dc:identifier	In the form of PID or stable URL. The value will be used by OpenAIRE to uniquely identify the product.
Access rights	dc:rights	Values from the COAR vocabulary of access rights
Full-text location	aire:file	Specify where the full-text of the product can be directly downloaded

3.2 Publishing dataset products

The current OpenAIRE guidelines for Data Archive Managers 2.0 (https://guidelines.openaire.eu/en/latest/data/index.html) are based on the DataCite Metadata Schema v3.1 by making some of the otherwise optional DataCite properties mandatory, as well as enforcing specific encoding schemes on the values of some DataCite properties. A new application profile based on DataCite Metadata Schema v4 is expected to be proposed by the OpenAIRE consortium in the context of the OpenAIRE-Advance project.

For this publishing APIs, it has been agreed not to adopt the current application profile for research data based on version 3.1 Datacite Metadata Schema, but to enhance it to adopt the new features of the Datacite Metadata Schema v4, in a similar way it has already been done for the new OpenAIRE guidelines for Literature Repository Managers v4. Namely, the new features that have been included to generate the application profile and the relative XML schema for the publishing APIs are:

- The adoption of COAR vocabularies for access rights: http://vocabularies.coar-repositories.org/documentation/access_rights/;
- The adoption of COAR vocabularies for resource types: http://vocabularies.coar-repositories.org/documentation/resource_types/;
- Use of the Datacite v4 field *datacite:fundingReference* for the specification of links to projects (instead of using *datacite:contributor* with *contributorType="Funder"*, as in the current guidelines for data repositories)

Table 2 lists the fields that are mandatory. An example of complete XML record compliant to the application profile is available at http://tinyurl.com/yyadfxtd. The XML schema adopted by the XML Validator component is available at https://tinyurl.com/y3s82hg7. If the submitted record cannot be validated via the schema, the request will be rejected.

Table 2. Mandatory fields in the application profile for dataset products

OpenAIRE field	**Metadata Element**	**Note**
Resource identifier	datacite:identifier	In the form of PID or stable URL. The value will be used by OpenAIRE to uniquely identify the product.
Creator	datacite:creator	Possibly with identifier, like ORCID
Title	datacite:title	

Publisher	dc:publisher	Mandatory if available.
Publication year	datacite:publicationYear	In the form YYYY
Date	datacite:date	The type of date must be specified
Access rights	datacite:rights	Values from the COAR vocabulary of access rights
Description	datacite:description	

3.3 Publishing software products

Metadata about research software can be pushed into the OpenAIRE Research Graph provided they are compliant to the OpenAIRE guidelines for Software Repository Managers (https://guidelines.openaire.eu/en/latest/software/index.html). [1] The application profile is based on the DataCite metadata schema v4, the OpenMinTed SHARE-OMTD software guidelines, DOE CODE and Codemeta initiatives.

Table 3 summarises the mandatory field defined by the application profile.

Examples of XML records compliant to the application profile is available at http://tinyurl.com/y43elymg. The XML schema adopted by the XML Validator component is available at https://tinyurl.com/y4puu46e. If the submitted record cannot be validated via the schema, the request will be rejected.

Table 3. Mandatory fields in the application profile for software products

OpenAIRE field	**Metadata Element**	**Note**
Resource identifier	datacite:identifier	In the form of PID or stable URL. The value will be used by OpenAIRE to uniquely identify the product.
Creator	datacite:creator	Possibly with identifier, like ORCID
Name	datacite:title	
Software Type	datacite:resourceType	The attribute resourceTypeGeneral must be "Software". The text value of the element can contain the specific type of software (e.g. Data mining, Statistical analysis)
Access rights	datacite:rights	Values from the COAR vocabulary of access rights

3.4 Publishing "other products"

Metadata about other research products can be pushed into the OpenAIRE Research Graph provided they are compliant to the OpenAIRE guidelines for Other Research Products (ORP) Repository Managers (http://guidelines-other-products.readthedocs.io/en/latest/index.html).

The application profile is based on the DataCite metadata schema v4, and the OpenMinTed SHARE-OMTD software guidelines.

[1] OpenAIRE guidelines are always open for consultation and possible changes in the future. We invite developers of external services using the OpenAIRE publishing APIs to also provide their feedback in order to improve the application profile and guidelines and contribute to best practices on sharing, citation and re-use of research products that are not literature, data, nor software.

Table 4 summarises the mandatory field defined by the application profile. Examples of complete XML records compliant to the application profile are available at http://tinyurl.com/y42av55o.
The XML schema adopted by the XML Validator component is available at https://tinyurl.com/y4golxfy.
If the submitted record cannot be validated via the schema, the request will be rejected.

Table 4. Mandatory fields in the application profile for other research products

OpenAIRE field	**Metadata Element**	**Note**
Resource identifier	datacite:identifier	In the form of PID or stable URL. The value will be used by OpenAIRE to uniquely identify the product.
Creator	datacite:creator	Possibly with identifier, like ORCID
Name	datacite:title	
Type	datacite:resourceType	The attribute resourceTypeGeneral must be one of the Datacite terms: "Other", "Workflow", "Service", "Model". The text value of the element can contain the specific type (e.g. protocol, trial, web service)
Access rights	datacite:rights	Values from the COAR vocabulary of access rights

4 Acknowledgments

This work was partially funded by the EC H2020 OpenAIRE-Connect project (grant agreement: 731011).