

D6.3

OJS plugins for the integration of the OpenAIRE Broker

Submission Date
2025.06.30

Version 1.0 – Submitted version

PU	Public	x
SEN	Sensitive	
R-UE/UE-R	EU classified	

Due Date
2025.06.30

Deliverable Title

OJS plugins for the integration of the OpenAIRE Broker

Deliverable No.

D6.3

Lead beneficiary

MU

Contributing WP

WP6

Type

OTHER



HORIZON-INFRA-2022-EOSC-01
Grant Agreement: 101094397

Project Full Title	Creating a Robust Accessible Federated Technology for Open Access
Project Acronym	CRAFT-OA
Project No.	101094397
Start Date	2023.01.01
End Date	2025.12.31
Duration	36 Months
Project Website	https://craft-oa.eu
Authors	Radek Gomola (https://orcid.org/0000-0002-6903-9937), Alessia Bardi (https://orcid.org/0000-0002-1112-1292)
Abstract	The plugin will enable the integration of enrichment from OpenAIRE into the local OJS systems via the OpenAIRE Broker Service.

Version and Revision History

Version	Date	Author/Reviewer/Contributors	Comments
0.1	2025.06.13	Authors: Radek Gomola (MU), Alessia Bardi (CNR) Contributor: Michele Artini(CNR)	Draft
0.2	2025.06.25	Reviewers: Martina Dvořáková (MU), Michal Růžička (MU) Author: Radek Gomola (MU)	First Internal Review
0.3	2025.06.26	Reviewers: Leonidas Pispiringas (OpenAIRE), Alessia Bardi (OpenAIRE)	Internal Review
0.4	2025.06.27	Reviewers: Hanna Varachkina (UGOE), Antti-Jussi Nygård (TSV)	External Review
0.5	2025.06.27	Author: Radek Gomola (MU) Reviewers: Martina Dvořáková (MU)	Final corrections for submission for final formal check
0.6	2025.06.30	Contributors: Theresa Waldmann (UGOE)	Final formal check
1.0	2025.06.30		Submitted version

Disclaimer



CRAFT-OA is funded by the European Union under Grant Agreement no. 101094397. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or European Commission. Neither the European Union nor the granting authority can be held responsible for them.



This deliverable is licensed under a Creative Commons Attribution 4.0 International License.



Funded by
the European Union

AAI	Authentication and Authorization Infrastructure
ACM	Association for Computing Machinery
API	Application Programming Interface
CRIS	Current Research Information System
cURL	Client for URLs
DDC	Dewey Decimal Classification
DOAJ	Directory of Open Access Journals
DOI	Digital Object Identifier
ID	Identifier
ISSN	International Standard Serial Number
JSON	JavaScript Object Notation
JEL	Journal of Economic Literature
OA	Open Access
OAI-PMH	The Open Archives Initiative Protocol for Metadata Harvesting
OJS	Open Journal Systems
OpenAIRE	Open Access Infrastructure for Research in Europe

ORCID	Open Researcher and Contributor ID
PHP	Hypertext Preprocessors
PID	Persistent Identifier
ROR	Research Organization Registry
UI	User Interface
URL	Uniform Resource Locator

Table of Content

1	Executive Summary.....	8
2	Introduction	9
3	The OpenAIRE Broker	10
3.1	Extension of the OpenAIRE Broker	11
3.2	Enrichments/Topics.....	11
3.3	REST API.....	14
4	OJS Plugin: Integration of the OpenAIRE Broker	16
4.1	Plugin availability and installation.....	16
4.1.1	Plugin availability	16
4.1.2	Installation	17
4.2	Status description.....	17
4.2.1	Journal-level Enrichments.....	17
4.2.2	Article-level Enrichments.....	18
4.3	Plugin functionality	19
4.3.1	OAI-PMH Matching Requirements	20
4.3.2	OJS Limitations.....	21
4.3.3	Limitations of the Broker API.....	22
4.3.4	Supported Enrichment Types	22
4.3.5	Validation and Filtering Logic.....	23
4.3.6	Version Compatibility Adjustments	25
4.3.7	Core Architecture and Functional Structure of the Plugin	26
4.4	Possible Future Improvements	28
4.4.1	Performance Improvements by adding Database	28

4.4.2	Adding New Subscriptions Dynamically.....	29
5	Journal's subscription registration in OpenAIRE Provide	32
5.1	Preliminary Step – Preparation	32
5.2	Registration Process	33
5.3	Subscriptions	34
6	Evaluation	37
7	Conclusion.....	38
8	References	39
8.1	List of References	39
8.2	List of Websites	39
9	List of Figures	40
10	List of Tables	42

1 EXECUTIVE SUMMARY

This report presents the development and implementation of a new plugin designed to integrate metadata enrichments from the Open Access Infrastructure for Research in Europe (OpenAIRE) infrastructure into the Open Journal Systems (OJS). The plugin leverages the OpenAIRE Broker Service to deliver value-added metadata directly to journal editors and journal managers.

OpenAIRE aggregates metadata from numerous sources across the research landscape and enhances it through internal deduplication and enrichment processes. These enrichments, available exclusively to data providers through their OpenAIRE Provide accounts, offer insights and metadata extensions based on inferred relationships and aggregated data. However, when a multi-journal system administrator acts as the registered provider, individual journal editors typically do not have access to these enrichments due to cross-journal data sensitivity.

To address this limitation, the plugin was developed to allow editors and journal managers to view enrichment data related only to their own journal, independently of the OpenAIRE Provide account. By connecting to the OpenAIRE Broker Application Programming Interface (API), the plugin fetches and displays available enrichments for published articles within the scope of the respective journal.

The plugin is released as an open-source project and is publicly available via the GitHub repository: <https://github.com/munipress/openAIREBrokerService>

The documentation outlines the fundamental principles of the OpenAIRE Broker API (<https://graph.openaire.eu/docs/apis/broker-api>), including example queries. It also provides a detailed description of the plugin's structure, which is divided into two main components:

- A module for displaying enrichments for individual published articles.
- A module for browsing all enrichments across the journal.

Additionally, a comprehensive list of available enrichment types is provided, including descriptions and contextual explanations. The report also details how subscriptions to specific enrichment topics can be configured, along with limitations currently applicable to the plugin.

The final sections describe practical usage scenarios, including the interaction between the plugin interface and the enrichments, as well as validation results across several journals within the system.

This development significantly improves the accessibility and usability of enriched metadata for journal stakeholders, supporting better editorial decision-making and enhancing metadata quality at the local level.

2 INTRODUCTION

The aim of this deliverable (D6.3) is to provide users of the Open Journal Systems (OJS) platform with a more effective way of accessing Enrichments from the Open Access Infrastructure for Research in Europe (OpenAIRE) through integration with the OpenAIRE Broker Application Programming Interface (API) Service.

The main output of this deliverable is a new plugin for OJS that enables journal editors and managers to view enrichment metadata related to their own journal content without requiring direct access to the OpenAIRE Provide account. This addresses a significant limitation in multi-journal OJS installations, where enrichments are typically available only to the central system administrator through the OpenAIRE Provide account.

Unlike previous OJS plugins from D6.1 focused on OpenAIRE integration, this plugin does not serve for metadata exporting or indexing, but instead facilitates access to enrichment data aggregated and deduplicated by OpenAIRE. The plugin retrieves and displays these enrichments using the OpenAIRE Broker API and provides two primary views: one for article-level enrichments and another for journal-wide enrichment overviews. The plugin does not directly modify or add metadata within the OJS system; instead, it serves a list that displays specific potential enrichments retrieved via the OpenAIRE Broker API.

The structure of the document is as follows:

- **Section 1:** Introduction and structure of the document.
- **Section 2:** Introductory notes on the OpenAIRE Broker API, its environment, and standalone testing options using the Swagger User Interface (UI).
- **Section 3:** Description of the new plugin, its technical functionality, limitations, and components.
- **Section 4:** Journal's subscription registration in OpenAIRE Provide.
- **Section 5:** Description and analysis of enrichment data retrieved through the plugin and their impact on participating journals.
- **Section 6:** The scope for future improvement.
- **Section 7:** Conclusions.

3 THE OPENAIRE BROKER

The OpenAIRE Graph is a scholarly knowledge graph that brings together and interlinks hundreds of millions of metadata records describing resources of the research life-cycle: research products (literature, data, software, other), organisations, project grants, research communities, data sources, persons.

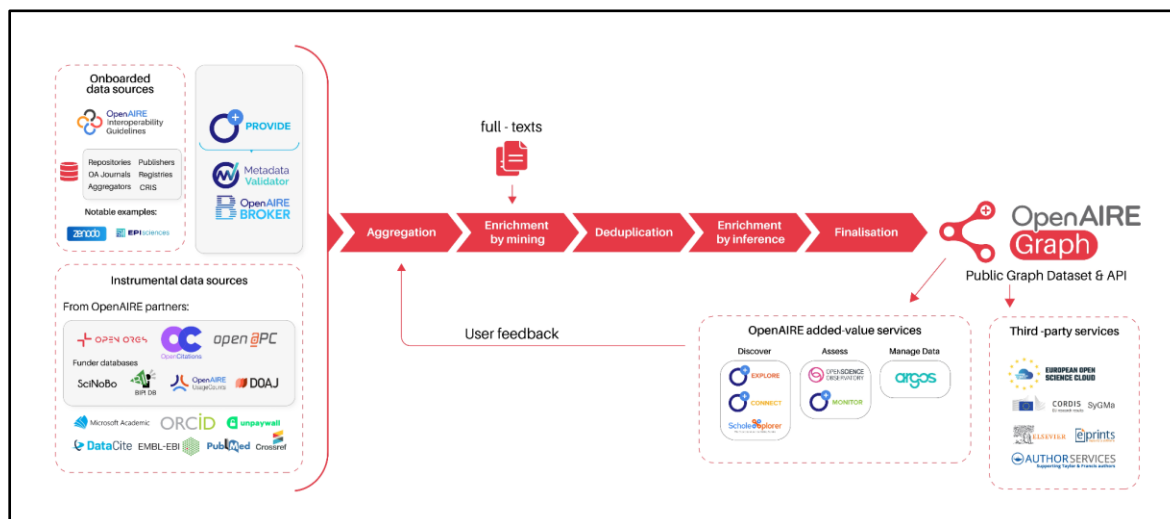


Figure 1: The aggregation-enrichment-deduplication pipeline of the OpenAIRE Graph

Figure 1 shows the pipeline that constructs the OpenAIRE Graph. Scheduled automatic processes collect metadata records from trusted sources of the scholarly communication ecosystem worldwide that are compliant with the OpenAIRE guidelines and registered in OpenAIRE Provide, such as institutional, national, and thematic repositories and aggregators, journals, pre-print services, data archives, Current Research Information System (CRIS) systems. Collected records are added to a data lake, together with metadata records from other relevant scholarly communication sources like Crossref, Datacite, PubMed, ORCID (Open Researcher and Contributor ID), Research Organization Registry (ROR), OpenAPC, OpenCitations and information about projects provided by national and international funders. Full-texts of Open Access (OA) publications are also analysed and used to extract additional information that is not available in the bibliographic metadata like links to funding projects, to datasets, to software and citations. Publications are also classified by Fields of Science and Sustainable Development Goals. Duplicated organisations and research products are identified and merged together to obtain an open, trusted, public resource enabling explorations of the scholarly communication landscape. The enriched graph is eventually made available via the OpenAIRE portals, API, deposited as a metadata snapshot and is further analysed to produce statistics for the OpenAIRE MONITOR, including the Publisher Dashboard developed in T6.2 of the CRAFT-OA project. The OpenAIRE Graph is updated once a month, while the metadata snapshot is deposited on Zenodo twice per year (<https://graph.openaire.eu/docs/category/downloads>).

The enrichments added by OpenAIRE to the records can be returned to the original sources, so that they can enhance their records with information that was not available before.

OpenAIRE facilitates the exchange of added-value information with the OpenAIRE Broker service. A process analyses the OpenAIRE Graph and thanks to the detailed provenance information at the level of the records, fields, and relationships, generates enrichment events that can be notified to the source of the enriched record. The Broker service features a graphical user interface integrated into OpenAIRE Provide and API for programmatic access.

Via OpenAIRE Provide, data source managers (e.g. repository managers, journal platform managers, journal editors) can explore a preview of the events and can decide to subscribe to specific types of event, based on their interest. The list of the supported types of events is detailed in Section 2.2.

3.1 Extension of the OpenAIRE Broker

The OpenAIRE Broker used to generate events only for repositories and CRIS systems. During the CRAFT-OA project, the service was extended to generate events also for journals, publisher's archives. The Broker was also enabled for the Directory of Open Access Journals (DOAJ).

The extension caused an increase in the number of events from 14.7M generated in March 2025 to 20.9M generated in June 2025, with a substantial increase of 42%. The Broker generated events for 93,468 data sources.

This significant spike caused some technical challenges that were addressed by optimizing the process that generates the events and updating the version of Apache Spark (<https://spark.apache.org/>, an engine to perform large scale data analytics) from 2.4 to 3.4.

3.2 Enrichments/Topics

Data source managers can subscribe via OpenAIRE Provide and be notified about:

- Additional Persistent Identifiers (PID) of their publications (e.g. Digital Object Identifiers (DOI)).
- ORCID identifiers that can be associated with an author of a publication.
- Links to OA versions.
- Additional classification subjects (e.g. subjects from standard classification schemes like Association for Computing Machinery (ACM), Journal of Economic Literature (JEL) and Dewey Decimal Classification (DDC)).
- Abstracts found in duplicate publications.
- Missing publication dates.

Topic	Description	#events
ENRICH/MORE/OPENACCESS_VERSION	Another OA version of a publication	5,533,483
ENRICH/MORE/PID	Another PID associated to your publication	4,231,821
ENRICH/MISSING/AUTHOR/ORCID	An ORCID that can be associated to an author of your publications	2,430,747
ENRICH/MISSING/ABSTRACT	An abstract describing among your publications	2,409,492
ENRICH/MISSING/PUBLICATION_DATE	A date of publication missing in your record	764,020
ENRICH/MISSING/PID	A PID associated to your publications	649,161
ENRICH/MISSING/SUBJECT/DDC	A DDC term that can be associated to your publications	227,052
ENRICH/MORE/SUBJECT/DDC	Another DDC term that can be associated to your publications	227,052
ENRICH/MISSING/SUBJECT/JEL	A JEL classification term that can be associated to your publications	132,802
ENRICH/MORE/SUBJECT/JEL	Another JEL classification term that can be associated to your publications	132,802
ENRICH/MISSING/OPENACCESS_VERSION	OA versions of your publications	65,960
ENRICH/MISSING/SUBJECT/ACM	An ACM classification term that can be associated to your publications	4
ENRICH/MORE/SUBJECT/ACM	Another ACM classification term that can be associated to your publications	4

Table 1: Broker event topics

More specifically, enrichments are organized into categories named “topics” that represent the different types of enrichments OpenAIRE can generate. Table 1 provides details on the topics of events generated in June 2025 by the Broker service. OpenAIRE Provide offers to data source managers the possibility to preview a set of enrichments relative to their data source that OpenAIRE can derive from the Graph. For each topic the preview consists of (see figures 2 and 3):

- 100 random “enrichment events”, a subset of all the possible enrichments pertinent to a given data source in the OpenAIRE Graph, that the user can explore by applying filters on different criteria.
- The total number of events that can be generated.

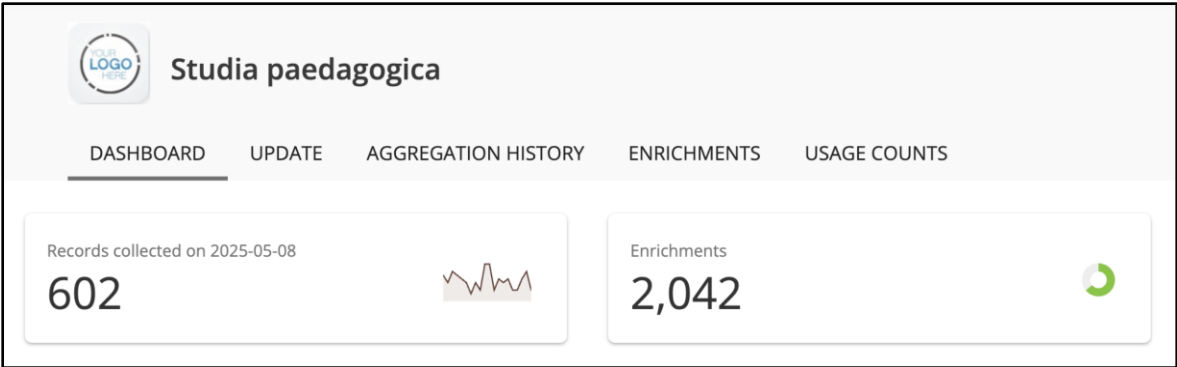


Figure 2: Screenshot of OpenAIRE Provide for the journal “Studia paedagogica”

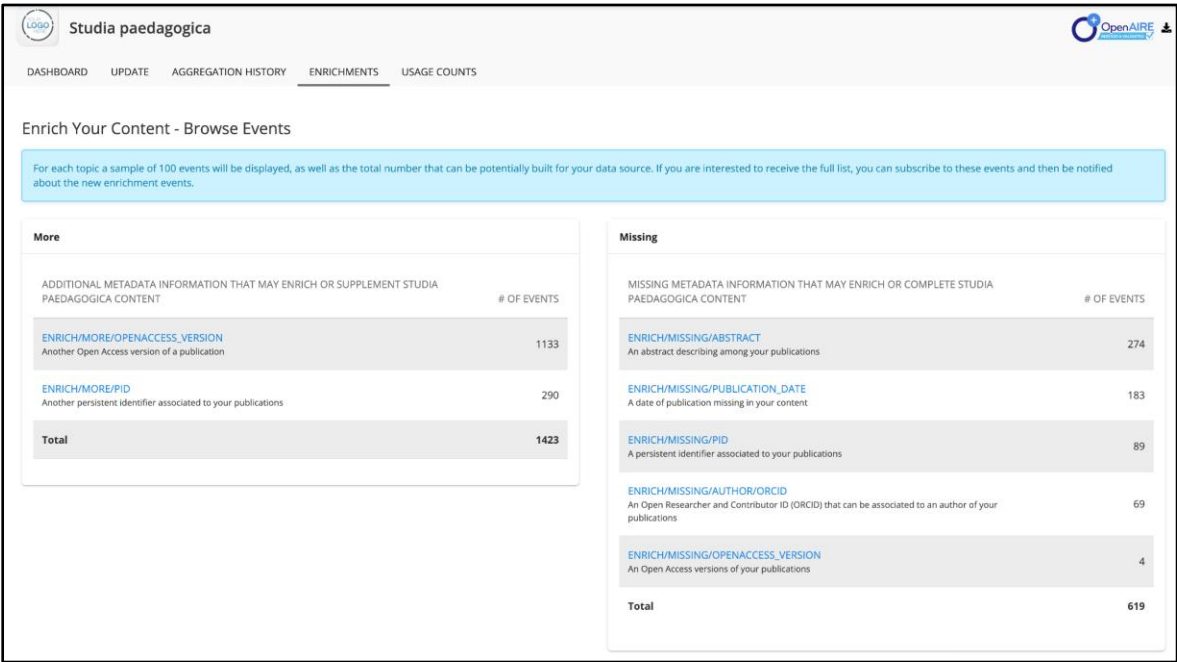


Figure 3: Preview of events generated for the journal “Studia paedagogica”

Data source managers can create subscriptions for specific topics, also including the filtering criteria they used to analyze the enrichments preview. Once the data source manager creates a subscription, the algorithm analyzing the OpenAIRE Graph will produce the full set of enrichments for the manager's source, possibly far beyond the 100 enrichments available in

the preview (see Figure 4). The enrichments are available as notifications in a dedicated section in OpenAIRE Provide to be further checked, as well as through the Broker API for programmatic access. Notifications will be sent to subscribers every time the OpenAIRE Graph is updated and analyzed to derive the enrichments.

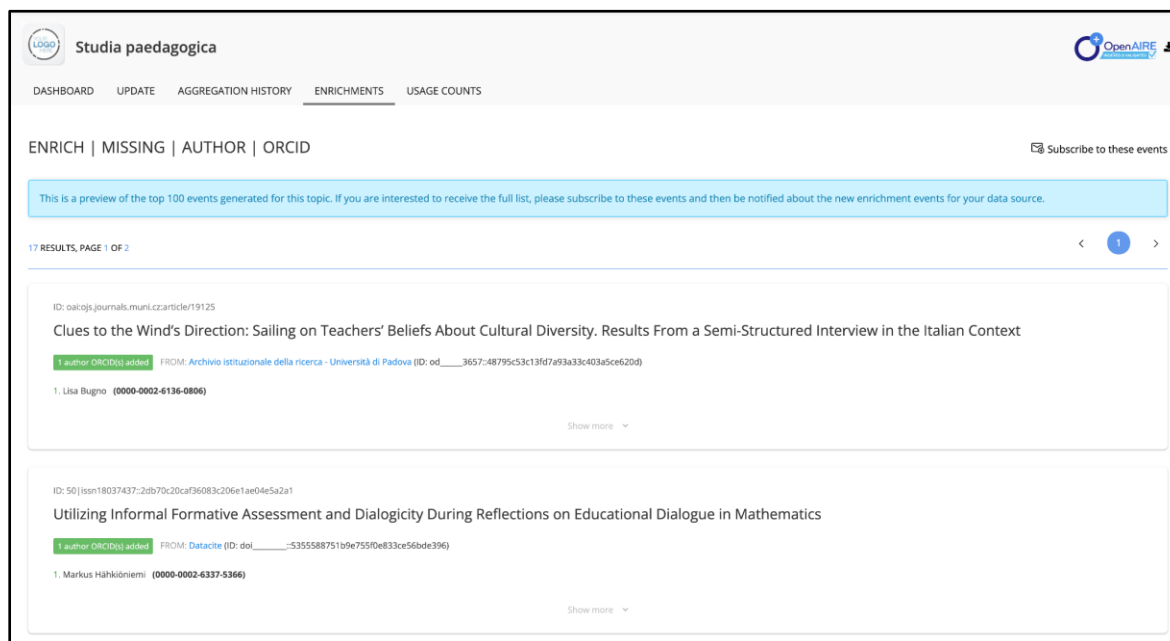


Figure 4: Preview of events about ORCID identifiers of authors

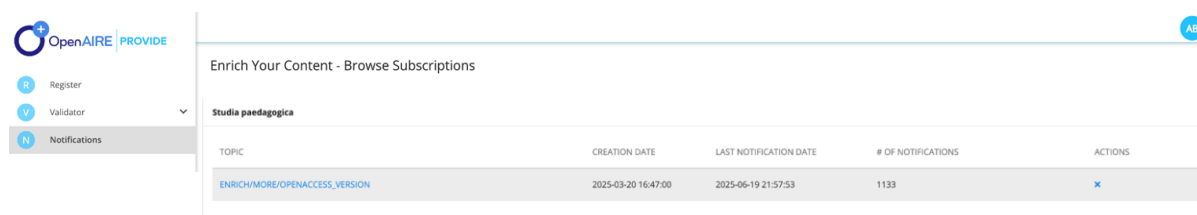


Figure 5: Notifications for the subscription to the topic ENRICH/MORE/OPENACCESS_VERSION

3.3 REST API

The Broker service is also available for programmatic access via API described at <https://graph.openaire.eu/docs/apis/broker-api>. A Swagger endpoint is available for testing at: <https://api.openaire.eu/broker/swagger-ui/index.html>.

The API features the following methods to retrieve events of subscribed topics:

- `/subscriptions`, param: email
Returns the list of subscriptions by user email in JavaScript Object Notation (JSON) format. The response body includes details of each subscription, including the topic, the data source and a subscription identifier (ID) that can be used in other methods to retrieve the enrichment notifications.

- `/scroll/notifications/bySubscriptionId/{subscrId}`
Where `{subscrId}` is a subscription ID. The method returns the first page of the list of notifications in JSON format. The response body includes:
 - `id`: scroll id to use to retrieve the next page of notifications (see next method). Each scroll ID has an expiration time of about 5 minutes.
 - `completed`: boolean that informs if this is the last page of notifications.
 - `values`: an array of events with the following fields:
 - `originalId`: the ID of the record at the source. This field can be used by clients to relate the enrichment to the record in the original data source.
 - `title`: title of the publication.
 - `trust`: a number from 0 to 1. Higher the value, higher the reliability of the enrichment. The trust level depends on the data source that provided the information and/or the accuracy of the algorithm that produced the enrichment.
 - `message`: the actual enrichment to be notified. It is a structured field whose format depends on the specific topic.
- `/scroll/notifications/{scrollId}`
Where `{scrollId}` is the ID returned by a call of the previous method. The method returns a page with a list of notifications in the same format as described above.
- `/file/notifications/bySubscriptionId/{subscrId}`
Where `{subscrId}` is a subscription ID. The method returns a gzipped json file with all the notifications for the given subscription ID. The format of the json is the same as the content of the `values` field described for the scroll method.

4 OJS PLUGIN: INTEGRATION OF THE OPENAIRE BROKER

The core component of this deliverable is the OpenAIRE Broker Service Integration of Enrichments plugin for OJS. The plugin enables journals to display metadata enrichments retrieved from the OpenAIRE Broker Service directly in the OJS interface, without requiring manual interaction with the OpenAIRE Provide platform.

4.1 Plugin availability and installation

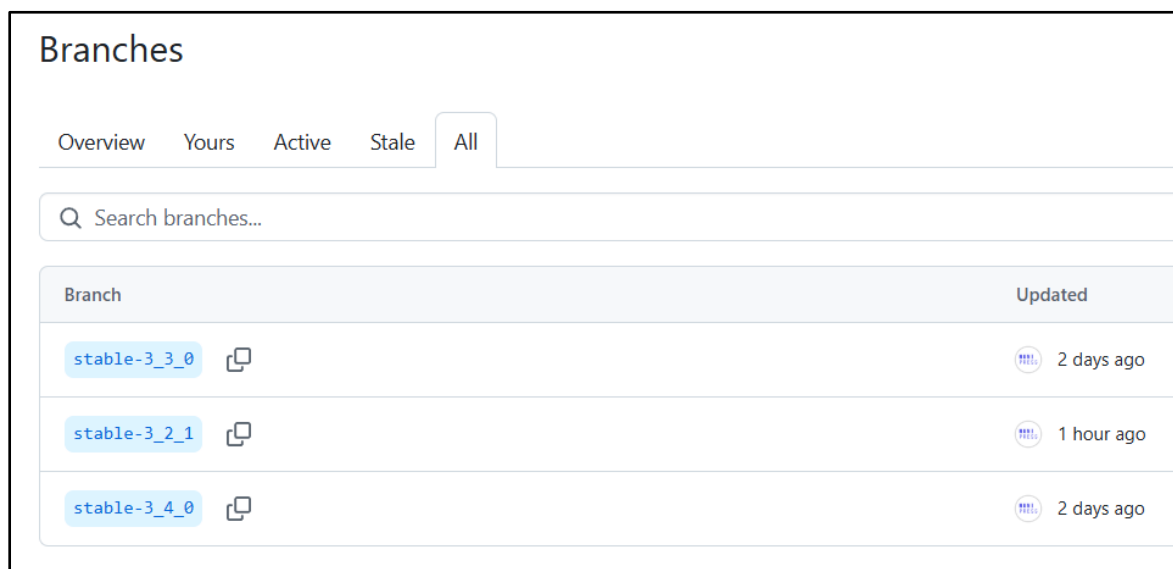
4.1.1 Plugin availability

The plugin is publicly available as an open source project Git repository at GitHub: <https://github.com/munipress/openAIREBrokerService>.

The repository provides access to the full source code and tagged release versions. Three branches are created for the most commonly used versions of OJS 3.2.x, 3.3.x and 3.4.x.

The current plugin maintained versions are:

- 3.2.x – For running on OJS 3.2.x.
- 3.3.x – For running on OJS 3.3.x.
- 3.4.x – For running on OJS 3.4.x.



The screenshot shows the 'Branches' page of a GitHub repository. It includes tabs for Overview, Yours, Active, Stale, and All, with 'All' selected. A search bar is present. Below is a table of branches:

Branch	Updated
stable-3_3_0	2 days ago
stable-3_2_1	1 hour ago
stable-3_4_0	2 days ago

Figure 6: Active branches in GitHub project

4.1.2 Installation

The plugin needs to be downloaded in the version matching the OJS version and added to the plugin set in the *[OJS root folder]/plugins/generic/* folder.

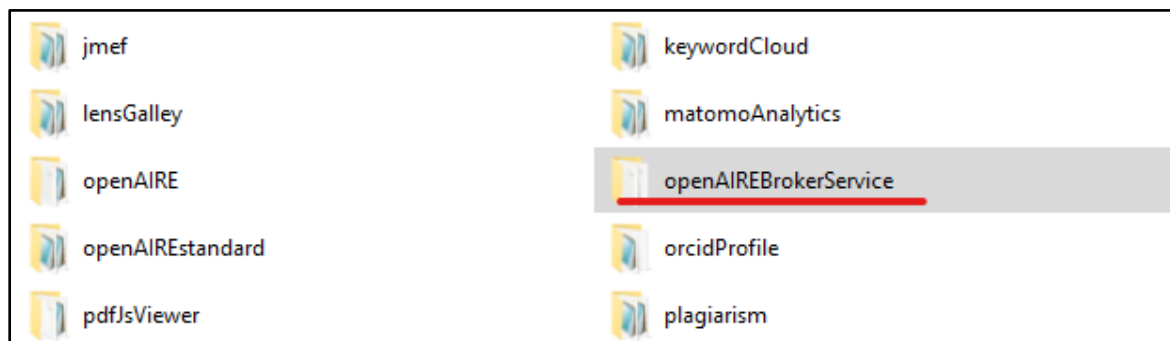


Figure 7: The plugin is listed in the generic folder

The plugin operates without modifying the OJS database schema, as it does not introduce any new database tables. Therefore, the use of the command-line upgrade tool is not required.

After installation, the plugin becomes visible in the OJS plugin gallery and can be enabled via the plugin settings interface under *Settings* → *Website* → *Plugins* → *Generic Plugins* → *OpenAIRE Enrichments*.

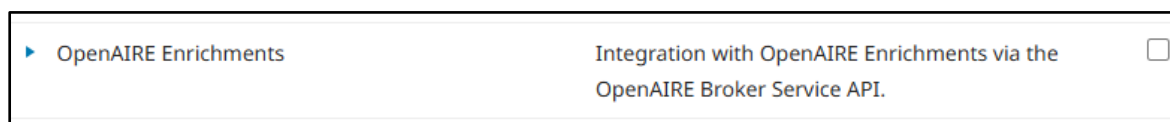


Figure 8: The plugin listed in the List of Plugins

Upon activation, the plugin's configuration interface is made available. This interface enables journal managers to manage and define specific topic-based subscriptions.

4.2 Status description

The plugin is currently fully functional and processes enrichment data on two levels:

4.2.1 Journal-level Enrichments

These represent aggregated enrichment results for the entire journal. They are displayed in a tabular view (grid) accessible via the **Website Settings** section in OJS. This overview allows editors and journal managers to review metadata enrichment suggestions across all published content.

Website Settings

Appearance Setup Plugins Static Pages **OpenAIRE Enrichments**

Depending on the size of your journal (the number of published articles) and the number of registered subscriptions, loading may take several minutes. Be sure you have setted up execution time high enough. If you have troubles contact your system administrator.

ID	Author; Title	Issue	Topic	Message
19087	Hellemans et al.; Benefits of a Small Research Study for the Teacher Education at a University of Applied Sciences: A Case Study	Vol. 22 No. 4 (2017): Studia paedagogica: Teacher Education and Educational Research	MISSING/PUBLICATION_DATE	Trust: 1 publication date: 2017-01-01
19225	Pol; Editorial	Vol. 26 No. 3 (2021): Studia paedagogica	MISSING/PID	Trust: 1 type: doi; value: 10.5817/sp2011-2-11

Figure 9: Journal-level enrichment overview as displayed in the Website Settings section. The grid will show all enrichments retrieved via the OpenAIRE Broker API for the selected subscriptions

4.2.2 Article-level Enrichments

Enrichments related to individual articles are displayed directly in the **Publication tab** within the article's **Workflow** interface. This allows editors to assess enrichment suggestions in the context of each specific article, making it easier to evaluate and act upon them when updating metadata.

This two-tier display ensures both high-level overview and detailed access, enabling efficient metadata improvement workflows within the editorial process.

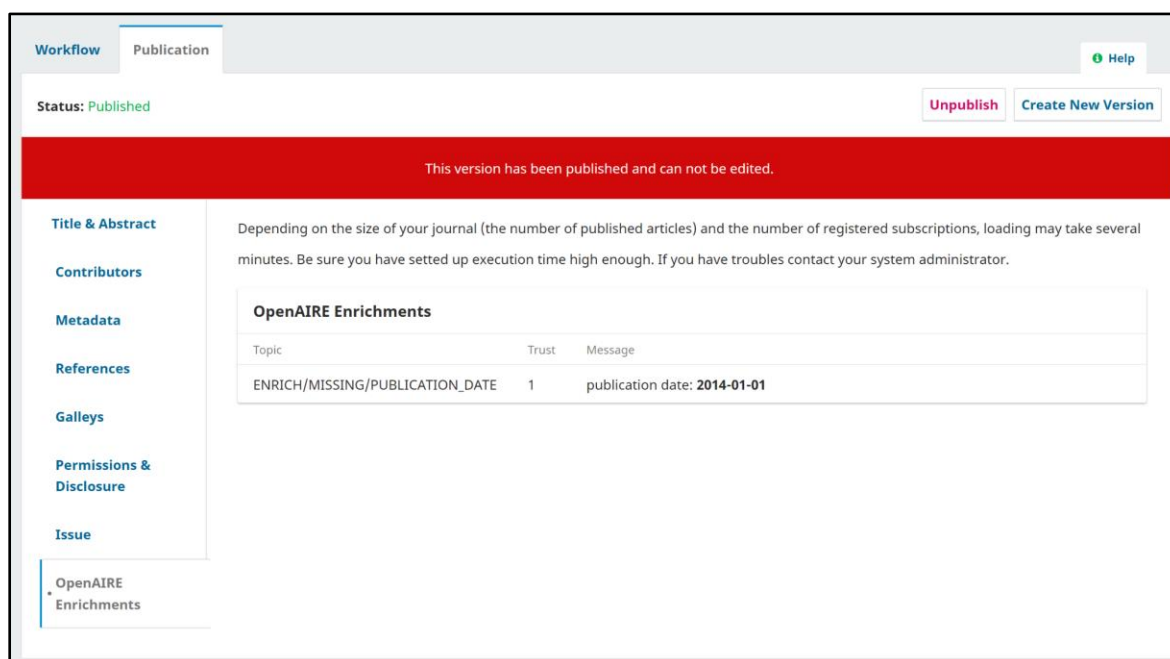


Figure 10: Article-level enrichment display within the Workflow → Publication tab. Only enrichments corresponding to active subscriptions in OJS are shown for the specific article

4.3 Plugin functionality

The plugin communicates with the OpenAIRE Broker API using Hypertext Preprocessors (PHP) native Client for URLs (cURL) function to sequentially load JSON-formatted data. It processes each subscription activated in OJS individually, starting by requesting the first page of enrichment notifications from:

[https://api.openaire.eu/broker/scroll/notifications/bySubscriptionId/\[subscription_id\]](https://api.openaire.eu/broker/scroll/notifications/bySubscriptionId/[subscription_id])

Subsequent pages are loaded using the scroll ID provided in the previous response, with the following pattern:

[https://api.openaire.eu/broker/scroll/notifications/\[scroll_id\]](https://api.openaire.eu/broker/scroll/notifications/[scroll_id])

The scroll ID is only valid for a limited time (approximately 5 minutes), and must be correctly Uniform Resource Locator- (URL) encoded to ensure proper retrieval of data. As a result, all pages of enrichment data for a given subscription must be fetched sequentially within this time frame, starting from the first page and proceeding to the last. Each JSON response includes a **"completed"** parameter. As long as **"completed": false**, more pages are expected. The final response returns **"completed": true**, marking the end of the data stream for that subscription, after which the plugin moves on to the next subscription.

4.3.1 OAI-PMH Matching Requirements

For the plugin to function correctly, the journal's metadata must be harvested by OpenAIRE directly from OJS using the The Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) protocol. The recommended approach is to use the **openAIREstandard plugin**¹, which ensures compliance with the *OpenAIRE Guidelines for Literature Repository Managers v4*². The enrichment records retrieved from OpenAIRE are matched to OJS articles based on their OAI-PMH identifier, which is derived from the internal OJS article ID.

Request was of type GetRecord.

OAI Record: oai:ojs.journals.muni.cz:article/18949

OAI Record Header

OAI Identifier	oai:ojs.journals.muni.cz:article/18949	oai_dc	formats
Datestamp	2022-02-03T14:04:09Z		
setSpec	studia-paedagogica:Studies	Identifiers	Records

Figure 11: Excerpt from the OAI-PMH output of an OJS installation. The highlighted OJS article ID is used as the linking identifier for enrichment matching

ENRICH | MISSING | PUBLICATION_DATE

This is a preview of the top 100 events generated for this topic. If you are interested to receive the full list, please subscribe to the

97 RESULTS, PAGE 10 OF 10

ID: oai:ojs.journals.muni.cz:article/18949

Analýza komunikace ohniskové skupiny ve výchovném ústavu

Show more ▾

Figure 12: Display of the corresponding linking identifier within the OpenAIRE Provide service. This identifier enables correct association between local OJS records and OpenAIRE enrichments

¹ You can download and install the plugin from the MuniPress GitHub repository:

<<https://github.com/munipress/openAIREstandard>>

² For more detailed information about the Aim of OpenAIRE Guidelines for Literature Repository Managers v4:

<<https://openaire-guidelines-for-literature-repository-managers.readthedocs.io/en/v4.0.0/introduction.html>>

The screenshot shows the OJS Publication page for a document titled "An Analysis of a Focus Group Communication in a Young Offenders' Institution" by Vávrová et al. The document is published (18949). The page has tabs for Workflow and Publication, with the Publication tab selected. The status is "Published". A red banner states: "This version has been published and can not be edited." On the left sidebar, there are links for Title & Abstract, Contributors, Metadata, References, Galleys, Permissions & Disclosure, Issue, and OpenAIRE Enrichments. The OpenAIRE Enrichments section is active, displaying a table of enrichments.

Topic	Trust	Message
ENRICH/MISSING/PUBLICATION_DATE	1	publication date: 2014-01-01

Figure 13: Display of the enrichment subscription matched to a specific article in OJS after successful integration with OpenAIRE Enrichments. Only enrichments linked to an active subscription are shown

4.3.2 OJS Limitations

One of the current limitations of the plugin arises from the architectural constraints of OJS. Specifically, the backend environment of OJS lacks sufficient hook coverage, which limits the plugin's ability to integrate its interface freely across all administrative sections of the system.

As a result the most suitable available integration point was found to be the `Template::Settings::website` hook, which has been used to render the journal-level enrichment grid within the *Website Settings* section.

This limitation does not affect the core functionality of the plugin but may constrain future usability extensions unless broader hook support is introduced in future versions of OJS.

4.3.3 Limitations of the Broker API

One of the main limitations of the OpenAIRE Broker API is the dependency on short-lived scroll IDs for paginated data retrieval. Because each scroll ID is valid for only a few minutes, the plugin must complete the entire data retrieval process for each subscription without significant delay. This design constraint makes it impractical to navigate enrichment pages manually, as users would likely exceed the scroll ID validity window. The plugin therefore handles this logic automatically, loading all available enrichment data for each subscription in sequence and without interruption.

4.3.4 Supported Enrichment Types

The plugin allows editors to selectively display enrichment suggestions from OpenAIRE based on their journal's needs. Each type of enrichment is managed through an individual subscription, identified by a unique subscription code. These codes are obtained after registering for specific enrichments in the OpenAIRE Provide interface.

It is not necessary to activate all available enrichments. Editors and journal managers are free to choose which enrichments to subscribe to and import into the OJS plugin interface. After subscriptions are created in OpenAIRE Provide, they can be manually entered into the plugin settings in OJS.

In cases where a journal is part of a multi-journal OJS installation and does not have direct access to the OpenAIRE Provide account, the central system administrator must first activate the subscriptions in OpenAIRE and then share the necessary codes with the relevant editorial teams.

For more information on subscription registration, see Section 4.

Currently supported enrichment types include:

- ENRICH/MISSING/ABSTRACT
- ENRICH/MISSING/AUTHOR/ORCID
- ENRICH/MISSING/OPENACCESS_VERSION
- ENRICH/MISSING/PID
- ENRICH/MISSING/PROJECT
- ENRICH/MISSING/PUBLICATION_DATE
- ENRICH/MISSING/SUBJECT/ACM
- ENRICH/MISSING/SUBJECT/ARXIV
- ENRICH/MISSING/SUBJECT/DDC
- ENRICH/MISSING/SUBJECT/JEL
- ENRICH/MISSING/SUBJECT/MESHEUROPMC
- ENRICH/MORE/OPENACCESS_VERSION
- ENRICH/MORE/PID
- ENRICH/MORE/SUBJECT/ACM

- ENRICH/MORE/SUBJECT/ARXIV
- ENRICH/MORE/SUBJECT/DDC
- ENRICH/MORE/SUBJECT/JEL
- ENRICH/MORE/SUBJECT/MESHEUROPMC

Each enrichment helps improve metadata quality by identifying missing or potentially better values for common metadata fields such as persistent identifiers (PIDs), abstracts, OA status, and various subject classifications (e.g. DDC, JEL, ACM, MeSH, arXiv).

4.3.5 Validation and Filtering Logic

At both the **article** and **journal** levels, the plugin strictly validates each enrichment entry against the list of active subscriptions stored in OJS. This means that only enrichments matching a correctly configured subscription are displayed. For example, if an editor has added a subscription for **ENRICH/MISSING/PUBLICATION_DATE** only, then this is the sole enrichment type that will appear across both views. Any enrichment type not linked to a stored subscription is automatically excluded, regardless of whether it is present in the response from the OpenAIRE Broker API.

This validation ensures consistency and accuracy in what is displayed to editors and journal managers. Enrichments are never shown unless they match a topic that has been explicitly registered via OpenAIRE Provide and saved in OJS.

It is important to note that the system is sensitive to incorrect configuration. If a subscription code is placed in the wrong field or mismatched with the intended enrichment topic, the plugin will not display any enrichments for that subscription. Therefore, correct setup is essential — each enrichment type requires a corresponding and correctly assigned subscription entry in the plugin configuration.

This design reinforces the plugin's underlying logic of **1 subscription = 1 enrichment topic**, ensuring both clarity and control for journal administrators.

OpenAIRE Enrichments ×

Subscriptions list
Below, you can configure subscriptions to enrichments you are subscribed to via your account in OpenAIRE Provide.

ENRICH/MORE/OPENACCESS_VERSION

ENRICH/MORE/PID

ENRICH/MISSING/AUTHOR/ORCID

ENRICH/MISSING/PID

ENRICH/MISSING/ABSTRACT

ENRICH/MISSING/SUBJECT/DDC

ENRICH/MORE/SUBJECT/DDC

ENRICH/MISSING/SUBJECT/JEL

ENRICH/MORE/SUBJECT/JEL

ENRICH/MISSING/PUBLICATION_DATE

Figure 14: Example of a correct plugin configuration in OJS for displaying enrichments related to missing PIDs and publication dates. The subscriptions for ENRICH/MISSING/PID and ENRICH/MISSING/PUBLICATION_DATE have been properly assigned, enabling the corresponding enrichments to be shown.

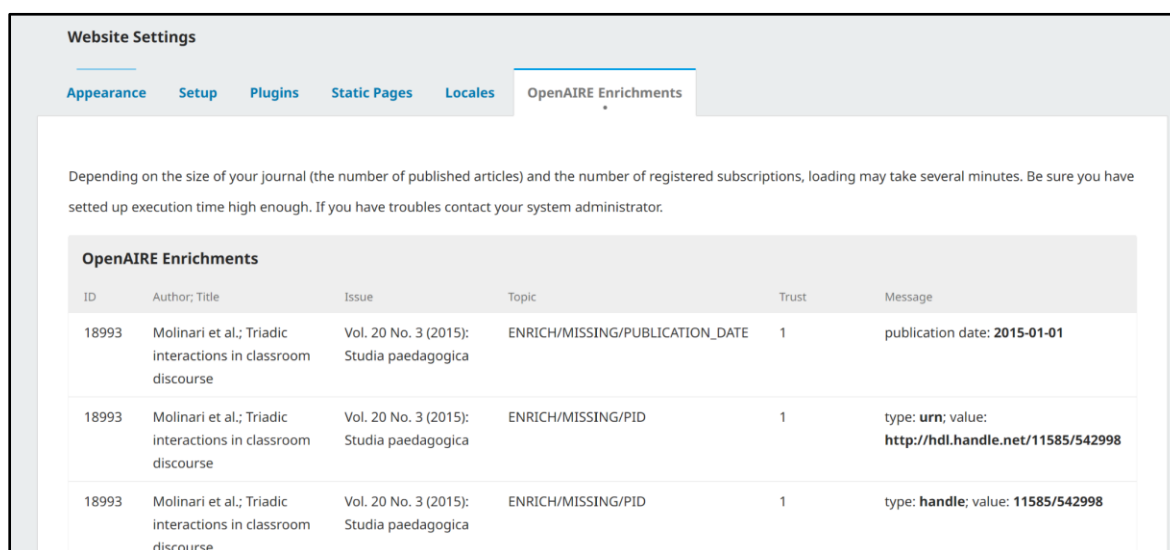
4.3.6 Version Compatibility Adjustments

Due to changes in the layout of the OJS backend interface in version 3.4, visual adjustments were necessary to ensure the plugin remains user-friendly and accessible across different OJS versions.

Specifically, OJS 3.4 uses a narrower layout for backend pages, which caused display issues with the original enrichment grid designed for wider layouts (such as in OJS 3.2.1). As a result, several modifications were implemented for journal-level enrichment listings:

- The **“Trust”** column was moved directly into the body of each enrichment message to conserve horizontal space.
- The repeated prefix *ENRICH/* was removed from enrichment type labels to improve readability and reduce visual clutter, given that all enrichment topics already share this common structure.

These changes affect only the **journal-level enrichment view**. The article-level view remains unchanged and continues to display full enrichment topic identifiers where applicable.



ID	Author; Title	Issue	Topic	Trust	Message
18993	Molinari et al.; Triadic interactions in classroom discourse	Vol. 20 No. 3 (2015): Studia paedagogica	ENRICH/MISSING/PUBLICATION_DATE	1	publication date: 2015-01-01
18993	Molinari et al.; Triadic interactions in classroom discourse	Vol. 20 No. 3 (2015): Studia paedagogica	ENRICH/MISSING/PID	1	type: urn ; value: http://hdl.handle.net/11585/542998
18993	Molinari et al.; Triadic interactions in classroom discourse	Vol. 20 No. 3 (2015): Studia paedagogica	ENRICH/MISSING/PID	1	type: handle ; value: 11585/542998

Figure 15: Shows the original layout in OJS 3.2.1 with a wider column layout

Website Settings				
Appearance	Setup	Plugins	Static Pages	OpenAIRE Enrichments
<p>Depending on the size of your journal (the number of published articles) and the number of registered subscriptions, loading may take several minutes. Be sure you have setted up execution time high enough. If you have troubles contact your system administrator.</p>				
OpenAIRE Enrichments				
ID	Author; Title	Issue	Topic	Message
18993	Molinari et al.; Triadic interactions in classroom discourse	Vol. 20 No. 3 (2015): Studia paedagogica	MISSING/PUBLICATION_DATE	Trust: 1 publication date: 2015-01-01
18993	Molinari et al.; Triadic interactions in classroom discourse	Vol. 20 No. 3 (2015): Studia paedagogica	MISSING/PID	Trust: 1 type: urn ; value: http://hdl.handle.net/11585/542998
18993	Molinari et al.; Triadic interactions in classroom discourse	Vol. 20 No. 3 (2015): Studia paedagogica	MISSING/PID	Trust: 1 type: handle ; value: 11585/542998

Figure 16: Shows the updated, more compact layout in OJS 3.4 with the adjusted display logic

4.3.7 Core Architecture and Functional Structure of the Plugin

The core functionality of the OpenAIRE Broker Service Plugin is divided into five interrelated components, each represented by a dedicated PHP class. Together, they ensure the seamless retrieval, processing, and display of enrichment data in OJS.

1. Plugin Initialization and Base Setup

Class: *OpenAIREBrokerServicePlugin*

This class handles the overall integration of the plugin into the OJS platform by:

- Extending the Context schema to store enrichment-related settings.
- Registering UI components for displaying enrichments at both the journal level and article level.
- Registering the necessary grid handlers.
- Ensuring proper plugin configuration and user access to the settings interface.

It serves as the main entry point for enabling the plugin's core functionalities within the OJS environment.

2. Subscription Configuration and Storage

Class: *OpenAIREBrokerServiceSettingsForm*

This class is responsible for managing the settings interface and ensuring that subscription data is:

- Properly structured and validated.
- Saved into the OJS database as part of the Context object configuration.
- Retrieved and reloaded correctly for display and editing.

It provides the administrative backend interface for editors to register and manage their desired enrichment subscriptions.

3. Article-Level Grid Display

Class: *OpenAIREBrokerServiceGridHandler*

This component:

- Initializes and renders the enrichment grid in the *Workflow* → *Publication tab* for each individual article.
- Displays only those enrichments that correspond to valid subscriptions saved in the system.
- Facilitates a clear, contextual view of enrichment metadata per article.

4. Journal-Level Grid Display

Class: *OpenAIREBrokerServiceContextGridHandler*

This handler is responsible for:

- Initializing and rendering the enrichment grid at the journal level, accessible via the *Website Settings tab*.
- Providing an aggregated view of all enrichment messages across the journal's content.

5. API Integration and Enrichment Data Processing

Class: *OpenAIREBrokerServiceEnrichments*

This crucial class manages all communication with the OpenAIRE Broker API. It:

- Connects to the Broker API using two different endpoints:
 - For the initial subscription:
[https://api.openaire.eu/broker/scroll/notifications/bySubscriptionId/\[subscription_id\]](https://api.openaire.eu/broker/scroll/notifications/bySubscriptionId/[subscription_id])
 - For subsequent pages:
[https://api.openaire.eu/broker/scroll/notifications/\[scroll_id\]](https://api.openaire.eu/broker/scroll/notifications/[scroll_id])
- Loads and validates all registered subscriptions.
- Matches enrichments against articles via their OAI-PMH identifiers.
- Verifies subscription correctness and ensures that only data for properly registered enrichment types is displayed.

- Parses the raw JSON responses into a clean, readable format tailored for editors and journal managers.

This modular structure ensures that the plugin remains maintainable, extensible, and well-aligned with OJS's architecture. Each class focuses on a clearly defined responsibility, supporting both current functionalities and future enhancements.

4.4 Possible Future Improvements

4.4.1 Performance Improvements by adding Database

Currently, the plugin connects to the OpenAIRE Broker API on each page load and actively checks for any newly available enrichment information. This approach is appropriate in scenarios where enrichment data on the OpenAIRE side is updated frequently and dynamically. However, in practice, enrichments are not updated continuously nor at predictable regular intervals.

A possible improvement would therefore be to change the plugin's behavior from live data retrieval to a cached data model. Instead of fetching enrichment data from the API during every user request, the plugin could store enrichment data locally in the database and serve it from there. This would significantly improve the performance and loading time of the enrichment display within OJS.

A reasonable update strategy would involve refreshing the cached data once per week. During this update, the user would experience a full API data fetch and longer loading time. However, all subsequent interactions would retrieve data directly from the local database, ensuring a faster and more efficient user experience.

Implementing this approach would require extending the plugin with:

- A custom database schema to store enrichment data.
- A logic layer to determine whether to load data from the local database or perform a fresh API call.
- A scheduled or user-triggered mechanism to update the stored data periodically.

While this enhancement would add complexity—most notably the need to maintain a database schema and manage data persistence—it would also open new possibilities for future development and scalability. The current plugin version avoids this complexity by design, as it does not introduce any changes to the database schema.

4.4.2 Adding New Subscriptions Dynamically

One of the major strengths of the current plugin architecture is its flexibility to support the integration of **new enrichment types** as they become available within the OpenAIRE Broker ecosystem. The plugin is not strictly bound to a fixed list of subscriptions and allows developers to extend it with minimal changes to the codebase.

The following steps outline how new enrichment subscriptions can be added:

1. Define the Subscription in the class *OpenAIREBrokerServicePlugin*

The plugin class contains a configuration constant named **CONFIG_VARS**, which defines the available enrichment subscription fields. To support a new enrichment type, simply add a new entry to this constant.

```
class OpenAIREBrokerServicePlugin extends GenericPlugin {

    const CONFIG_VARS = array(
        'enrich_more_openaccess_version' => 'string',
        'enrich_more_pid' => 'string',
        'enrich_missing_author_orcid' => 'string',
        'enrich_missing_pid' => 'string',
        'enrich_missing_abstract' => 'string',
        'enrich_missing_subject_ddc' => 'string',
        'enrich_more_subject_ddc' => 'string',
        'enrich_missing_subject_jel' => 'string',
        'enrich_more_subject_jel' => 'string',
        'enrich_missing_publication_date' => 'string',
        'enrich_missing_openaccess_version' => 'string',
        'enrich_missing_subject_acm' => 'string',
        'enrich_more_subject_acm' => 'string',
        'enrich_missing_project' => 'string',
        'enrich_missing_subject_mesheuropmc' => 'string',
        'enrich_more_subject_mesheuropmc' => 'string',
        'enrich_missing_subject_arxiv' => 'string',
        'enrich_more_subject_arxiv' => 'string'
    );
}
```

Figure 17: Configuration constant CONFIG_VARS in OpenAIREBrokerServicePlugin.php

2. Add the Setting to the Plugin Settings Handler

The same constant (CONFIG_VARS) is also referenced in the plugin's settings handler class *OpenAIREBrokerServiceSettingsForm*, where it defines the fields to be stored in the database.

```
class OpenAIREBrokerServiceSettingsForm extends Form {

  /** @var int */
  var $_contextId;

  /** @var object */
  var $_plugin;

  /** @var context */
  var $_context;

  const CONFIG_VARS = array(
    'enrich_more_openaccess_version' => 'string',
    'enrich_more_pid' => 'string',
    'enrich_missing_author_orcid' => 'string',
    'enrich_missing_pid' => 'string',
    'enrich_missing_abstract' => 'string',
    'enrich_missing_subject_ddc' => 'string',
    'enrich_more_subject_ddc' => 'string',
    'enrich_missing_subject_jel' => 'string',
    'enrich_more_subject_jel' => 'string',
    'enrich_missing_publication_date' => 'string',
    'enrich_missing_openaccess_version' => 'string',
    'enrich_missing_subject_acm' => 'string',
    'enrich_more_subject_acm' => 'string',
    'enrich_missing_project' => 'string',
    'enrich_missing_subject_mesheuropmc' => 'string',
    'enrich_more_subject_mesheuropmc' => 'string',
    'enrich_missing_subject_arxiv' => 'string',
    'enrich_more_subject_arxiv' => 'string'
  );
}
```

Figure 18: Usage of CONFIG_VARS in the plugin settings class

3. Extend the Settings Template

Next, update the plugin's settings form template: *templates/settingsForm.tpl*

Add a new form input field corresponding to the new subscription key, allowing editors to configure the new enrichment subscription via the OJS interface.

4. Enable Processing in *OpenAIREBrokerServiceEnrichments*

To ensure the new enrichment data is correctly processed and displayed:

Extend the array protected `$_enrichs` in the *OpenAIREBrokerServiceEnrichments* class by adding a mapping like: `'ENRICH/MORE/OPENACCESS_VERSION' => 'enrich_more_openaccess_version'`

Update the *resultsToArray()* function to include logic for parsing and formatting the new enrichment type messages so that it appears correctly in both journal-level and article-level listings.

These enhancements allow the plugin to stay up-to-date with future developments in OpenAIRE Broker enrichments, providing a sustainable and scalable approach to enrichment integration.

5 JOURNAL'S SUBSCRIPTION REGISTRATION IN OPENAIRE PROVIDE

5.1 Preliminary Step – Preparation

Before starting the registration process, it is recommended to install and activate the [OpenAIREstandard plugin](#) in OJS to ensure proper metadata export in accordance with the OpenAIRE guidelines.

You can independently verify whether the new metadata format is active and whether the generated XML is correctly structured for harvesting your articles via your OAI-PMH endpoint (e.g. <https://your-journal.org/index.php/abc/oai>).

The full registration and validation process is described in **D6.1 OJS Connector for OpenAIRE Research Graph**³.

Metadata Format

metadataPrefix	oai_openaire
metadataNamespace	https://openaire-guidelines-for-literature-repository-managers.readthedocs.io/en/v4.0.0/application_profile.html
schema	https://www.openaire.eu/schema/repo-lit/4.0/openaire.xsd

Figure 19: Metadata format with metadataPrefix oai_openaire

³ CRAFT-OA Deliverable 6.1_OJS Connector for OpenAIRE Research Graph (Gomola, Radek, Dvořáková, Martina, Růžička, Michal): <<https://zenodo.org/doi/10.5281/zenodo.12633202>>


```
<resource xmlns:schemaLocation="http://namespace.openaire.eu/schema/openaire https://www.openaire.eu/schema/repo-lit/4.0/openaire.xsd" >
  <datacite:title>
    <datacite:title xml:lang="en" >I'll be there for you? The bystander intervention model and cyber aggression</datacite:title>
  </datacite:title>
  <datacite:creators>
    <datacite:creator>
      <datacite:creatorName nameType="personal" >Karasavva, Vasileia</datacite:creatorName>
      <datacite:givenName>Vasileia</datacite:givenName>
      <datacite:familyName>Karasavva</datacite:familyName>
      <datacite:affiliation>Department of Psychology, University of British Columbia, Vancouver, BC, Canada</datacite:affiliation>
      <datacite:nameIdentifier nameIdentifierScheme="ORCID" schemeURI="http://orcid.org" >https://orcid.org/0000-0003-0778-8454</datacite:nameIdentifier>
    </datacite:creator>
    <datacite:creator>
      <datacite:creatorName nameType="personal" >Mikami, Amori</datacite:creatorName>
      <datacite:givenName>Amori</datacite:givenName>
      <datacite:familyName>Mikami</datacite:familyName>
      <datacite:affiliation>Department of Psychology, University of British Columbia, Vancouver, BC, Canada</datacite:affiliation>
      <datacite:nameIdentifier nameIdentifierScheme="ORCID" schemeURI="http://orcid.org" >https://orcid.org/0000-0001-5821-0674</datacite:nameIdentifier>
    </datacite:creator>
  </datacite:creators>
  <datacite:alternatId>
    <datacite:alternatId identifierType="DOI" >10.5817/CP2024-2-1</datacite:alternatId>
  </datacite:alternatId>
  <dc:language>en</dc:language>
  <dc:source>eISSN: 1802-7962</dc:source>
  <dc:publisher>Masaryk University, Faculty of Social Studies</dc:publisher>
  <datacite:dates>
    <datacite:date dateType="Issued" >2024-04-11</datacite:date>
  </datacite:dates>
  <oaire:resourceType uri="http://purl.org/coar/resource_type/c_2df8fbb1" resourceTypeGeneral="literature" >research article</oaire:resourceType>
  <dc:description xml:lang="en" >The Bystander Intervention Model (BIM) has been validated for face-to-face emergencies and dictates that observers' decision to intervene hinges on five sequential experiences with depression, social anxiety, and cyber aggression either as the target or the aggressor influence bystanders. In our pre-registered study, emerging adults (N = 1,093) viewed pilot-tested cy would take, most participants chose non-intervention (36.3%) or private direct intervention (39.4%). Path analysis suggested that overall, the BIM can explain bystanders' responses to cyber aggression. face applications of the BIM that prescribes barriers to affect only a single specific step, here we found some barriers were negatively linked to multiple steps. These findings elucidate ways in which cy
  </dc:description>
  <dc:format>application/pdf</dc:format>
  <datacite:identifier identifierType="URL" >https://cyberpsychology.eu/article/view/36754</datacite:identifier>
  <datacite:rights identifierType="http://purl.org/coar/access_right/c_abf2" >open access</datacite:rights>
  <datacite:subjects>
    <datacite:subject xml:lang="en" >cyber aggression</datacite:subject>
    <datacite:subject xml:lang="en" >Bystander Intervention Model</datacite:subject>
    <datacite:subject xml:lang="en" >helping</datacite:subject>
    <datacite:subject xml:lang="en" >cyberbullying</datacite:subject>
  </datacite:subjects>
  <oaire:license startDate="2024-04-11" uri="https://creativecommons.org/licenses/by-sa/4.0/" >CC Attribution-ShareAlike 4.0</oaire:licenseCondition>
  <datacite:size>
    <datacite:size>4.69 MB</datacite:size>
  </datacite:size>
  <oaire:file accessRightsURI="http://purl.org/coar/access_right/c_abf2" mimeType="application/pdf" objectType="fulltext" >https://cyberpsychology.eu/article/download/36754/32534</oaire:file>
  <oaire:citationTitle>Cyberpsychology: Journal of Psychosocial Research on Cyberspace</oaire:citationTitle>
  <oaire:citationVolume>18</oaire:citationVolume>
  <oaire:citationIssue>2</oaire:citationIssue>
</resource>
```

Figure 20: XML generated with metadataPrefix oai_openaire

5.2 Registration Process

1. Login to OpenAIRE Provide:

- Go to <https://provide.openaire.eu>.
- Log in via Authentication and Authorization Infrastructure (AAI) (eduGAIN) using your institutional credentials or create an OpenAIRE account.

2. Register a New Data Source (Journal):

- Fill in the required journal information:
 - i. *Software Platform*: OJS
 - ii. *Official Name, International Standard Serial Number (ISSN), Description, Country, Entry URL, Admin Email, etc.*
 - iii. *Data Source Type*: Institutional/Thematic Repository

3. Register the Interface (OAI-PMH):

- Provide the base OAI-PMH URL (e.g. <https://your-journal.org/index.php/abc/oai>)
- Configure:
 - i. *Set (optional)*: restrict harvesting to specific journal sections (e.g., ART for articles)
 - ii. *Desired compatibility level*: Always select OpenAIRE 4.0 (institutional/thematic repo)

4. Accept Terms of Use:

- Confirm the reuse conditions and terms of data harvesting, including full-text reuse if applicable.

5. Validation Process:

- Once the registration is submitted, automatic validation is triggered.
- Metadata are checked against the **OpenAIRE Guidelines for Literature, Institutional and Thematic Repositories v4.0**⁴ using the OpenAIRE Validator.

5.3 Subscriptions

For each of your registered journals in your OpenAIRE Provide account, there is an **“Enrichments”** tab where you can view all available enrichment types for which subscriptions can be created. Each enrichment type can be explored in more detail by clicking its link.

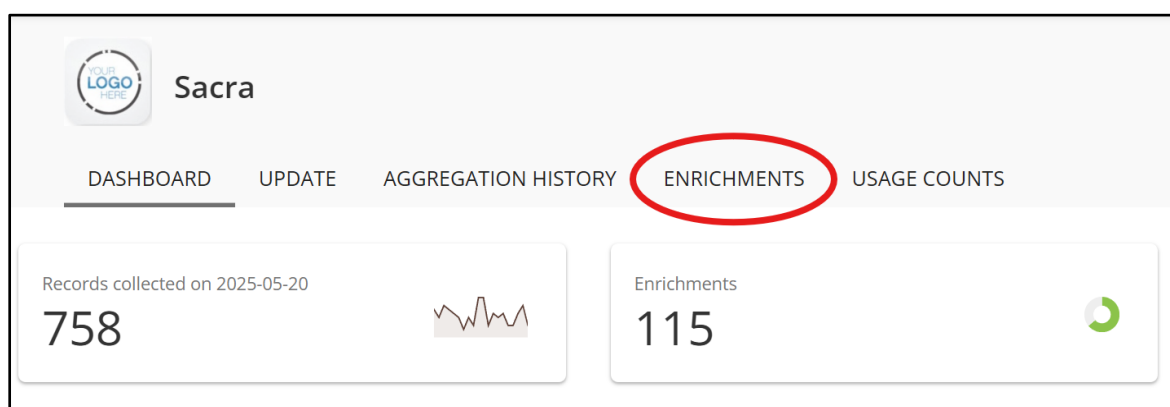


Figure 21: The figure shows the Enrichments tab in the journal dashboard within the OpenAIRE Provide administration interface

To register a subscription for use in OJS, click the **“Subscribe to these events”** button located in the **top right corner** of the page. After clicking the button, you will receive an **email containing the Subscription ID**. This Subscription ID must be copied and inserted into the corresponding field in OJS to enable the connection with the selected enrichment data.

⁴ For more detailed information about the Aim of OpenAIRE Guidelines for Literature Repository Managers v4: <https://openaire-guidelines-for-literature-repository-managers.readthedocs.io/en/v4.0.0/introduction.html>

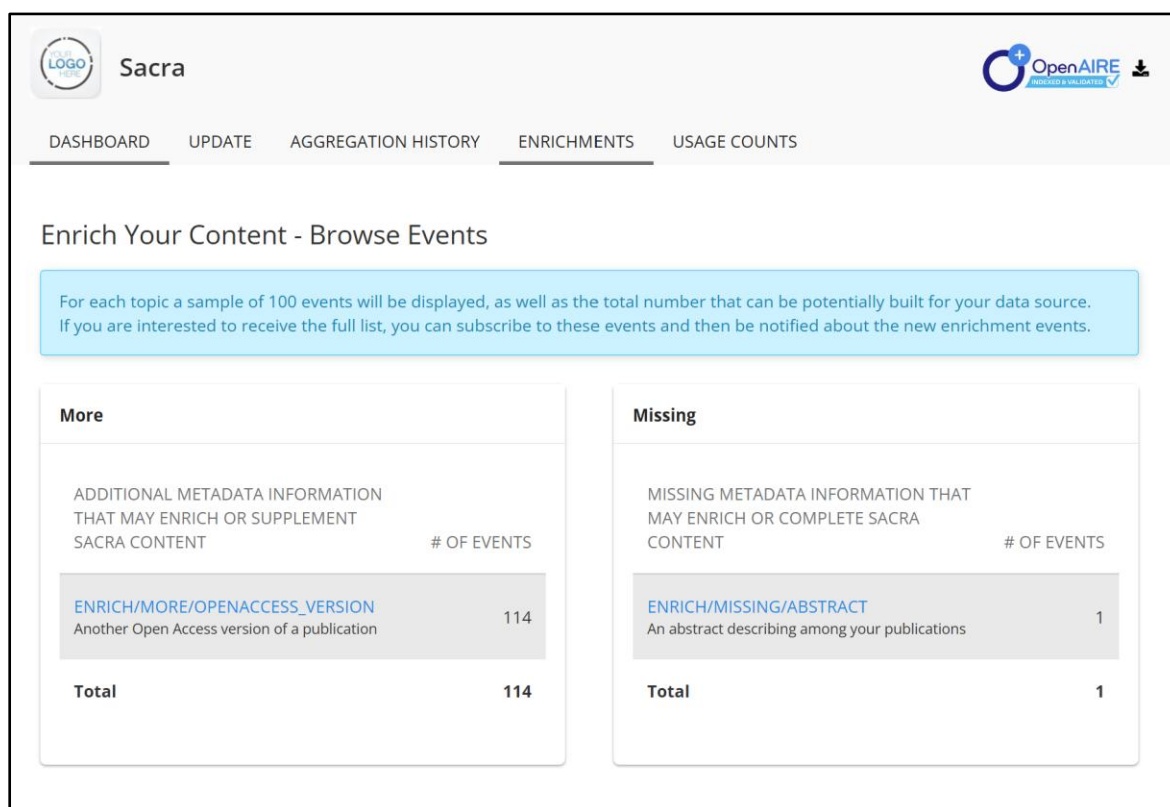


Figure 22: The figure displays all available enrichments that can be subscribed to via the Subscriptions feature

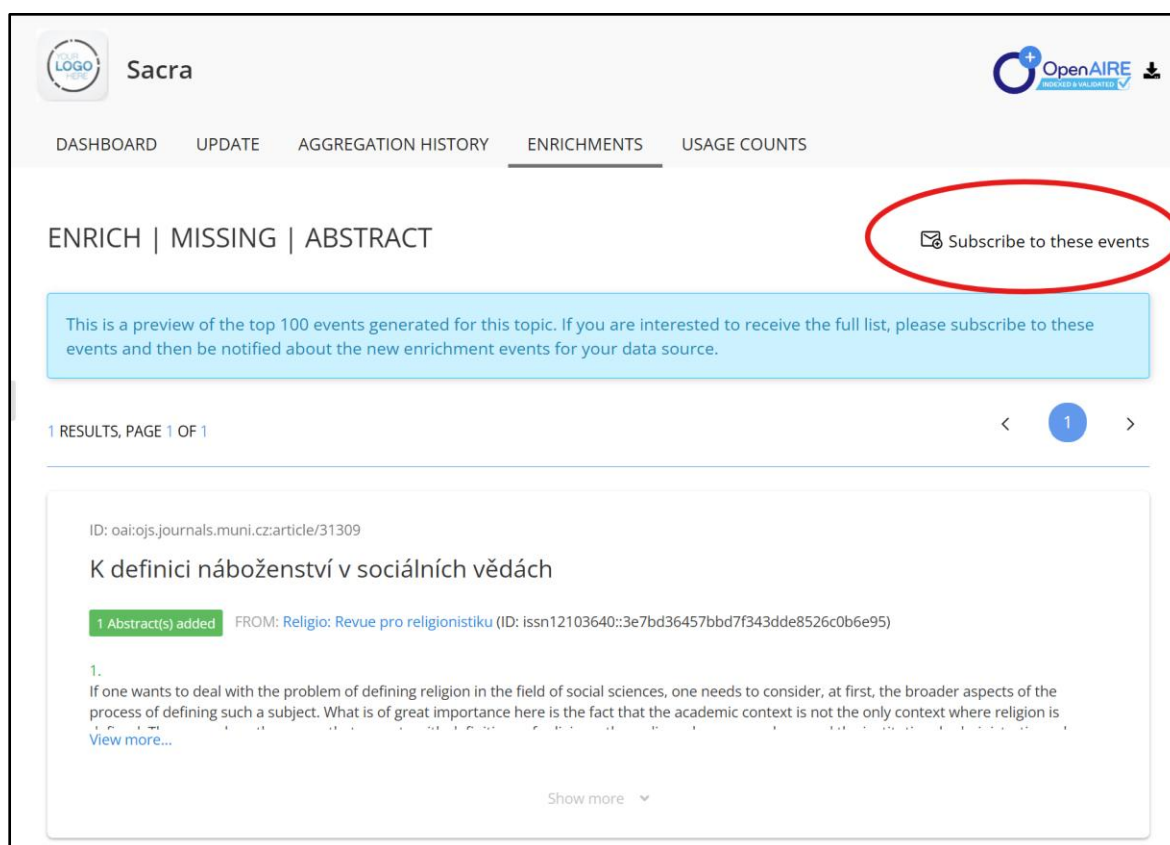


Figure 23: The figure shows a specific enrichment with the option to "Subscribe to this event"

Note: The confirmation email may take some time to arrive — it is not always delivered immediately.

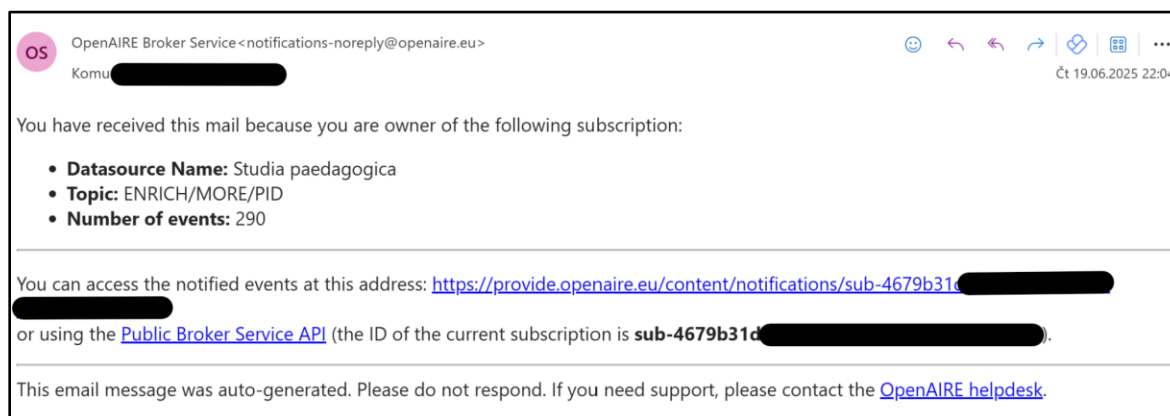


Figure 24: The figure displays the confirmation email containing the Subscription ID required for registration in OJS

6 EVALUATION

The initial validation of the new plugin was carried out on 2 selected journals hosted on the **MUNI Journals portal**⁵. These journals were used to verify the correct functionality of the plugin and its ability to successfully connect to the OpenAIRE Broker service and process the returned data.

Due to the extended limited availability of the Enrichments service for broader public use, further testing and real-world deployment of the plugin will proceed gradually. As with any custom-developed plugin for OJS, we expect ongoing updates and enhancements based on user feedback, further testing, and the long-term goal of improving sustainability and functionality.

The plugin is designed to support multiple OJS versions and is prepared for long-term use and maintenance. Future versions will continue to improve enrichment handling, data caching, and overall performance as the OpenAIRE Broker service evolves.

⁵ See: <<https://journals.muni.cz/>>



7 CONCLUSION

This document has presented the development and implementation of a new OJS plugin designed to facilitate the integration of enrichment data from the OpenAIRE Broker API into the OJS environment. Unlike the previous export-focused plugin (openAIREstandard plugin⁶), this tool enables real-time access to selected metadata enrichments—such as missing PIDs, ORCID identifiers, or publication dates—directly within the OJS backend. It provides journal editors with an intuitive interface to review enrichment suggestions both at the article and journal level, helping them improve metadata quality in line with OpenAIRE standards.

The plugin is fully configurable through the OJS settings interface and allows editors to selectively subscribe to specific enrichment types (topics). The integration logic ensures that only enrichments matching valid and explicitly registered subscriptions are displayed, minimizing noise and maximizing relevance.

The current version of the plugin is compatible with OJS 3.2.1, 3.3, and 3.4. It has been deployed in a production environment and is actively used by journals at Masaryk University, contributing to improved metadata quality and discoverability through OpenAIRE services. The plugin source code is openly available on GitHub at <https://github.com/munipress/openAIREBrokerService>, enabling further adoption and collaborative development.

Future improvements may include local caching of enrichment data to enhance performance, extended support for additional enrichment types as they become available through the OpenAIRE Broker, as well as the functionality to provide feedback about the correctness of the enrichments, so the editors can inform OpenAIRE that an enrichment is wrong and should not be generated anymore.

⁶ Downloadable on github: <<https://github.com/munipress/openAIREstandard>>

8 REFERENCES

8.1 List of References

OpenAIRE Broker API documentation .

Available at: <https://graph.openaire.eu/docs/apis/broker-api/> (Accessed 27/06/2025)

Gomola, R. (2025) *OpenAIRE Broker API – Enrichments. Open Journal Systems plugin.*

[Computer software]. Available at <https://github.com/munipress/openAIREBrokerService>

8.2 List of Websites

- <https://github.com/munipress/openAIREBrokerService>
- <https://graph.openaire.eu/docs/apis/broker-api>
- <https://graph.openaire.eu/docs/category/downloads>
- <https://spark.apache.org/>
- <https://api.openaire.eu/broker/swagger-ui/index.html>
- <https://github.com/munipress/openAIREstandard>
- <https://openaire-guidelines-for-literature-repository-managers.readthedocs.io/en/v4.0.0/introduction.html>
- <https://provide.openaire.eu>
- <https://journals.muni.cz/>

9 LIST OF FIGURES

Figure 1: The aggregation-enrichment-deduplication pipeline of the OpenAIRE Graph	10
Figure 2: Screenshot of OpenAIRE Provide for the journal “Studia paedagogica”	13
Figure 3: Preview of events generated for the journal “Studia paedagogica”	13
Figure 4: Preview of events about ORCID identifiers of authors.....	14
Figure 5: Notifications for the subscription to the topic ENRICH/MORE/OPENACCESS_VERSION	14
Figure 6: Active branches in GitHub project.....	16
Figure 7: The plugin is listed in the generic folder.....	17
Figure 8: The plugin listed in the List of Plugins	17
Figure 9: Journal-level enrichment overview as displayed in the Website Settings section. The grid will show all enrichments retrieved via the OpenAIRE Broker API for the selected subscriptions	18
Figure 10: Article-level enrichment display within the Workflow → Publication tab. Only enrichments corresponding to active subscriptions in OJS are shown for the specific article	19
Figure 11: Excerpt from the OAI-PMH output of an OJS installation. The highlighted OJS article ID is used as the linking identifier for enrichment matching	20
Figure 12: Display of the corresponding linking identifier within the OpenAIRE Provide service. This identifier enables correct association between local OJS records and OpenAIRE enrichments	20
Figure 13: Display of the enrichment subscription matched to a specific article in OJS after successful integration with OpenAIRE Enrichments. Only enrichments linked to an active subscription are shown	21
Figure 14: Example of a correct plugin configuration in OJS for displaying enrichments related to missing PIDs and publication dates. The subscriptions for ENRICH/MISSING/PID and ENRICH/MISSING/PUBLICATION_DATE have been properly assigned, enabling the corresponding enrichments to be shown.....	24
Figure 15: Shows the original layout in OJS 3.2.1 with a wider column layout.....	25
Figure 16: Shows the updated, more compact layout in OJS 3.4 with the adjusted display logic	26

Figure 17: Configuration constant CONFIG_VARS in OpenAIREBrokerServicePlugin.php	29
Figure 18: Usage of CONFIG_VARS in the plugin settings class.....	30
Figure 19: Metadata format with metadataPrefix oai_openaire	32
Figure 20: XML generated with metadataPrefix oai_openaire	33
Figure 21: The figure shows the Enrichments tab in the journal dashboard within the OpenAIRE Provide administration interface.....	34
Figure 22: The figure displays all available enrichments that can be subscribed to via the Subscriptions feature.....	35
Figure 23: The figure shows a specific enrichment with the option to "Subscribe to this event"	35
Figure 24: The figure displays the confirmation email containing the Subscription ID required for registration in OJS	36

10 LIST OF TABLES

Table 1: Broker event topics	12
------------------------------------	----

