EMPOLIS INFORMATION MANAGEMENT

ICMS INFORMATION CONTENT MANAGEMENT SOLUTIONS

IIRDS REQUEST API
HOW TO USE THE API

tekom iiRDS

2024-05-27

Content

1 About the API

This documentation provides additional information to the iiRDS API specification. The API enables to access data via iiRDS, catering to increasing business demands. It will offer system providers and programmers a standardized interface, ensuring modern and state-of-the-art functionality to support seamless integration.

The iiRDS API offers extended capabilities by covering a broad range of use cases and allows developers to choose the most suitable query method for their specific needs. The iiRDS API specific query language uses JSON AST (abstract syntax tree) constructs to query the requested data. As JSON is supported by most programming languages it can be easily integrated into different environments, which facilitates interoperability. The query language supports hierarchical queries, selection of returned properties, and ordering of results. It tries to find balance between simplicity of implementation and feature richness. As iiRDS is based on RDF it would be logical to use SPARQL as the query language. Implementing SPARQL on non-native RDF applications could set a big obstacle to implementors. However, an optional end point for SPARQL queries is part of the API.

As JSON is natively supported in JavaScript, it is a common standard for use in web-based applications and APIs. It facilitates the diverse applicability of the API specification in various environments. In addition, the JSON query is extensible and allows the definition of custom data structures and formats should this be required by iiRDS updates or specific extensions. This makes it possible to adapt the query to specific requirements. There is a large community of developers who support JSON due to its simplicity, flexibility, and widespread use. This results in the availability of numerous tools, libraries, and resources that facilitate the process of working with JSON.

For the same reasons, JSON-LD (see https://json-ld.org/) is used as the default response format. JSON-LD is a lightweight Linked Data format, building upon the widely used JSON format. It facilitates the readability and writing of data for humans while enabling the representation of graph-based data. Ideal for programming environments and RESTful Web services, JSON-LD offers a seamless approach to representing and exchanging data. Moreover, there is a defined serialization of RDF to JSON-LD. Nevertheless, implementers are permitted to choose to support native RDF-XML as a format for query results.

2 Overview of the End Points

While there are specific endpoints, a separate endpoint was not created for each iiRDS class to query specific data. This maintains the independence of the iiRDS specification regarding iiRDS versions. The QUERY endpoints allow for the querying of all data and the application of filter mechanisms.

IIRDS RESOURCES

POST `/api/{apiversion}/resources`
Retrieve iiRDS RDF resources via a query

GET `/api/{apiversion}/resources/schema`
Retrieve the iiRDS schema

GET `/api/{apiversion}/resources/{iri}`
Retrieve an iiRDS resource by IRI

GET `/api/{apiversion}/resources/informationunits/topics/{iri}`

Retrieve properties of an iirds:Topic

GET `/api/{apiversion}/resources/informationunits/fragments/{iri}`
Retrieve properties of an iirds:Fragment

GET `/api/{apiversion}/resources/informationunits/documents/{iri}`
Retrieve properties of an iirds:Document

GET `/api/{apiversion}/resources/informationunits/packages/{iri}`
Retrieve properties of an iirds:Package

IIRDS DIRECTORY NODE

GET `/api/{apiversion}/resources/informationunits/{iri}/directorynodes`
Retrieve properties for all iirds:DirectoryNode that belong to a given iirds:InformationUnit

GET `/api/{apiversion}/resources/directorynodes/{iri}/roots`
Retrieve properties for all root level iirds:DirectoryNodes of a given iirds:DirectoryNode

GET `/api/{apiversion}/resources/directorynodes/{iri}`
Retrieve properties and object resources for an iirds:DirectoryNode

GET `/api/{apiversion}/resources/directorynodes/{iri}/children`
Retrieve properties for all children of a given iirds:DirectoryNode

GET `/api/{apiversion}/resources/directorynodes/{iri}/parents`
Retrieve properties for all parents of a given iirds:DirectoryNode

GET `/api/{apiversion}/resources/directorynodes/{iri}/previous`
Retrieve properties for the previous sibling iirds:DirectoryNode of a given iirds:DirectoryNode

GET `/api/{apiversion}/resources/directorynodes/{iri}/next`
Retrieve properties for the following sibling iirds:DirectoryNode of a given iirds:DirectoryNode

GET `/api/{apiversion}/resources/directorynodes/{iri}/trees`
Retrieve directory node trees of iirds:DirectoryNodes

IIRDS PACKAGE

GET `/api/{apiversion}/resources/informationunits/packages/{iri}/files/{iiRDSPackageFilePath}`
Retrieve a specific file from an iiRDS package

GET `/api/{apiversion}/resources/package/{iri}/content`
Retrieve an iiRDS Container as ZIP archive

RENDITION CONTENT

GET `/api/{apiversion}/resources/informationunits/{iri}/renditions/{mimetype}/content`
Retrieve the (rendition) content of a given iirds:InformationUnit in a specific format

SPARQL

POST `/api/{apiversion}/sparql`
Retrieve results of a SPARQL query

ADMINISTRATION

GET `/api/{apiversion}/features`
Retrieve supported API features

3 Using the API

Details and descriptions of the endpoints can be found in the OpenAPI specification. Further information is described in this chapter.

3.1 OpenAPI Specification

An OpenAPI specification is the description of a RESTful API. It defines the structure, endpoints, parameters, response formats and other important details of the API. The OpenAPI specification is written in YAML format and serves as the basis for the documentation, testing and automation of the API. (see also: https://swagger.io/specification/)

3.2 Display of an OpenAPI Specification

The OpenAPI standard allows an OpenAPI specification to be displayed as a HTML page. Swagger, the publisher of the standard, as well as third-party providers such as ReDocly, offer free viewers for this purpose, which are available as web applications, standalone applications or plugins within third-party software (e.g. ReDoc, SwaggerUI).

The OpenAPI specification of the iiRDS request API is optimized for display with ReDoc, yet also comaptible with SwaggerUI.

3.3 Parameters

The API specification requires parameters as part of requests. The following parameters are used in various end points. There are further endpoint-specific parameters, that are described in the OpenAPI specification.

- **{iri}**
  An IRI is a unique identifier for resources such as topics or metadata within the iiRDS package. IRIs enable the identification and retrieval of resources in a language-independent and globally consistent manner. Every resource should have an IRI to identify and retrieve the resource.
- **{apiversion}**
  The version of the API is a component that allows changes and improvements to be made to an API while ensuring backward compatibility for existing integrations. The API versioning follows a semantic versioning scheme consisting of a major version number, a minor version number and a patch number (e.g. v0.0.1) in accordance to prefixed “Semantic Versioning”.

3.4 Requirements

In iiRDS, it is allowed to use blank nodes as resources. For iirds:directoryNodes to be retrievable and requestable, it is necessary that IRIs are assigned to these nodes. As a fallback implementors can choose to generated IRIs in the case they were not present in the original iiRDS packages.

- Each directoryNode **MUST** have an IRI

3.5 Querying iiRDS Information

As iiRDS is an extensible standard that has specific extensions and is constantly being developed further, the API specification should be as universal as possible. Therefore, there are only some specific routes. Many complex use cases can be realized by the query endpoints:

`POST` `/api/{apiversion}/sparql` → SPARQL is a common standard to query graph-based data (see also https://www.w3.org/TR/sparql11-query/)

The request body includes SPARQL queries, limited to SELECT queries. Results are returned in the native format of the SPARQL query, which can be XML, JSON, CSV, or TSV based on the requester's accepted

format. Implementing this feature is optional, and implementations may support only certain result formats.

POST `/api/{apiversion}/resources?query` → The documentation to this iiRDS specific query language can be found in the OpenAPI specification.

This endpoint implements querying iiRDS RDF resources. There are two variants: “native query” and “iiRDS query”. Both return the results as JSON-LD.

- “native query” allows and implementor to provide his own query language as query interface. The query is provided as one string.
- “iiRDS query” is a new query language, defined in JSON with a schema. It is extensible and RDF aware. The iiRDS query supports hierarchical querying.

The iiRDS query language was designed in such a way that systems that are not natively based on RDF can also implement search mechanisms relatively easily, as the implementation of SPARQL is complicated.

The endpoint is designed to provide search functionality that is essential for typical iiRDS use cases. However, it also allows for full-text search in renditions, as well as paging and sorting of results. It should be noted that the endpoint does not claim to support all theoretically possible complex query use cases.

In the case of particularly complex searches, the use of multiple API calls may become necessary to achieve the desired outcome.

3.6 Error Handling

The API uses Standard HTTP status codes commonly used in APIs.

- 200 Successful operation.
- 400 Invalid parameters provided with request.
- 404 The requested resource could not be found.
- 500 Internal server error.

3.7 Example Requests

Use Case	What are the properties to the resource with the IRI `http://myCompany.com/versions/io_1/de/1`?
Request	GET `/api/{apiversion}/resources/http%3A%2F%2FmyCompany.com%2Fversions%2Fio_1%2Fde%2F1`

Result	{
	"@context": {
	"iirds": "http://iirds.tekom.de/iirds#",
	"iirdsMch": "http://iirds.tekom.de/iirds/domain/machinery#",
	"iirdsReq": "http://iirds.org/schema/iirds-request#",
	"iirdsSft": "http://iirds.tekom.de/iirds/domain/software#",
	"pifan": "https://www.i4icm.de/pifan#",
	"rdf": "http://www.w3.org/1999/02/22-rdf-syntax-ns#",
	"rdfs": "http://www.w3.org/2000/01/rdf-schema#",
	"vcard": "http://www.w3.org/2006/vcard/ns#"
	},
	"@graph": [
	{
	"@id": "http://myCompany.com/versions/io_1/de/1",
	"@type": "iirds:Topic",
	"iirds:dateOfCreation": {
	"@type": "xsd:dateTimeStamp",
	"@value": "2019-01-09T09:52:00+01:00"
	},
	"iirds:has-content-lifecycle-status": {
	"@id": "_:N89ddb41492b04fa998c2bfe727eceb69"
	},
	"iirds:has-rendition": {
	"@id": "_:N049bb0f348f745dc8098c2e045989a1d"
	},
	"iirds:has-topic-type": {
	"@id": "iirds:GenericTask"
	},
	"iirds:is-part-of-package": {
	"@id": "http://myCompany.com/iiRDS-parent"
	},
	"iirds:language": "en",
	"iirds:relates-to-component": [
	{
	"@id": "https://www.i4icm.de/pifan#PIFan"
	},
	{
	"@id": "https://www.i4icm.de/pifan#Rotor"
	}
	],
	"iirds:relates-to-product-lifecycle-phase": [
	{
	"@id": "iirds:Maintenance"
	},
	{
	"@id": "iirdsMch:Cleaning"
	}
	],
	"iirds:relates-to-product-variant": {
	"@id": "https://www.i4icm.de/pifan#X5-DH2"
	},
	"iirds:relates-to-qualification": [
	{
	"@id": "https://www.i4icm.de/pifan#ServiceTechnician"
	},

```
      },
      "iirds:is-applicable-for-document-type": [
        {
          "@id": "iirds:RepairInstructions"
        },
        {
          "@id": "iirds:OperatingInstructions"
        },
        {
          "@id": "iirds:MaintenanceInstructions"
        },
        {
          "@id": "iirds:QuickGuide"
        }
      ],
```

```
        {
          "@id": "https://www.i4icm.de/pifan#Operator"
        }
      ],
      "iirds:rights": "Content Copyright (c) 2015, PI-Fan
Project\niiRDS Implementation Copyright (c) 2019, iiRDS
Consortium\n\n\nTHE SOFTWARE IS PROVIDED \"AS IS\", WITHOUT
WARRANTY OF ANY KIND, EXPRESS OR\nIMPLIED, INCLUDING BUT NOT
LIMITED TO THE WARRANTIES OF MERCHANTABILITY,\nFITNESS FOR A
PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
THE\nAUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES
OR OTHER\nLIABILITY, WHETHER IN AN ACTION aOF CONTRACT, TORT OR
OTHERWISE, ARISING FROM,\nOUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN\nTHE SOFTWARE.",
      "iirds:title": "Cleaning the rotor"
    },
    {
      "@id": "_:N89ddb41492b04fa998c2bfe727eceb69",
      "@type": "iirds:ContentLifeCycleStatus",
      "iirds:dateOfStatus": {
        "@type": "xsd:dateTimeStamp",
        "@value": "2019-08-11T09:52:00+01:00"
      },
      "iirds:has-content-lifecycle-status-value": {
        "@id": "iirds:Reviewed"
      },
      "iirds:relates-to-party": {
        "@id": "http://myCompany.com/supplier/SupCo"
      }
    },
    {
      "@id": "_:N049bb0f348f745dc8098c2e045989a1d",
      "@type": "iirds:Rendition",
      "iirds:format": "application/xhtml+xml",
      "iirds:source":
"content/6_Maintenance/6_1_rotor_cleaning.xhtml"
    }
  ]
}
```


Use Case	How is the rotor cleaned by the X5-DH2?
Request	**POST** `/api/{apiversion}/resources?query`


```
"query": {
      "type": "iirdsquery",
      "context": {
        "iirds": "http://iirds.tekom.de/iirds#",
        "pifan": "https://www.i4icm.de/pifan#",
        "iirdsMch":
"http://iirds.tekom.de/iirds/domain/machinery#"
      },
      "resource": [
        {
          "type": [
            "iirds:InformationUnit"
          ],
          "asResult": true,
          "predicates": {
            "iirds:relates-to-product-variant": {
              "resource": [
                {
                  "id": "pifan:X5-DH2",
                  "asResult": false
                }
              ]
```

```
            },
            "iirds:relates-to-component": {
              "resource": [
                {
                  "id": "pifan:Rotor",
                  "asResult": false
                }
              ]
            },
            "iirds:relates-to-product-lifecycle-phase": {
              "resource": [
                {
                  "id": "iirdsMch:Cleaning",
                  "asResult": false
                }
              ]
            }
          }
        }
      ]
    }
```

Result


```
{
  "@context": {
    "iirds": "http://iirds.tekom.de/iirds#",
    "iirdsMch":
"http://iirds.tekom.de/iirds/domain/machinery#",
    "iirdsReq": "http://iirds.org/schema/iirds-request#",
    "iirdsSft":
"http://iirds.tekom.de/iirds/domain/software#",
    "pifan": "https://www.i4icm.de/pifan#",
    "rdf": "http://www.w3.org/1999/02/22-rdf-syntax-ns#",
    "rdfs": "http://www.w3.org/2000/01/rdf-schema#",
    "vcard": "http://www.w3.org/2006/vcard/ns#"
  },
  "@graph": [
    {
      "@type": "iirdsReq:QueryResults",
      "iirdsReq:count": "1",
      "iirdsReq:offset": "0",
      "iirdsReq:results": {
        "@type": "rdf:Seq",
        "rdf:_1": {
          "iirdsReq:resource": {
            "@id": "http://myCompany.com/versions/io_1/de/1"
          },
          "iirdsReq:score": "1.0"
        }
      }
    },
    {
      "@id": "http://myCompany.com/versions/io_1/de/1",
      "@type": "iirds:Topic",
      "iirds:dateOfCreation": {
        "@type": "xsd:dateTimeStamp",
        "@value": "2019-01-09T09:52:00+01:00"
      },
      "iirds:has-content-lifecycle-status": {
        "@id": "_:N89ddb41492b04fa998c2bfe727eceb69"
      },
      "iirds:has-rendition": {
        "@id": "_:N049bb0f348f745dc8098c2e045989a1d"
      },
      "iirds:has-topic-type": {
        "@id": "iirds:GenericTask"
      },
      "iirds:is-applicable-for-document-type": [
        {
          "@id": "iirds:RepairInstructions"
        },
        {
          "@id": "iirds:OperatingInstructions"
        },
        {
          "@id": "iirds:MaintenanceInstructions"
        },
        {
          "@id": "iirds:QuickGuide"
        }
      ],
      "iirds:is-part-of-package": {
        "@id": "http://myCompany.com/iiRDS-parent"
      },
      "iirds:language": "en",
      "iirds:relates-to-component": [
        {
          "@id": "https://www.i4icm.de/pifan#PIFan"
        },
        {
          "@id": "https://www.i4icm.de/pifan#Rotor"
        }
      ],
      "iirds:relates-to-product-lifecycle-phase": [
        {
          "@id": "iirds:Maintenance"
        },
        {
          "@id": "iirdsMch:Cleaning"
        }
      ],
      "iirds:relates-to-product-variant": {
        "@id": "https://www.i4icm.de/pifan#X5-DH2"
      },
      "iirds:relates-to-qualification": [
        {
          "@id":
"https://www.i4icm.de/pifan#ServiceTechnician"
        },
        {
          "@id": "https://www.i4icm.de/pifan#Operator"
        }
      ],
      "iirds:rights": "Content Copyright (c) 2015, PI-Fan
Project\niiRDS Implementation Copyright (c) 2019, iiRDS
Consortium\n\n\nTHE SOFTWARE IS PROVIDED \"AS IS\", WITHOUT
WARRANTY OF ANY KIND, EXPRESS OR\nIMPLIED, INCLUDING BUT NOT
LIMITED TO THE WARRANTIES OF MERCHANTABILITY,\nFITNESS FOR A
PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
THE\nAUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM,
DAMAGES OR OTHER\nLIABILITY, WHETHER IN AN ACTION aOF CONTRACT,
TORT OR OTHERWISE, ARISING FROM,\nOUT OF OR IN CONNECTION WITH
THE SOFTWARE OR THE USE OR OTHER DEALINGS IN\nTHE SOFTWARE.",
      "iirds:title": "Cleaning the rotor"
    },
    {
      "@id": "_:N89ddb41492b04fa998c2bfe727eceb69",
      "@type": "iirds:ContentLifeCycleStatus",
      "iirds:dateOfStatus": {
        "@type": "xsd:dateTimeStamp",
        "@value": "2019-08-11T09:52:00+01:00"
      },
      "iirds:has-content-lifecycle-status-value": {
        "@id": "iirds:Reviewed"
      },
      "iirds:relates-to-party": {
        "@id": "http://myCompany.com/supplier/SupCo"
      }
    },
    {
      "@id": "_:N049bb0f348f745dc8098c2e045989a1d",
      "@type": "iirds:Rendition",
      "iirds:format": "application/xhtml+xml",
      "iirds:source":
"content/6_Maintenance/6_1_rotor_cleaning.xhtml"
    }
  ]
}
```


4 Applied Terminology

The terminology utilized in the API specification is based on that of RDF. The metadata file in iiRDS is written in RDF, which results in the use of a standardized terminology. Furthermore, the terminology is also common in JSON-LD, as it also represents graph-based data.

The following terms are used frequently in the specification:

- A ***resource*** denotes a subject or object of an RDF statement. The term can be used synonymous with "entity" as it is also used in the RDF Semantics specification.
- A ***property*** is the predicate of an RDF statement. It covers all iirdsRelations and iirdsAttributes.

5 iiRDS-API Boilerplate

There are various applications that allow a REST API server hull to be generated based on an OpenAPI specification. Such a server hull implements the basic functionality of a REST API server with the specific routes/endpoints of the OpenAPI specification and serves as a starting point for implementing the logic of the corresponding endpoints. Depending on the application, various runtime environments and corresponding REST server frameworks are supported (e.g. NodeJS/Express, Go/Chi, Python/Flask, Java/JAX-RS etc.).

A server hulls can be generated using the Postman application (https://www.postman.com/) or swagger (https://editor-next.swagger.io/).

6 API Security

API security ensures secure access to your API endpoints while also maintaining usability.

The iiRDS Request specification purposely does not address authentication and authorization at all. It leaves it up to the implementor of the API. Thus, implementing security might add additional HTTP request headers, request parameters or requests requirements. It is recommended to just use HTTP header fields for the defined requests.

Here are some common approaches:

- OAuth 2.0: OAuth 2.0 is an industry-standard protocol for authorization. It enables third-party applications to access a user's data without exposing their credentials. OAuth 2.0 provides various grant types such as Authorization Code Grant, Implicit Grant, Client Credentials Grant, and Resource Owner Password Credentials Grant.
- API Keys: API keys are simple strings that clients include in their API requests to authenticate themselves. These keys are typically long, randomly generated strings that are associated with a specific user or application. API keys are straightforward to implement but should be used in conjunction with other security measures, as they can be more susceptible to abuse if leaked.
- HTTP Basic Authentication: This method involves including a username and password in the HTTP header of each request. While simple to implement, it is less secure than other methods because credentials are transmitted with each request. It is recommended to use HTTPS when using basic authentication to encrypt the credentials.
- Token-based authentication and authorization: In this approach, upon successful authentication, the server provides the client with a token (such as JWT or a session token). The client then includes this token in subsequent requests to authenticate itself. Tokens can have expiration times and can be invalidated if necessary.

It is recommended to choose either tokens or API keys mechanisms to keep the API simple in use and to transport the tokens or API keys via HTTP-header fields over HTTP to the server. Read https://swagger.io/docs/specification/authentication/ for hints about implementing API security.

When implementing API security, it's essential to consider factors such usability, scalability, and regulatory compliance. Additionally, regularly review and update your authentication mechanisms to address new threats and vulnerabilities.

7 Compliance

Following the guidelines and structures defined by the OpenAPI specification for designing and documenting the API ensures consistency and interoperability.

- All endpoints must be implemented to be compliant, only the optional features do not have to be implemented.
- The API must cover all requirements from the specification of the endpoints.
- It must be ensured that responses producing iiRDS RDF are compliant with the iiRDS specification.

By following these guidelines, you will achieve a compliant and standardized implementation.