INTERFACES FOR DIGITAL PRESERVATION SERVICES

v2.0.2

This document forms part of the Ministry of Education and Culture’s Open science and digital cultural heritage entity
Licence
Creative Commons Finland CC-BY-NC-SA 4.0 (http://creativecommons.org/licenses/by-nc-sa/4.0/deed.fi)

Users of this Specification are entitled to distribute the report, i.e. copy, circulate, display publicly and perform publicly the Specification and modify it under the following conditions:

- The Ministry of Education and Culture is appointed the Original Author (not, however, so that notification would refer to a licensee or means by which the Specification is used as supported by the licensor).
- The user is not entitled to use the Specification commercially.
- If the user makes any modifications to the Specification or uses it as the basis for their own works, the derivative work shall be distributed in the same manner or under the same type of licence.
This is an unofficial translation of national Interfaces for Digital Preservation Services specification. Should this translation conflict with the Finnish version, the Finnish version will have higher priority.
1 INTRODUCTION

This specification defines the interfaces of the Finnish national digital preservation (DP) services. Partner organizations can utilize these interfaces for example when they use ingest and dissemination functionalities of the service. The main purpose of these interfaces is to enable machine-to-machine communication. The interfaces are designed to be as simple as possible, but the application architecture is scalable, which means that new features can be added as required.

1.1 Digital Preservation Services

DP services refer to the national services produced for the digital preservation of cultural heritage resources and research data together.

The DP service produced for cultural heritage resources, guarantees the preservation of essential national information resources held in libraries, archives and museums. Digital cultural heritage resources cover both digitized and born-digital information resources: publications falling within the bounds of legal deposit, government publications belonging to the national cultural heritage, and other digital information resources worth preserving created by organizations operating under the Ministry of Education and Culture.

The DP service produced for research data ensures the availability and preservation of digital research resources. This DP service supports a permanent and coordinated approach to support the management of research resources. The aim is to ensure the verifiability and repeatability of research at various stages of the life cycle and to make the results easy to use. This enables that research results can be reused, evaluated, utilized in decision-making and secured by increasing digit data for future generations of researchers.

1.2 Interfaces of Digital Preservation Services

The content to be ingested must be meet the requirements of both the Metadata Requirements and Preparing Content for Digital Preservation [META] specification and the File Formats [TDSTO] specification. Well-formed Submission Information Packages (SIPs) can be sent to the DP service using an SFTP connection [IETF_SFTP] and the interface defined in this specification. The DP service generates validation reports in both HTML and PREMIS formats. The reports can be retrieved by the same SFTP connection.

Search and retrieval of Dissemination Information Packages (DIPs) is based on the HTTP protocol [RFC_7230]. This specification defines the REST queries and HTTP responses required for these transactions. The syntax of the exchanged messages is based on JSON [RFC_7159]; the preserved content (both data and metadata about it) is delivered in the requested format. DIPs are delivered as a binary ZIP or TAR packages and metadata in an XML format.

If any errors or other problems take place during transactions, the codes from the IETF memo [IETF_SFTP] are used for the SFTP protocol and HTTP status codes [RFC_7231] for the REST interface. Best current practice recommendations in the usage of these specifications should be taken into account, unless otherwise required in this specification.

1.3 Use Cases

This section describes two common use cases: transfer of SIPs into the DP services and retrieval of DIPs from there. The purpose of the use cases is to give an overview of the interfaces. The use cases are simple examples and are not intended to describe the interface functionality exhaustively.

1.3.1 Transfer of SIPs into the DP Services

Transfer of SIPs into the DP service and the subsequent ingest of these packages is depicted in Figure 1. Firstly, the user opens a SFTP connection to the DP service by logging into the DP SFTP server and sends the SIPs – which have been are created according to packaging specification [META, TDSTO] – to the server. In
Figure 1, three SIPs are transferred. The SFTP protocol acknowledges the transfers (2). Validation of the SIPs in the DP takes some time. After the content has been processed and transferred to preservation in the DP service, the user receives a validation report to his/her home directory. If the validation fails, the user receives a failure report. The user can utilize SFTP operations for search, retrieval, and deletion of validation reports. In Figure 1, reports are fetched when two SIPs have been processed and one is still in the queue. As a response, the user gets reports related to those packages that have been dealt with (4). Following the ingest of the last SIP, even the third report becomes available and can be retrieved (5, 6). The SFTP interface and SIP ingestion process are specified in more detail in Section 2 and it is used according to the SFTP protocol specification [IETF_SFTP].

![Diagram](image1.png)

**Figure 1:** Transfer of SIPs into the DP service and retrieval of validation reports

### 1.3.2 Retrieval of DIPs from the DP services

Retrieval of DIPs from the DP services is described in Figure 2. The user searches for some previously ingested content (1) and in response receives (2) a list of preserved content (A), as specified in Section 4.2.1. Then, the user selects the relevant content (or part of it) and makes a request to create a DIP (3). The user gets a response (3) that a process for creating a DIP has been started, but the DIP is not yet available (4) since creating them may take some time. With this response, the user will however get the future location of the forthcoming DIP (B). Should the user try to retrieve the DIP before it is ready (5), the server will respond with a Not Found message (6). When the DIP is available, the user can download it (7). In the response (8), the user gets an OK message with the DIP (C). Retrieval and creation of DIPs using the REST interface [RFC_7231] over HTTP protocol [RFC_7230] is specified detailed in Section 3.

![Diagram](image2.png)

**Figure 2:** Fetching content and DIP from the DP service.
1.4 Maintenance of this Specification

This specification will be updated annually according to the needs of the partner organizations. Existing functionalities can be enriched by using more comprehensive message structures. Further, new functionalities such as, for example, requests to update the preserved content, retrieval of validation reports using the REST interface, or hiding/deleting preserved content, may be implemented.

The three-level version number of the definition describes the level of changes that have been made to the specification. If the first digit is updated, it means significant changes to partner organizations’ systems that are integrated to the DP service. The middle digit reflects situations where changes may affect partner organizations’ systems. An update to the third digit describes situations where only minor corrections or clarifications have been made. For example, updates to examples in the specification are minor corrections.

The REST interface specified in this document is not compatible with the corresponding specification versions 1.0.X.
2 TRANSFER OF SIP PACKAGES TO DP SERVICES

An SFTP interface [IETF_SFTP] is used for transfer of DIPs into DP services and retrieval of validation reports from these services. SFTP enables secure communication. The SFTP interface can also be applied for retrieval of DIPs.

2.1 Authentication and Monitoring

SFTP connection uses public key authentication of users based on the guidelines given to producers as a part of the DP service deployment process. A user can only access his own SIPS and DIPS; he cannot modify other users’ archived content or change its ownership. For each SFTP connection, the following information will be logged:

- Who did? (userid)
- What was done (commands, bytes sent/received)
- When? (timestamps)
- From where? (IP address)

2.2 User Directories

Each user has the following directory tree under his home directory:

- /accepted – Directory for validation reports of accepted SIPS.
- /rejected – Directory for rejected SIPS and their validation reports.
- /transfer – Directory for SIPS to be ingested.
- /disseminated – Directory for DIPS (cf. Section 3.4.2)

Validation reports will remain in the user’s home directory for at least 20 days, and DIPS and rejected SIPS for at least ten (10) days, unless the user deletes the package earlier. More detailed information regarding DIPS is given in Section 3.5.

2.3 Transfer to the DP Services

A user sends SIPS to his transfer directory (/transfer) using the SFTP protocol.

When the ingest process of a SIP begins, the package is automatically transferred from the /transfer directory for validation. If the information content of the SIP is approved, it is transferred to preservation. The user gets a validation report to his /accepted directory. If the content is rejected, the validation report is placed in the /reject directory. Rejection process is discussed in greater detail in Section 2.4. The validation report includes:

- The user (user name of the user)
- Date and time of the beginning of the validation process
- The structure of the SIP
- Validation actions and results, validated targets, and timestamps
- Date and time of the end of the validation process (which is also the time when the DP service took the responsibility of preserving the content)

The user gets two files: a detailed validation report in PREMIS format and a summary in HTML format. If the SIP has passed validation, the directory tree of these reports is:

/accepted/<date>/<transfer>/transfer/<uuid>-ingest-report.xml
/accepted/<date>/<transfer>/transfer/<uuid>-ingest-report.html

The file path parts are as follows:
2.4 Content Rejected in Ingest

If a SIP is rejected in ingest, the package is transferred to the user’s /rejected directory. The directory tree is as follows:

```
/rejected/<date>/transfer/<transfer>-<uuid>/
```

And the corresponding validation reports are:

```
/rejected/<date>/transfer/<transfer>-<uuid>-ingest-report.xml
/rejected/<date>/transfer/<transfer>-<uuid>-ingest-report.html
```

The qualifiers are as follows:

- `<date>` – Date (year-month-day) when the report was delivered to the user.
- `<transfer>` – The SIP’s filename.
- `<uuid>` – Unique identifier for separating different transfers.
- `xml` – File extension for a PREMIS report.
- `html` – File extension for a HTML summary.

The SIP in the /rejected directory can be repaired without sending the replacement files over the SFTP interface again. Erroneous files in the SIP can be removed and replaced with new ones. When the content has been fixed and the modified SIP has been signed, the user can move the package from the /rejected directory to the /transfer directory using the SFTP rename command. Then the ingest process for the modified SIP starts as described in Section 2.3.
3 MANAGING AND RETRIEVING CONTENTS

A REST interface over HTTP protocol is used for search, management, and retrieval of archived content.

The following qualifiers are used in the REST messages:

- `<base>` – String: https://pas.csc.fi/api/2.0
- `<contract>` – Contract identifier which restricts the scope to the content to a particular DP service contract.
- Commands may have qualifiers which are defined below, in that command’s detailed specification.

3.1 Authentication and Management

Authentication uses both TSL/SSL and HTTP Basic Access authentication. An authenticated user can use REST messages only authorized contract identifiers. The DP services records a log of the user’s actions so that for each REST query message at least the following information is stored automatically:

- Who made the request? (userid)
- What was done? (the content of REST query message)
- Date and time of the transaction (timestamp of a query)
- From where? (source IP address of the message)
- What was the response? (HTTP status code and the size of the response)

3.2 Message Structure

3.2.1 Query Messages

Query messages have the format:

{GET, POST} <base>/<contract>/<term>...

where `<term>` describes exiting resource or tool, and it is one of the following:

- `search` – Search too. Functionality in the interface is described in Section 3.3.
- `preserved` – AIP resource. Functionality in the interface is described in Section 3.4.
- `disseminated` – DIP resource. Functionality in the interface is described in Section 3.5.

In the future, other terms, such as `statistics`, can be added.

Query message format in this specification is given without optional HTTP headers. These headers can be added to any massage according to the RFC7231 specification. For example, to a message:

GET https://pas.csc.fi/api/2.0/csc

can be added headers as follows:

GET https://pas.csc.fi/api/2.0/csc HTTP/1.1
Date: Mon, 13 Nov 2017 08:12:31 GMT
Host: pas.csc.fi
...

3.2.2 Response Messages

The JSON format [RFC_7159] with the JSend structure [JSEND] is used in a response message content. In a successful case, a response message structure is as follows:

{
   "status": "success",
   "data": { ... }
}
In the response messages, status tells about the success or failure of the query, and data the result set. In the unsuccessful case, the following message is given as a response message:

```json
{
    "status": "fail",
    "data": { "message": "..." }
}
```

In the case of failure, message gives the error message. However, if the query parameter has been incorrect, the error message will be returned with the parameter name:

```json
{
    "status": "fail",
    "data": { "<parameter>": "..." }
}
```

For example, if there was a parameter limit in the query and an incorrect value has been given, the message could be as follows:

```json
{
    "status": "fail",
    "data": { "limit": "Value can only be an integer in range 1-1000" }
}
```

If a user attempts to manage any preserved content to which he is not entitled to access, the code "404 Not Found" instead of "403 Forbidden" in the RFC 7231 specification is returned.

The returned preserved content is a ZIP or TAR dissemination package, or an XML metadata document.

### 3.3 Search Messages

In DP systems, certain content is stored in one or more archival information packages. Each archival information package includes a METS document describing the content of the package which is indexed to a separate database. The METS documents of existing dissemination information packages are also indexed. With the search functionality, it is possible to search for archival information and dissemination information packages by doing searches in this database and fetch a list of the information packages matching to given search conditions. If identifiers of archival information and dissemination information is known before, the identifiers can be used directly, and this search function can be bypassed. Updating the preserved content does not change already existing archival information packages, but new archival information packages are created for updated versions. This is how the search function and the identifiers returned make it possible to distinguish between the different versions of the content.

**Message format:**

```
GET <base>/<contract>/search?<parameters>
```

**Message qualifiers:**

- `<parameters>` – Zero, one or more of the following parameters separated with & characters.
  
  `q=<query>`

  The search criteria used for searching result set. The search criteria is presented separately in Section 3.3.1. The default is the search for all the content. Note that the generated search criteria must be URL encoded for the REST interface.

  `limit=<limit>`

  The parameter allows the results to be paginated with the page parameter, where `<limit>` tells how many items (at most) are returned per page, and the page tells the page number. The default value is 20. Integers between 1 and 1000 are permitted.
The parameter allows the results to be paginated with the limit parameter, where the limit tells how many items (at most) are returned per page, and <page> tells the page number. The default value is 1, which corresponds to the first page.

Response message in normal cases:

- HTTP 200 OK – Successful operation. The response contains a list of found information packages in JSON format. The preserved content can be retrieved using the URLs in the message.
- HTTP 400 Bad Request – Syntax error in a command. For example, incorrect date in the period parameter.
- HTTP 404 Not Found – The preserved content was not found (e.g. incorrect contract identifier).
- HTTP 405 Method Not Allowed – Wrong method in a message (other than GET).

The format of a response message to a successful query is as follows:

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 2345
Allow: GET
Date: Mon, 13 Nov 2017 10:42:37 GMT

{
  "status": "success",
  "data": {
    "results": [
      {
        "location": "<base>/<contract>/preserved/<aip_id-1>",
        "createdate": "<aip_create_date-1>",
        "lastmoddate": "<dip_lastmoddate-1>",
        "match": <search_match-obj-1>,
        "id": "<aip_id-1>",
        "pkg_type": "AIP"
      },
      {
        "location": "<base>/<contract>/disseminated/<dip_id-1>",
        "createdate": "<dip_create_date-1>",
        "lastmoddate": "<dip_lastmoddate-1>",
        "match": <search_match-obj-2>,
        "id": "<dip_id-1>",
        "pkg_type": "DIP"
      },
      ...
    ],
    "links": {
      "self": "<current-url>",
      "next": "<next-url>",
      "previous": "<previous-url>"
    }
  }
}
```

The structure of the response messages of the search function has been searched from the structure of the JSON API interface [JSON_API] and the Invenio repository application [INVENIO]. In the response, the results is the result set, and the links is the paging function. The result set has zero, one or more AIP or DIP result entries. A result entry contains the following information:

- location – Address, that can be used for managing AIP or DIP (cf. Sections 3.4 and 3.5).
- **createdate** – The value of the CREATEDATE attribute in the METS document.
- **lastmoddate** – The value of the LASTMODDATE attribute in the METS document. Given, if it exists.
- **match** – Search criteria match, which gives information in which part of the METS database the search criteria was matched.
- **id** – Information package identifier.
- **pkg_type** – Term AIP refers to archival information package and DIP refers to dissemination information package.

The paging function contains the following information:
- **self** – Current address.
- **next** – Next page. Given if the next page exists.
- **previous** – Previous page. Given if previous page exists.

In a result entry, **createdate** and **lastmoddate** are intended to describe the content version to which the information package belongs. Further information can be added later if needed.

### 3.3.1 Search Function

The METS documents of the archival information packages and existing dissemination information packages are fully indexed to the METS database and their data can be retrieved as described in this section. The search criteria uses the Apache Lucene syntax [LUCENE], which allows a variety of functions. The search always uses the key:value pairs where the key corresponds to a particular element or attribute in the METS document and the value of the value in that key.

The search database is not real-time. Archival and dissemination information packages must first be indexed so they can be searched. If the preservation responsibility of an archival information package has recently been changed, or the dissemination information package is recently created, it may not have been indexed yet to the database and the package cannot be found in the search function. Likewise, when a dissemination information package is removed, its data will be cleared from the search database with a delay. However, this delay does not affect other interface functions.

#### 3.3.1.1 Search Functionality and Syntax

In the search functionality, the functions are available using the Apache Lucene syntax [LUCENE] in accordance with the luqum application library [LUQUM] including, for example, the following functionalities:

- logical operations AND, OR, and NOT
- fuzzy search
- proximity search
- wildcards (?) and *)
- range search
- boosting a term

#### 3.3.1.2 Key/Value Pairs

In a search criteria, the key name matches a particular element or attribute in the METS document. Value is something that this element or attribute should contain. The XML path can also be used as the key, and the names of the elements/attributes are described as underscored without namespaces. However, the whole path is not mandatory, but only the remainder is enough. Values are handled case insensitive. For example:

---

1 See instructions and examples to use the search terms syntax at [https://lucene.apache.org/core/3_6_0/queryparsersyntax.html](https://lucene.apache.org/core/3_6_0/queryparsersyntax.html)
- q='mets_OBJID:id-123' – Get information about packages that have a <mets> element in the METS document with the @OBJID attribute value id-123. In this example, the key is mets_OBJID.
- q='mets_dmdSec_mdWrap_xmlData_subject:xxx' – Get information about packages that have in the METS document path <mets>/<dmdSec>/<mdWrap>/<xmlData>/<subject>, where the <subject> element contains the value xxx. The namespace in the XML structure may change in the middle of the path. In practice, the descriptive metadata is requested from the <subject> root element. In supported descriptive metadata formats this is only possible in the Dublin Core format.
- q='subject:xxx' – Get information about packages that have a value of xxx in any <subject> element or any @subject attribute in the METS document. Of the supported metadata formats, this also takes into account <subject> elements of the EAD and MODS formats, in addition to the Dublin Core. Since the full path is not given, the key may be anywhere in the METS document.

### 3.3.1.3 Restricting the Type of an Information Package

Using the AND-operator, pkg_type:AIP or pkg_type:DIP should be added to the search criteria, depending on whether the search should be applied only to archival information packages (AIP) or dissemination information packages (DIP). Rules of the closure mark should be taken into account. By default, both resources are searched.

### 3.4 Archival Information Package Management

Currently, the interface allows only one archival information package to be managed at a time and in full. If, in the future, there is a sufficient need for functionality that require the management of multiple archival information packages or their parts, the necessary functions may be added alongside this.

#### 3.4.1 Commands for Archival Information Package

This function provides a list of the commands that can be applied to the selected archival information package. Currently, the only command is to create a dissemination information package.

**Message format:**

GET <base>/<contract>/preserved/<aip-id>

**Message qualifiers:**

- `<aip-id>` – An archival information package identifier, based on which the query is limited to the preserved content. Section 3.3 describes how identifiers can be searched

**Response message in normal cases:**

- HTTP 200 OK – Successful message. The response contains a list of all versions in JSON format.
- HTTP 404 Not Found – The content was not found.
- HTTP 405 Method Not Allowed – Wrong method in a message (other than GET).

**The format of a response message to a successful query is as follows:**

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 2345
Allow: GET
Date: Mon, 13 Nov 2017 10:42:37 GMT
{
  "status": "success",
  "data": {
    "disseminate": ":<base>/<contract>/preserved/<aip-id>/disseminate"
  }
}
```
Returned URL syntax is described in Section 3.4.2.

### 3.4.2 Creation of Dissemination Information Packages

This function is used to create a DIP from an AIP. Section 3.5 describes in detailed.

**Message format:**

```
POST <base>/<contract>/preserved/<aip-id>/disseminate?<parameters>
```

**Message qualifiers:**

- `<aip-id>` – An archival information package identifier, based on which a DIP is created from preserved content. Section 3.3 describes how identifiers can be searched.
- `<parameters>` – Zero, one or more of the following parameters separated with & characters.
  
  - `catalog=<catalog>`
    
    The dissemination information package is created in accordance with the schema catalog version `<catalog>`. Only the first two digits of the version number is given as a parameter. For example, `catalog = 1.6` returns the results according to the schema catalog 1.6.X, where X is the most recent version of the 1.6 version series. A dissemination information package cannot be created using an older catalog version. The default is the latest version.
  
  - `format=<format>`
    
    The dissemination information package is wrapped in a file format `<format>`. The allowed values are `zip` and `tar`. By default, ZIP compression is used.

**Response message in normal cases:**

- HTTP 202 Accepted – Successful operation. The content was found and the creation of a DIP has started, but the process is not yet complete. The response message contains already a link to the DIP to be created. In addition to the REST interface, the DIP, when completed, will also be available via SFTP connection from the user’s `/disseminated` directory (cf. Section 2.2).
- HTTP 400 Bad Request – Incorrect specification version.
- HTTP 404 Not Found – The searched content was not found.
- HTTP 405 Method Not Allowed – Wrong method in the message (other than POST).

The format of a response message to a successful query is as follows:

```
HTTP/1.1 202 Accepted
Content-Type: application/json
Location: <base>/<contract>/disseminated/<dip-id>
Allow: POST
Date: Mon, 13 Nov 2017 10:42:37 GMT
{
    "status": "success",
    "data": {
        "disseminated": "<base>/<contract>/disseminated/<dip-id>"
    }
}
```

In the response message `<dip-id>` is dissemination package identifier. Returned URL syntax is described in Section 3.4.2.

### 3.5 Dissemination Information Packages

When content is retrieved from the DP system, a dissemination information package must be first created (cf. Section 3.4.2). The dissemination information package is created to conform to the metadata
requirements and preparing content for digital preservation specification [META]. A new identifier will be given to the package and a new METS document will be created. Finally, the integrity of the package is verified and then it will be given available. The DIP can be fetched either using the REST or SFTP interface. Retrieving content does not remove it from the preservation.

### 3.5.1 Tracking a DIP and Commands to DIPs

This function allows keeping track on the DIP creation process. This function also provides a list of the commands that can be applied to the dissemination information package.

**Message format:**

```
GET <base>/<contract>/disseminated/<dip-id>
```

**Message qualifiers:**

- `<dip-id>` – Identifier of the DIP. It is the identifier that is received in the DIP creation query message (cf. Section 3.4.2). Section 3.3 describes how DIPs can be searched.

**Response message in normal cases:**

- HTTP 200 OK – Successful operation. The response contains commands to a DIP in JSON format.
- HTTP 404 Not Found – The searched content was not found.
- HTTP 405 Method Not Allowed – Wrong method in a message (other than GET).

The format of a response message to a successful query is as follows:

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 2345
Allow: GET
Date: Mon, 13 Nov 2017 10:42:37 GMT
{
  "status": "success",
  "data": {
    "complete": <complete>,
    "actions": <actions>
  }
}
```

The response message `<complete>` is one of the following: "false" (DIP creation is in progress) or "true" (DIP is ready). The in the message, `<actions>` is a JSON structure of the functions that can be performed on dissemination information packages. When the dissemination information package creation is still in progress, the structure is empty `{}`. When the dissemination information package is complete, the structure is as follows:

```
{
  "download": "<base>/<contract>/disseminated/<dip-id>/download",
  "metadata": "<base>/<contract>/disseminated/<dip-id>/metadata",
  "history": "<base>/<contract>/disseminated/<dip-id>/history"
}
```

The given functions are available immediately. Returned URL syntax is described in Section 3.4.2.

### 3.5.2 Retrieval of DIPs

This function allows retrieving created dissemination information packages or their metadata. Dissemination information packages can also be downloaded from the user's home directory /disseminated directory.
via the SFTP interface. Created dissemination information packages will remain ten (10) days unless the user removes them previously from his home directory.

Message format:

GET <base>/<contract>/disseminated/<dip-id>/download
GET <base>/<contract>/disseminated/<dip-id>/metadata
GET <base>/<contract>/disseminated/<dip-id>/history

Message qualifiers:

- `<dip-id>` – The identifier of the DIP, which the user received in the DIP creation message (cf. Section 3.4.2). Created DIPs can be searched as described in Section 3.3.
- `download` – The whole DIP is delivered. The DIP may also be retrieved directly from the user’s /disseminated directory using the SFTP interface.
- `metadata` – Only the METS document is fetched.
- `history` – The whole provenance information of the package is fetched in PREMIS format.

Response message in normal cases:

- HTTP 200 OK – Successful operation. DIP, or its METS document, or its provenance report is included in the response message. The DIP is returned as a ZIP or TAR package and its metadata as an XML document. A JSON formatted message is not returned.
- HTTP 404 Not Found – The DIP was not found (either it has not been created or creation is in progress).
- HTTP 405 Method Not Allowed – Wrong method in a message (other than GET).

The format of a response message to a successful query is as follows:

- Dissemination information package in a ZIP format:
  
  HTTP/1.1 200 OK
  Content-Type: application/zip
  Content-Length: 23456
  Allow: GET
  Date: Mon, 13 Nov 2017 10:42:37 GMT

  <dip.zip>

- Dissemination information package in a TAR format:
  
  HTTP/1.1 200 OK
  Content-Type: application/x-tar
  Content-Length: 23456
  Allow: GET
  Date: Mon, 13 Nov 2017 10:42:37 GMT

  <dip.tar>

- METS document of the dissemination information package:
  
  HTTP/1.1 200 OK
  Content-Type: text/xml
  Content-Length: 2345
  Allow: GET
  Date: Mon, 13 Nov 2017 10:42:37 GMT

  <mets...
- The provenance report of the dissemination information package:
  
  HTTP/1.1 200 OK
  Content-Type: text/xml
  Content-Length: 2345
  Allow: GET
  Date: Mon, 13 Nov 2017 10:42:37 GMT

  <premis...>

### 3.6 Public Keys

This functional can be used to retrieve the public part of the key pair used in the digital signature of the dissemination information package. The key can be used to verify that the dissemination information package has arrived for the DP service and that it is secure.

**Message format:**

- GET <base>/public_key/dip

**Response message in normal cases:**

- HTTP 200 OK – Successful operation. The public key for the DIP digital signature is returned in the response message.
- HTTP 405 Method Not Allowed – Wrong method in a message (other than GET).
ANNEX A. SIP VALIDATION REPORT

The validation report contains a set of objects, events, and agents as specified in PREMIS Data Dictionary, in PREMIS XML format. This Annex contains brief descriptions of the relevant PREMIS properties. The validation report contains only single occurrence of each property, unless stated otherwise. The event name, result and filenames of objects related to the event are shown in a separate human readable HTML formatted report.

A.1. Objects

PREMIS XML formatted statements are created for all entities employed and created during the ingest process. These entities are PREMIS objects, which are processed in events (cf. Section A.2). These events in turn and performed by various agents (cf. Section A.3).

Entities are submission information package, digital signature, METS document, digital object, and created archival information package. This section describes the content of these PREMIS objects.

Submission information package:

<table>
<thead>
<tr>
<th>Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;object&gt;</td>
<td>SIP in a validation report</td>
</tr>
<tr>
<td>@xsi:type</td>
<td>“representation” (type of a PREMIS object)</td>
</tr>
<tr>
<td>&lt;&lt;objectIdentifier&gt;</td>
<td>SIP identifier in a validation report</td>
</tr>
<tr>
<td>&gt;&gt;&lt;&lt;objectIdentifierType&gt;</td>
<td>“preservation-sip-id” (type of the identifier)</td>
</tr>
<tr>
<td>&gt;&gt;&lt;&lt;objectIdentifierValue&gt;</td>
<td>Value of the identifier</td>
</tr>
<tr>
<td>&lt;&lt;originalName&gt;</td>
<td>The original ZIP or TAR filename of the SIP</td>
</tr>
<tr>
<td>&lt;&lt;environment&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;&lt;dependency&gt;</td>
<td>Dependency to a SIP</td>
</tr>
<tr>
<td>&lt;&lt;&lt;dependencyIdentifier&gt;</td>
<td>SIP identifier in the METS document</td>
</tr>
<tr>
<td>&gt;&gt;&gt;&lt;&lt;dependencyIdentifierType&gt;</td>
<td>“mets:OBJID”</td>
</tr>
<tr>
<td>&gt;&gt;&gt;&lt;&lt;dependencyIdentifierValue&gt;</td>
<td>Value of the SIP identifier</td>
</tr>
</tbody>
</table>

Digital signature of a SIP:

<table>
<thead>
<tr>
<th>Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;object&gt;</td>
<td>The digital signature file of a SIP in a validation report</td>
</tr>
<tr>
<td>@xsi:type</td>
<td>“representation” (type of a PREMIS object)</td>
</tr>
<tr>
<td>&lt;&lt;objectIdentifier&gt;</td>
<td>The digital signature file identifier in a validation report</td>
</tr>
<tr>
<td>&gt;&gt;&lt;&lt;objectIdentifierType&gt;</td>
<td>“preservation-signature-id” (type of the identifier)</td>
</tr>
<tr>
<td>&gt;&gt;&lt;&lt;objectIdentifierValue&gt;</td>
<td>Value of the identifier</td>
</tr>
<tr>
<td>&lt;&lt;originalName&gt;</td>
<td>Filename of the digital signature</td>
</tr>
<tr>
<td>&lt;&lt;relationship&gt;</td>
<td>Relationship to a SIP</td>
</tr>
<tr>
<td>&gt;&gt;&gt;&lt;&lt;relationshipType&gt;</td>
<td>“structural” (structural relationship)</td>
</tr>
<tr>
<td>&gt;&gt;&gt;&lt;&lt;relationshipSubType&gt;</td>
<td>“is included in”</td>
</tr>
<tr>
<td>&gt;&gt;&gt;&lt;&lt;relatedObjectIdentification&gt;</td>
<td>SIP identifier in a validation report</td>
</tr>
<tr>
<td>&gt;&gt;&gt;&lt;&lt;relatedObjectIdentifierType&gt;</td>
<td>“preservation-sip-id” (type of the SIP identifier)</td>
</tr>
<tr>
<td>&gt;&gt;&gt;&lt;&lt;relatedObjectIdentifierValue&gt;</td>
<td>Value of the SIP identifier</td>
</tr>
</tbody>
</table>

METS document:

<table>
<thead>
<tr>
<th>Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;object&gt;</td>
<td>METS document in a validation report</td>
</tr>
<tr>
<td>@xsi:type</td>
<td>“representation” (type of a PREMIS object)</td>
</tr>
<tr>
<td>&lt;&lt;objectIdentifier&gt;</td>
<td>METS document identifier in a validation report</td>
</tr>
</tbody>
</table>
Digital object: The report contains a separate PREMIS object for each digital object.

<table>
<thead>
<tr>
<th>Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;object&gt;</code></td>
<td>Digital object in a validation report</td>
</tr>
<tr>
<td>@xsi:type</td>
<td>“representation” (type of a PREMIS object)</td>
</tr>
<tr>
<td><code>&lt;objectIdentifier&gt;</code></td>
<td>Digital object identifier in a validation report</td>
</tr>
<tr>
<td><code>&lt;objectIdentifierType&gt;</code></td>
<td>“preservation-object-id” (type of the identifier)</td>
</tr>
<tr>
<td><code>&lt;objectIdentifierValue&gt;</code></td>
<td>Value of the identifier</td>
</tr>
<tr>
<td><code>&lt;originalName&gt;</code></td>
<td>Digital object file name</td>
</tr>
<tr>
<td><code>&lt;environment&gt;</code></td>
<td></td>
</tr>
<tr>
<td><code>--&lt;dependency&gt;</code></td>
<td>Dependency to the digital object</td>
</tr>
<tr>
<td><code>--&lt;dependencyIdentifier&gt;</code></td>
<td>Digital object identifier in the METS document</td>
</tr>
<tr>
<td><code>--&lt;dependencyIdentifierType&gt;</code></td>
<td>Type of the digital object identifier</td>
</tr>
<tr>
<td><code>--&lt;dependencyIdentifierValue&gt;</code></td>
<td>Value of the digital object identifier</td>
</tr>
<tr>
<td><code>&lt;relationship&gt;</code></td>
<td>Relationship to a SIP</td>
</tr>
<tr>
<td><code>--&lt;relationshipType&gt;</code></td>
<td>“structural” (structural relationship)</td>
</tr>
<tr>
<td><code>--&lt;relationshipSubType&gt;</code></td>
<td>“is included in”</td>
</tr>
<tr>
<td><code>--&lt;relatedObjectIdentification&gt;</code></td>
<td>SIP identifier in a validation report</td>
</tr>
<tr>
<td><code>--&lt;relatedObjectIdentifierType&gt;</code></td>
<td>“preservation-sip-id” (type of the SIP identifier)</td>
</tr>
<tr>
<td><code>--&lt;relatedObjectIdentifierValue&gt;</code></td>
<td>Value of the SIP identifier</td>
</tr>
</tbody>
</table>

Archival information package:

<table>
<thead>
<tr>
<th>Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;object&gt;</code></td>
<td>SIP in a validation report</td>
</tr>
<tr>
<td>@xsi:type</td>
<td>“representation” (type of a PREMIS object)</td>
</tr>
<tr>
<td><code>&lt;objectIdentifier&gt;</code></td>
<td>SIP identifier in a validation report</td>
</tr>
<tr>
<td><code>&lt;objectIdentifierType&gt;</code></td>
<td>“preservation-aip-id” (type of the identifier)</td>
</tr>
<tr>
<td><code>&lt;objectIdentifierValue&gt;</code></td>
<td>Value of the identifier</td>
</tr>
<tr>
<td><code>&lt;originalName&gt;</code></td>
<td>SIP file name</td>
</tr>
<tr>
<td><code>&lt;relationship&gt;</code></td>
<td>Relationship to a SIP</td>
</tr>
<tr>
<td><code>--&lt;relationshipType&gt;</code></td>
<td>“derivation”</td>
</tr>
<tr>
<td><code>--&lt;relationshipSubType&gt;</code></td>
<td>“has source”</td>
</tr>
<tr>
<td><code>--&lt;relatedObjectIdentifier&gt;</code></td>
<td>SIP identifier in a validation report</td>
</tr>
<tr>
<td><code>--&lt;relatedObjectIdentifierType&gt;</code></td>
<td>“preservation-sip-id” (type of the SIP identifier)</td>
</tr>
<tr>
<td><code>--&lt;relatedObjectIdentifierValue&gt;</code></td>
<td>Value of the SIP identifier</td>
</tr>
</tbody>
</table>
A.2. Events

PREMIS XML formatted statements are created of actions performed in ingest. These are called PREMIS events. Ingest actions are targeted to different objects (cf. Section A.1) and performed by different agents (cf. Section A.3). This section describes the content of all relevant PREMIS events.

The table below describes core elements which are shared by all events. After the table there is a list of all ingest events and further details about some of the core elements.

Core elements:

<table>
<thead>
<tr>
<th>Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;event&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;-&lt;eventIdentifier&gt;</td>
<td>Identifier of an event</td>
</tr>
<tr>
<td>--&lt;eventIdentifierType&gt;</td>
<td>Type of the identifier</td>
</tr>
<tr>
<td>--&lt;eventIdentifierValue&gt;</td>
<td>Value of the identifier</td>
</tr>
<tr>
<td>&lt;-&lt;eventType&gt;</td>
<td>Type of the event</td>
</tr>
<tr>
<td>&lt;-&lt;eventDateTime&gt;</td>
<td>Timestamp of the event</td>
</tr>
<tr>
<td>&lt;-&lt;eventDetail&gt;</td>
<td>Name of the event</td>
</tr>
<tr>
<td>&lt;-&lt;eventOutcomeInformation&gt;</td>
<td></td>
</tr>
<tr>
<td>--&lt;eventOutcome&gt;</td>
<td>Result of the event: success or failure</td>
</tr>
<tr>
<td>--&lt;eventOutcomeDetail&gt;</td>
<td></td>
</tr>
<tr>
<td>---&lt;eventOutcomeDetailNote&gt;</td>
<td>Additional information of the event result in text format. May be used in any event, unless otherwise stated.</td>
</tr>
<tr>
<td>---&lt;eventOutcomeDetailExtension&gt;</td>
<td>Additional information of the event result in XML format. May be used in any event, unless otherwise stated.</td>
</tr>
<tr>
<td>&lt;-&lt;linkingAgentIdentifier&gt;</td>
<td>Reference to an agent in a validation report</td>
</tr>
<tr>
<td>--&lt;linkingAgentIdentifierType&gt;</td>
<td>Type of the agent identifier</td>
</tr>
<tr>
<td>--&lt;linkingAgentIdentifierValue&gt;</td>
<td>Value of the agent identifier</td>
</tr>
<tr>
<td>&lt;-&lt;linkingObjectIdentifier&gt;</td>
<td>Reference to an object in a validation report</td>
</tr>
<tr>
<td>--&lt;linkingObjectIdentifierType&gt;</td>
<td>Type of the object identifier</td>
</tr>
<tr>
<td>--&lt;linkingObjectIdentifierValue&gt;</td>
<td>Value of the object identifier</td>
</tr>
</tbody>
</table>

Transfer of a SIP:

<table>
<thead>
<tr>
<th>Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;-&lt;eventType&gt;</td>
<td>“transfer”</td>
</tr>
<tr>
<td>&lt;-&lt;eventDetail&gt;</td>
<td>“Transfer of submission information package”</td>
</tr>
<tr>
<td>&lt;-&lt;linkingAgentIdentifier&gt;</td>
<td>Reference to the user</td>
</tr>
<tr>
<td>&lt;-&lt;linkingObjectIdentifier&gt;</td>
<td>Reference to a SIP</td>
</tr>
</tbody>
</table>

Decompression of a SIP:

<table>
<thead>
<tr>
<th>Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;-&lt;eventType&gt;</td>
<td>“decompression”</td>
</tr>
<tr>
<td>&lt;-&lt;eventDetail&gt;</td>
<td>“Decompression of submission information package”</td>
</tr>
<tr>
<td>&lt;-&lt;linkingAgentIdentifier&gt;</td>
<td>Reference to the decompression software</td>
</tr>
<tr>
<td>&lt;-&lt;linkingObjectIdentifier&gt;</td>
<td>Reference to a SIP</td>
</tr>
</tbody>
</table>

Virus check:

<table>
<thead>
<tr>
<th>Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;-&lt;eventType&gt;</td>
<td>“virus check”</td>
</tr>
<tr>
<td>&lt;-&lt;eventDetail&gt;</td>
<td>“Virus check of submitted files”</td>
</tr>
</tbody>
</table>
- `<linkingAgentIdentifier>` Reference to the virus check software
- `<linkingObjectIdentifier>` Reference to a SIP

Digital signature validation:

<table>
<thead>
<tr>
<th>Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;eventType&gt;</code></td>
<td>“digital signature validation”</td>
</tr>
<tr>
<td><code>&lt;eventDetail&gt;</code></td>
<td>“Submission information package digital signature validation”</td>
</tr>
<tr>
<td><code>&lt;linkingAgentIdentifier&gt;</code></td>
<td>Reference to the digital signature validation software</td>
</tr>
<tr>
<td><code>&lt;linkingObjectIdentifier&gt;</code></td>
<td>Reference to a digital signature</td>
</tr>
</tbody>
</table>

METS schema validation:

<table>
<thead>
<tr>
<th>Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;eventType&gt;</code></td>
<td>“validation”</td>
</tr>
<tr>
<td><code>&lt;eventDetail&gt;</code></td>
<td>“METS schema validation”</td>
</tr>
<tr>
<td><code>&lt;linkingAgentIdentifier&gt;</code></td>
<td>Reference to the schema validation software</td>
</tr>
<tr>
<td><code>&lt;linkingObjectIdentifier&gt;</code></td>
<td>Reference to a METS document</td>
</tr>
</tbody>
</table>

Additional METS validation of required features:

<table>
<thead>
<tr>
<th>Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;eventType&gt;</code></td>
<td>“validation”</td>
</tr>
<tr>
<td><code>&lt;eventDetail&gt;</code></td>
<td>“Additional METS validation of required features”</td>
</tr>
<tr>
<td>--- <code>&lt;eventOutcomeDetailNote&gt;</code></td>
<td>Not used</td>
</tr>
<tr>
<td>--- <code>&lt;eventOutcomeDetailExtension&gt;</code></td>
<td>Used</td>
</tr>
<tr>
<td><code>&lt;linkingAgentIdentifier&gt;</code></td>
<td>Reference to the validation software</td>
</tr>
<tr>
<td><code>&lt;linkingObjectIdentifier&gt;</code></td>
<td>Reference to a METS document</td>
</tr>
</tbody>
</table>

Fixity check of digital objects: The event contains the fixity checks of all digital objects of the SIP in one bundle.

<table>
<thead>
<tr>
<th>Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;eventType&gt;</code></td>
<td>“fixity check”</td>
</tr>
<tr>
<td><code>&lt;eventDetail&gt;</code></td>
<td>“Fixity check of digital objects in submission information package”</td>
</tr>
<tr>
<td><code>&lt;linkingAgentIdentifier&gt;</code></td>
<td>Reference to the fixity check software</td>
</tr>
<tr>
<td><code>&lt;linkingObjectIdentifier&gt;</code></td>
<td>Reference to a SIP</td>
</tr>
</tbody>
</table>

Digital object validation: For each digital object in the SIP one or more events are created.

<table>
<thead>
<tr>
<th>Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;eventType&gt;</code></td>
<td>“validation”</td>
</tr>
<tr>
<td><code>&lt;eventDetail&gt;</code></td>
<td>“Digital object validation”</td>
</tr>
<tr>
<td>--- <code>&lt;eventOutcomeDetailNote&gt;</code></td>
<td>Occurs, if the <code>&lt;eventOutcomeDetailExtension&gt;</code> element is not used.</td>
</tr>
<tr>
<td>--- <code>&lt;eventOutcomeDetailExtension&gt;</code></td>
<td>Occurs, if the <code>&lt;eventOutcomeDetailNote&gt;</code> element is not used.</td>
</tr>
<tr>
<td><code>&lt;linkingAgentIdentifier&gt;</code></td>
<td>Reference to the digital object validation software</td>
</tr>
<tr>
<td><code>&lt;linkingObjectIdentifier&gt;</code></td>
<td>Reference to a digital object</td>
</tr>
</tbody>
</table>

Summary of SIP validation:

<table>
<thead>
<tr>
<th>Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;eventType&gt;</code></td>
<td>“validation”</td>
</tr>
<tr>
<td><code>&lt;eventDetail&gt;</code></td>
<td>“Validation compilation of submission information package”</td>
</tr>
<tr>
<td><code>&lt;linkingAgentIdentifier&gt;</code></td>
<td>Reference to the validation software</td>
</tr>
</tbody>
</table>
Creation of archival information package:

<table>
<thead>
<tr>
<th>Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>- &lt;linkingObjectIdentifier&gt;</td>
<td>Reference to a SIP</td>
</tr>
</tbody>
</table>

**Preparation responsibility change:**

<table>
<thead>
<tr>
<th>Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>- &lt;eventType&gt;</td>
<td>“preservation responsibility change”</td>
</tr>
<tr>
<td>- &lt;eventDetail&gt;</td>
<td>“Preservation responsibility change to the digital preservation system”</td>
</tr>
<tr>
<td>- &lt;linkingAgentIdentifier&gt;</td>
<td>Reference to the transfer software</td>
</tr>
<tr>
<td>- &lt;linkingObjectIdentifier&gt;</td>
<td>Reference to a SIP</td>
</tr>
</tbody>
</table>

### A.3. Agents

PREMIS XML formatted statements are created for all the parties performing actions in ingest (cf. Section A.2). These parties are called PREMIS agents. An agent can be either a human (such as digital preservation specialist) or an application. This section lists the data elements describing these PREMIS agents.

**User:**

<table>
<thead>
<tr>
<th>Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;agent&gt;</td>
<td></td>
</tr>
<tr>
<td>- &lt;agentIdentifier&gt;</td>
<td>Identifier of an agent</td>
</tr>
<tr>
<td>-- &lt;agentIdentifierType&gt;</td>
<td>Type of the identifier</td>
</tr>
<tr>
<td>-- &lt;agentIdentifierValue&gt;</td>
<td>Value of the identifier</td>
</tr>
<tr>
<td>- &lt;agentName&gt;</td>
<td>The user (organization username)</td>
</tr>
<tr>
<td>- &lt;agentType&gt;</td>
<td>“organization” (type of an agent)</td>
</tr>
</tbody>
</table>

**A DP application:** Each application referenced in the report must have its own agent description.

<table>
<thead>
<tr>
<th>Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;agent&gt;</td>
<td></td>
</tr>
<tr>
<td>- &lt;agentIdentifier&gt;</td>
<td>Identifier of an agent</td>
</tr>
<tr>
<td>-- &lt;agentIdentifierType&gt;</td>
<td>Type of the identifier</td>
</tr>
<tr>
<td>-- &lt;agentIdentifierValue&gt;</td>
<td>Value of the identifier</td>
</tr>
<tr>
<td>- &lt;agentName&gt;</td>
<td>Name of the application</td>
</tr>
<tr>
<td>- &lt;agentType&gt;</td>
<td>“software” (type of an agent)</td>
</tr>
</tbody>
</table>
REFERENCES


[JSON_API] JSON API. http://jsonapi.org/


