
edeposit.amqp.aleph

Vydání 1.9.5

14.12.2015

1	Module API	3
1.1	aleph package	3
1.2	Data structures	19
1.3	Aleph's lowlevel API	25
2	Indices and tables	29
	Rejstřík modulů Pythonu	31

This module is used to read and export metadata about epublications to/from [Aleph](#).
Module is developed as part of the [e-deposit](#) project.

Module API

1.1 aleph package

1.1.1 Query workflow

AQMP is handled by `edeposit.amqp` module, this package provides just datastructures and `reactToAMQPMessage()` function, which is used in daemon to translate highlevel requests to lowlevel queries to Aleph's webapi.

AMQP query

To query Aleph thru AMQP, run `edeposit_amqp_alephdaemon` (from `edeposit.amqp` package) and create one of the Queries - `ISBNQuery` for example and put it into `SearchRequest` wrapper and send the message to the Aleph's exchange:

```
request = SearchRequest(
    ISBNQuery("80-251-0225-4")
)

amqp.send( # you can use pika library to send data to AMQP queue
    message = serialize(request),
    properties = "..",
    exchange = "ALEPH'S_EXCHANGE"
)
```

and you will get back AMQP message with `SearchResult`.

Poznámka: You don't have to import all structures from `datastructures`, they should be automatically imported and made global in `__init__.py`.

Count requests

If you want to just get count of how many items is there in Aleph, just wrap the `ISBNQuery` with `CountRequest` class:

```
isbnq = ISBNQuery("80-251-0225-4")
request = CountRequest(isbnq)

# rest is same..
```

and you will get back *CountResult*.

Poznámka: You should always use *CountRequest* instead of just calling *len()* to *SearchResult.records* - it doesn't put that much load to Aleph. Also Aleph is restricted to 150 requests per second.

Direct queries

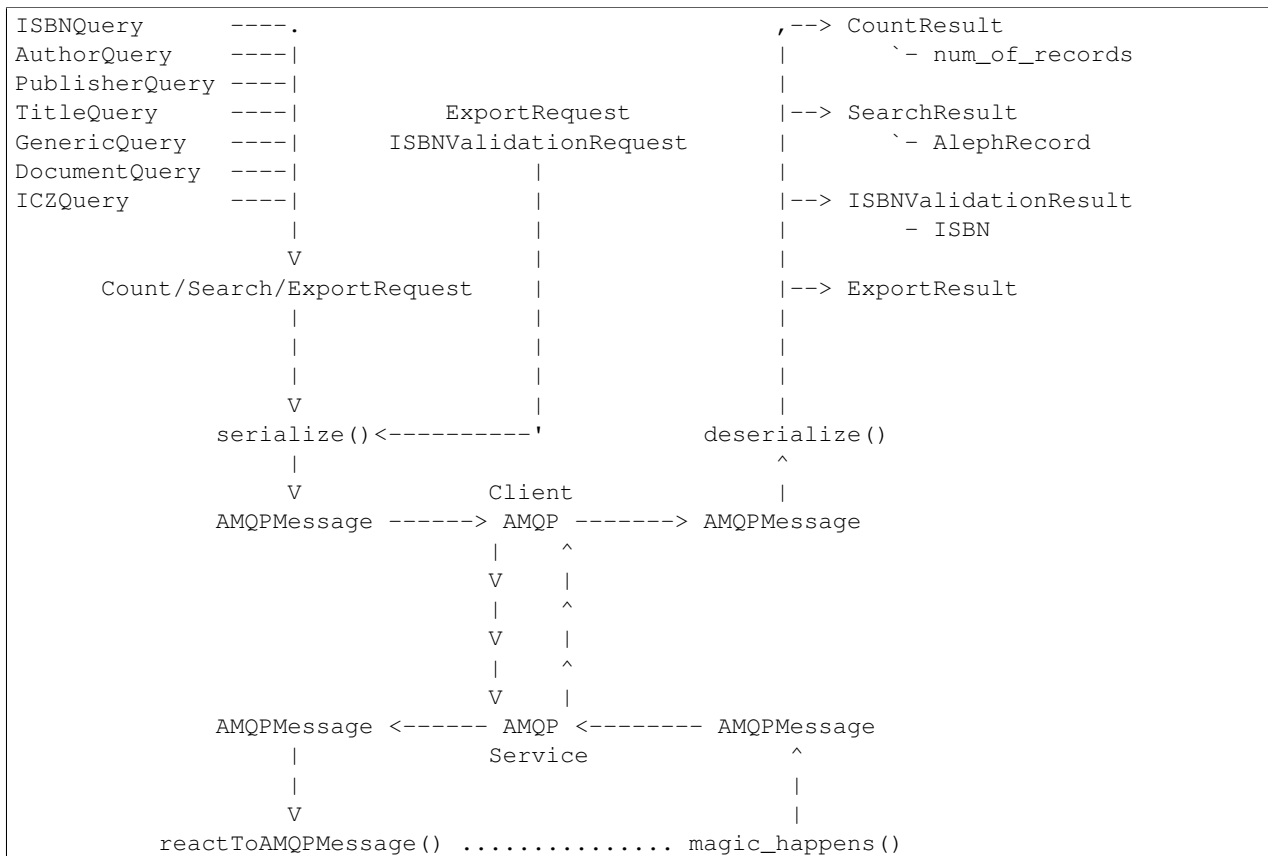
As I said, this module provides only direct access to Aleph, AMQP communication is handled in *edeposit.amqp*.

If you want to access module directly, you can use *reactToAMQPMessage()* wrapper, or query *aleph* submodule directly.

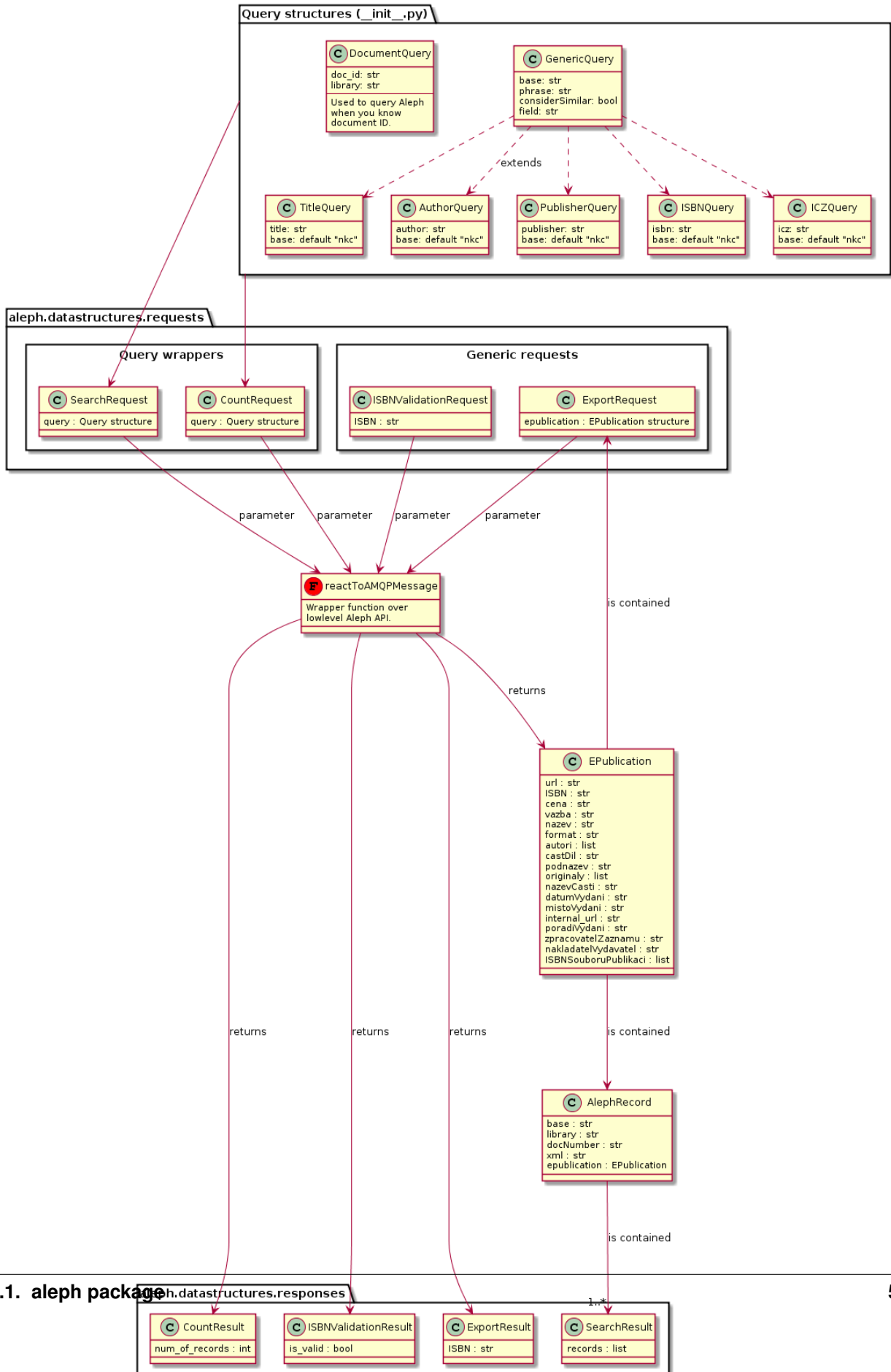
reactToAMQPMessage() is preferred, because in that case, you don't have to deal with Aleph lowlevel API, which can be little bit annoying.

Diagrams

Here is ASCII flow diagram for you:



and here is (pseudo) UML:



Neat, isn't it?

API

class `aleph.GenericQuery`

Nadřídí: `aleph.GenericQuery`, `aleph._QueryTemplate`

Used for generic queries to Aleph.

Parametry

- **base** (*str*) – Which base in Aleph will be queried. This depends on settings of your server. See `aleph.getListOfBases()` for details.
- **phrase** (*str*) – What are you looking for.
- **considerSimilar** (*bool*) – Don't use this, it usually doesn't work.
- **field** (*str*) – Which field you want to use for search. See `aleph.VALID_ALEPH_FIELDS` for list of valid bases.

For details of base/phrase/.. parameters, see `aleph.searchInAleph()`. All parameters also serves as properties.

This is used mainly if you want to search by your own parameters and don't want to use prepared wrappers (`AuthorQuery/ISBNQuery/..`).

class `aleph.DocumentQuery`

Nadřídí: `aleph.DocumentQuery`

Query Aleph when you know the Document ID.

Parametry

- **doc_id** (*str*) – ID number as string.
- **library** (*str*, *default settings.DEFAULT_LIBRARY*) – Library.

`getSearchResult()`

Vrací `SearchResult` document with given `doc_id`.

Typ návratové hodnoty `object`

Raises `aleph.DocumentNotFoundException` – When document is not found.

`getCountResult()`

Vrací 0/1 whether the document is found or not.

Typ návratové hodnoty `int`

class `aleph.ISBNQuery`

Nadřídí: `aleph.ISBNQuery`, `aleph._QueryTemplate`

Used to query Aleph to get books by ISBN.

Parametry

- **ISBN** (*str*) – ISBN 10/13.
- **base** (*str*, *optional*) – If not set, `settings.ALEPH_DEFAULT_BASE` is used.

Poznámka: ISBN is not unique, so you can get back lot of books with same ISBN. Some books also have two or more ISBNs.

class `aleph.AuthorQuery`

Nadřídí: `aleph.AuthorQuery`, `aleph._QueryTemplate`

Used to query Aleph to get books by Author.

Parametry

- **author** (*str*) – Author’s name/lastname in UTF-8.
- **base** (*str, optional*) – If not set, `settings.ALEPH_DEFAULT_BASE` is used.

class `aleph.PublisherQuery`

Nadřídí: `aleph.PublisherQuery`, `aleph._QueryTemplate`

Used to query Aleph to get books by Publisher.

Parametry

- **publisher** (*str*) – Publisher’s name in UTF-8.
- **base** (*str, optional*) – If not set, `settings.ALEPH_DEFAULT_BASE` is used.

class `aleph.TitleQuery`

Nadřídí: `aleph._QueryTemplate`, `aleph.TitleQuery`

Used to query Aleph to get books by book’s title/name.

Parametry

- **title** (*str*) – Book’s title in UTF-8.
- **base** (*str, optional*) – If not set, `settings.ALEPH_DEFAULT_BASE` is used.

class `aleph.ICZQuery`

Nadřídí: `aleph._QueryTemplate`, `aleph.ICZQuery`

Used to query Aleph to get books by record’s identification number *icz*.

Parametry

- **icz** (*str*) – Identification number (nkc20150003029 for example).
- **base** (*str, optional*) – If not set, `settings.ALEPH_DEFAULT_BASE` is used.

`aleph.reactToAMQPMessage` (*req, send_back*)

React to given (AMQP) message.

This function is used by `edeposit.amqp.alephdaemon`. It works as highlevel wrapper for whole module.

Example

```
>>> import aleph
```

```
>>> request = aleph.SearchRequest (
...     aleph.ISBNQuery ("80-251-0225-4")
... )
>>> request
SearchRequest (query=ISBNQuery (ISBN='80-251-0225-4', base='nkc'))
```

```
>>> response = aleph.reactToAMQPMessage (request, None)
```

```

>>> response # formatted by hand for purposes of example
SearchResult(
  records=[
    AlephRecord(
      base='nkc',
      library='NKC01',
      docNumber=1492461,
      xml='HERE IS WHOLE MARC OAI RECORD',
      epublication=EPublication(
        ISBN=['80-251-0225-4'],
        nazev='Umění programování v UNIXu /',
        podnazev='',
        vazba='(brož.) :',
        cena='Kč 590,00',
        castDil='',
        nazevCasti='',
        nakladatelVydavatel='Computer Press,',
        datumVydani='2004',
        poradiVydani='1. vyd.',
        zpracovatelZaznamu='BOA001',
        format='23 cm',
        url='',
        mistoVydani='Brno :',
        ISBNsouboruPublikaci=[],
        autori=[
          Author(
            firstName='Eric S.',
            lastName='Raymond',
            title=''
          )
        ],
        originaly=[
          'Art of UNIX programming'
        ],
        internal_url=''
      )
    ]
  )
)

```

Parametry

- **req** (*Request class*) – Any of the Request class from [aleph.datastructures.requests](#).
- **send_back** (*fn reference*) – Reference to function for responding. This is useful for progress monitoring for example. Function takes one parameter, which may be response structure/namedtuple, or string or whatever would be normally returned.

Vrací Result of search in Aleph. See [aleph.datastructures.results](#) submodule.

Typ návratové hodnoty Result class

Raises ValueError – If bad type of *req* structure is given.

1.1.2 Submodules

Aleph lowlevel API

Aleph X-Service wrapper.

This module allows you to query Aleph's X-Services module (Aleph server is defined by `aleph.settings.ALEPH_URL` in `settings.py`).

There are two levels of abstraction.

Lowlevel

You can use this functions to access Aleph:

```
searchInAleph(base, phrase, considerSimilar, field)
downloadRecords(search_result, [from_doc])
getDocumentIDs(aleph_search_result, [number_of_docs])
downloadMARCXML(doc_id, library)
downloadMARCOAI(doc_id, base)
```

Workflow Aleph works in strange way, that he won't allow you to access desired information directly.

You have to create search request by calling `searchInAleph()` first, which will return dictionary with few important informations about session.

This dictionary can be later used as parameter to `getDocumentIDs()` function, which will give you list of `DocumentID` named tuples.

Poznámka: `namedtuple()` is used, because to access your document, you don't need just `document ID` number, but also `library ID` string.

Depending on your system, there may be just only one accessible library, or multiple ones, and then you will be glad, that you get both of this informations together.

`DocumentID` can be used as parameter to `downloadMARCXML()`.

Lets look at some code:

```
ids = getDocumentIDs(searchInAleph("nkc", "test", False, "wrđ"))
for id_num, library in ids:
    XML = downloadMARCXML(id_num, library)

    # processDocument(XML)
```

High-level

XML wrappers This wrappers returns full XML records from Aleph:

- `getISBNsXML()`
- `getAuthorsBooksXML()`
- `getPublishersBooksXML()`
- `getBooksTitleXML()`
- `getICZBooksXML()`

ID wrappers There are wrappers, which returns ID's of matching document in Aleph:

- `getISBNsIDs ()`
- `getAuthorsBooksIDs ()`
- `getPublishersBooksIDs ()`
- `getBooksTitleIDs ()`
- `getICZBooksIDs ()`

You can then download them using `downloadMARCXML ()` or `downloadMARCOAI ()`.

Count wrappers Count wrappers returns just the number of records with given parameters are there in aleph.

- `getISBNCount ()`
- `getAuthorsBooksCount ()`
- `getPublishersBooksCount ()`
- `getBooksTitleCount ()`
- `getICZBooksCount ()`

Poznámka: Counting functions are by one request faster than just counting results from standard getters. It is preferred to use them to reduce load to Aleph.

Other noteworthy properties

List of valid bases can be obtained by calling `getListOfBases ()`, which returns list of strings.

There is also defined exception tree - see `AlephException` doc-string for details.

`aleph.aleph.VALID_ALEPH_FIELDS = ['wrđ', 'wtl', 'wau', 'kwk', 'txt', 'wpb', 'wpp', 'wyr', 'ssn', 'sbn', 'isn', 'ob', 'wpf']`

- `wrd` - Všechny údaje [*All fields*]
- `wtl` - Název [*Title/name of the book*]
- `wau` - Autor (osoba, korporace) [*Author (person, corporation)*]
- `wkw` - Předmět (klíčová slova) [*Subject (keywords)*]
- `txt` - Slova z obsahu (table of cont.) [*Words from table of content*]
- `wpb` - Nakladatel [*Publisher*]
- `wpp` - Místo vydání [*Place of publication*]
- `wyr` - Rok vydání [*Year of publication*]
- `ssn` - ISSN
- `sbn` - ISBN / ISMN
- `isn` - ISBN / ISMN / ISSN
- `ob` - Obsazení (hudební díla) [*Cast (musical works)*]
- `wpf` - Periodicita [*Periodicity*]
- `wpv` - Kód země vydání [*Country code*]
- `wln` - Kód jazyka dokumentu [*Language code*]

- wlo - Kód jazyka originálu [*Lanugage code of original*]
- wtp - Druh dokumentu [*Type of document*]
- sg - Signatura [*Signature*]
- bar - Čárový kód [*Barcode*]
- cnb - Číslo národní bibl. [*Number of national bibl.*]
- icz - Identifikační číslo [*Identification number*]
- sys - Systémové číslo [*System number*]
- wpk

exception aleph.aleph.**AlephException** (*message*)

Exception tree:

```

- AlephException
  |- InvalidAlephBaseException
  |- InvalidAlephFieldException
  |- LibraryNotFoundException
  ` - DocumentNotFoundException
```

exception aleph.aleph.**InvalidAlephBaseException** (*message*)

exception aleph.aleph.**InvalidAlephFieldException** (*message*)

exception aleph.aleph.**LibraryNotFoundException** (*message*)

exception aleph.aleph.**DocumentNotFoundException** (*message*)

class aleph.aleph.**DocumentID**

This structure is used to store “*pointer*” to document in aleph.

id

int – ID of document.

library

str – This can be different for each document, depend on your system.

base

str – Default “*nkC*”, but really depends on what bases you have defined in your Aleph server.

aleph.aleph.**getListOfBases** ()

This function is here mainly for purposes of unittest

Vrací Valid bases as they are used as URL parameters in links at Aleph main page.

Typ návratové hodnoty list of str

aleph.aleph.**searchInAleph** (*base, phrase, considerSimilar, field*)

Send request to the aleph search engine.

Request itself is pretty useless, but it can be later used as parameter for *getDocumentIDs* (), which can fetch records from Aleph.

Parametry

- **base** (*str*) – which database you want to use
- **phrase** (*str*) – what do you want to search
- **considerSimilar** (*bool*) – fuzzy search, which is not working at all, so don’t use it
- **field** (*str*) – where you want to look (see: *VALID_ALEPH_FIELDS*)

Vrací

consisting from following fields:

error (optional): present if there was some form of error
no_entries (int): number of entries that can be fetch from aleph
no_records (int): no idea what is this, but it is always \geq than *no_entries*
set_number (int): important - something like ID of your request
session-id (str): used to count users for licensing purposes

Typ návratové hodnoty dictionary

Example

Returned dict:

```
{
  'session-id': 'YLI54HBQJESUTS678YYUNKEU4BNAUJDKA914GMF39J6K89VSCB',
  'set_number': 36520,
  'no_records': 1,
  'no_entries': 1
}
```

Raises

- *AlephException* – if Aleph doesn't return any information
- *InvalidAlephFieldException* – if specified field is not valid

`aleph.aleph.downloadRecords` (*search_result*, *from_doc=1*)

Download *MAX_RECORDS* documents from *search_result* starting from *from_doc*.

Attr: *search_result* (dict): returned from *searchInAleph()*. *from_doc* (int, default 1): Start from document number *from_doc*.

Vrací List of XML strings with documents in MARC OAI.

Typ návratové hodnoty list

`aleph.aleph.getDocumentIDs` (*aleph_search_result*, *number_of_docs=-1*)

Get IDs, which can be used as parameters for other functions.

Parametry

- **aleph_search_result** (*dict*) – returned from *searchInAleph()*
- **number_of_docs** (*int*, *optional*) – how many *DocumentID* from set given by *aleph_search_result* should be returned. Default -1 for all of them.

Vrací *DocumentID* named tuples to given *aleph_search_result*.

Typ návratové hodnoty list

Raises *AlephException* – If Aleph returns unknown format of data.

Poznámka: Returned *DocumentID* can be used as parameters to *downloadMARCXML()*.

`aleph.aleph.downloadMARCXML (doc_id, library, base='nkc')`

Download MARC XML document with given *doc_id* from given *library*.

Parametry

- **doc_id** (DocumentID) – You will get this from `getDocumentIDs()`.
- **library** (*str*) – “NKC01” in our case, but don’t worry, `getDocumentIDs()` adds library specification into *DocumentID* named tuple.

Vrací MARC XML unicode string.

Typ návratové hodnoty *str*

Raises

- *LibraryNotFoundException*
- *DocumentNotFoundException*

`aleph.aleph.downloadMARCOAI (doc_id, base)`

Download MARC OAI document with given *doc_id* from given (logical) *base*.

Funny part is, that some documents can be obtained only with this function in their full text.

Parametry

- **doc_id** (*str*) – You will get this from `getDocumentIDs()`.
- **base** (*str, optional*) – Base from which you want to download Aleph document. This seems to be duplicate with `searchInAleph()` parameters, but it’s just something Aleph’s X-
Services wants, so ..

Vrací MARC XML Unicode string.

Typ návratové hodnoty *str*

Raises

- *InvalidAlephBaseException*
- *DocumentNotFoundException*

`aleph.aleph.getISBNsXML (isbn, base='nkc')`

Download full XML record for given *isbn* in *base*.

Parametry

- **isbn** (*str*) – ISBN of the books you want to get.
- **base** (*str*) – Base on which will be search performed. Default `aleph.settings.ALEPH_DEFAULT_BASE`.

Vrací List of strings with full OAI XML representation of the record.

Typ návratové hodnoty *list*

`aleph.aleph.getISSNsXML (issn, base='nkc')`

Download full XML record for given *issn* in *base*.

Parametry

- **issn** (*str*) – ISSN of the books you want to get.
- **base** (*str*) – Base on which will be search performed. Default `aleph.settings.ALEPH_DEFAULT_BASE`.

Vrací List of strings with full OAI XML representation of the record.

Typ návratové hodnoty *list*

`aleph.aleph.getAuthorsBooksXML (author, base='nkc')`

Download full XML record for given *author* in *base*.

Parametry

- **author** (*str*) – Name of the *author* of the books you want to get.
- **base** (*str*) – Base on which will be search performed. Default `aleph.settings.ALEPH_DEFAULT_BASE`.

Vrací List of strings with full **OAI** XML representation of the record.

Typ návratové hodnoty *list*

`aleph.aleph.getPublishersBooksXML (publisher, base='nkc')`

Download full XML record for given *publisher* in *base*.

Parametry

- **publisher** (*str*) – Name of the *publisher* of the books you want to get.
- **base** (*str*) – Base on which will be search performed. Default `aleph.settings.ALEPH_DEFAULT_BASE`.

Vrací List of strings with full **OAI** XML representation of the record.

Typ návratové hodnoty *list*

`aleph.aleph.getBooksTitleXML (title, base='nkc')`

Download full XML record for given *title* in *base*.

Parametry

- **title** (*str*) – *title* of the books you want to get.
- **base** (*str*) – Base on which will be search performed. Default `aleph.settings.ALEPH_DEFAULT_BASE`.

Vrací List of strings with full **OAI** XML representation of the record.

Typ návratové hodnoty *list*

`aleph.aleph.getICZBooksXML (icz, base='nkc')`

Download full XML record for given *icz* (identification number) in *base*.

Parametry

- **icz** (*str*) – Identification number used to search Aleph.
- **base** (*str*) – Base on which will be search performed. Default `aleph.settings.ALEPH_DEFAULT_BASE`.

Vrací List of strings with full **OAI** XML representation of the record.

Typ návratové hodnoty *list*

`aleph.aleph.getISBNsIDs (isbn, base='nkc')`

Get list of *DocumentID* objects of documents with given *isbn*.

Parametry

- **isbn** (*str*) – ISBN string.
- **base** (*str*; *optional*) – Base on which will be search performed. Default `aleph.settings.ALEPH_DEFAULT_BASE`.

Vrací of *DocumentID* objects

Typ návratové hodnoty *list*

`aleph.aleph.getAuthorsBooksIDs (author, base='nkc')`

Get list of *DocumentID* objects of documents with given *author*.

Parametry

- **author** (*str*) – Authors name/lastname in UTF-8.
- **base** (*str; optional*) – base on which will be search performed. Default `aleph.settings.ALEPH_DEFAULT_BASE`.

Vrací of *DocumentID* objects

Typ návratové hodnoty *list*

`aleph.aleph.getPublishersBooksIDs (publisher, base='nkc')`

Get list of *DocumentID* objects of documents with given *publisher*.

Parametry

- **publisher** (*str*) – Name of publisher which will be used to search Aleph.
- **base** (*str; optional*) – base on which will be search performed. Default `aleph.settings.ALEPH_DEFAULT_BASE`.

Vrací of *DocumentID* objects

Typ návratové hodnoty *list*

`aleph.aleph.getBooksTitleIDs (title, base='nkc')`

Get list of *DocumentID* objects of documents with given *title*.

Parametry

- **title** (*str*) – Title (name) of the book which will be used to search in Aleph.
- **base** (*str; optional*) – base on which will be search performed. Default `aleph.settings.ALEPH_DEFAULT_BASE`.

Vrací of *DocumentID* objects

Typ návratové hodnoty *list*

`aleph.aleph.getICZBooksIDs (icz, base='nkc')`

Get list of *DocumentID* objects of documents with given *icz* (identification number).

Parametry

- **icz** (*str*) – Identification number used to search Aleph.
- **base** (*str; optional*) – base on which will be search performed. Default `aleph.settings.ALEPH_DEFAULT_BASE`.

Vrací of *DocumentID* objects

Typ návratové hodnoty *list*

`aleph.aleph.getISBNCount (isbn, base='nkc')`

Get number of records in Aleph which match given *isbn*.

Parametry

- **isbn** (*str*) – ISBN string.

- **base** (*str*; *optional*) – Base on which will be search performed. Default `aleph.settings.ALEPH_DEFAULT_BASE`.

Vrací Number of matching documents in Aleph.

Typ návratové hodnoty `int`

`aleph.aleph.getAuthorsBooksCount` (*author*, *base='nkc'*)

Get number of records in Aleph which match given *author*.

Parametry

- **author** (*str*) – Authors name/lastname in UTF-8.
- **base** (*str*; *optional*) – base on which will be search performed. Default `aleph.settings.ALEPH_DEFAULT_BASE`.

Vrací Number of matching documents in Aleph.

Typ návratové hodnoty `int`

`aleph.aleph.getPublishersBooksCount` (*publisher*, *base='nkc'*)

Get number of records in Aleph which match given *publisher*.

Parametry

- **publisher** (*str*) – Name of publisher which will be used to search Aleph.
- **base** (*str*; *optional*) – base on which will be search performed. Default `aleph.settings.ALEPH_DEFAULT_BASE`.

Vrací Number of matching documents in Aleph.

Typ návratové hodnoty `int`

`aleph.aleph.getBooksTitleCount` (*title*, *base='nkc'*)

Get number of records in Aleph which match given *title*.

Parametry

- **title** (*str*) – Title (name) of book which will be used to search Aleph.
- **base** (*str*; *optional*) – base on which will be search performed. Default `aleph.settings.ALEPH_DEFAULT_BASE`.

Vrací Number of matching documents in Aleph.

Typ návratové hodnoty `int`

`aleph.aleph.getICZBooksCount` (*icz*, *base='nkc'*)

Get number of records in Aleph which match given *title*.

Parametry

- **icz** (*str*) – Identification number used to search Aleph.
- **base** (*str*; *optional*) – base on which will be search performed. Default `aleph.settings.ALEPH_DEFAULT_BASE`.

Vrací Number of matching documents in Aleph.

Typ návratové hodnoty `int`

Export module

This module is used to put data to Aleph. It is based on custom made webform, which is currently used to report new books by publishers.

Most important function from this module is `exportEPublication()`, which will do everything, that is needed to do, to export `EPublication` structure to the Aleph.

Varování: This whole module is highly dependent on processes, which are defined as import processes at the Czech National Library.

Varování: If you want to use export ability in your library, you should rewrite this and take care, that you are sending data somewhere, where someone will process them. Otherwise, you can fill your library's database with crap.

Poznámka: Source code of the webform is not available at this moment (it was created by third party), but it is possible, that it will be in future. This will highly depend on number of people, which will use this project.

exception `aleph.export.ExportException` (*message*)

exception `aleph.export.InvalidISBNException` (*message*)

exception `aleph.export.ExportRejectedException` (*message*)

class `aleph.export.PostData` (*epub*)

This class is used to transform data from `EPublication` to dictionary, which is sent as POST request to Aleph third-party `webform`.

Poznámka: Class is used instead of simple function, because there is 29 POST parameters with internal dependencies, which need to be processed and validated before they can be passed to webform.

Parametry `epub` (`EPublication`) – structure, which will be converted (see `EPublication` for details).

Attr: `_POST` (dict): dictionary with parsed data mapping (dict): dictionary with some of mapping, which are applied to

`_POST` dict in post processing

Varování: Don't manipulate `_POST` property directly, if you didn't really know the internal structure and how the mapping is applied.

get_POST_data ()

Vrací POST data, which can be sent to webform using `urllib` or similar library

Typ návratové hodnoty dict

`aleph.export.exportEPublication` (*epub*)

Send *epub* `EPublication` object to Aleph, where it will be processed by librarians.

Parametry `epub` (`EPublication`) – structure for export

Varování: The export function is expecting some of the EPublication properties to be filled with non-blank data.

Specifically:

- `EPublication.ISBN`
- `EPublication.nazev`
- `EPublication.mistoVydani`
- `EPublication.datumVydani`
- `EPublication.poradiVydani`
- `EPublication.zpracovatelZaznamu`
- `EPublication.vazba`
- `EPublication.format`
- `EPublication.format`
- `EPublication.nakladatelVydavatel`

Settings and configuration

Module is containing all necessary global variables for package.

Module also has ability to read user-defined data from two paths: `$HOME/_SETTINGS_PATH` and `/etc/_SETTINGS_PATH`.

Poznámka: If the first path is found, other is ignored.

Example of the configuration file (`$HOME/edeposit/aleph.json`):

```
{
  "EDEPOSIT_EXPORT_SIGNATURE": "edeposit fancy signature",
  "EDEPOSIT_EXPORT_REFERERER": "from edeposit ^-^"
}
```

Attributes

`aleph.settings.BASE_PATH = '/home/docs/checkouts/readthedocs.org/user_builds/edeposit-amqp-aleph/checkouts/latest/'`
Module's path.

`aleph.settings.ALEPH_DEFAULT_BASE = 'nkc'`
Default base used to search in Aleph

`aleph.settings.DEFAULT_LIBRARY = 'CZE01'`
Default library in aleph.

`aleph.settings.ALEPH_URL = 'http://aleph.nkp.cz'`
URL used to read from Aleph. See Aleph's X-service module.

`aleph.settings.EDEPOSIT_EXPORT_SIGNATURE = 'edeposit'`
Signature used when the module is writing to the Aleph

`aleph.settings.EDEPOSIT_EXPORT_REFERERER = 'edeposit'`
Referer, which is used when module is writing to the Aleph

`aleph.settings.ALEPH_EXPORT_URL = 'http://aleph.nkp.cz/aleph-cgi/e-deposit'`
URL, of form, which is used to write to the Aleph

`aleph.settings.get_all_constants ()`
Get list of all uppercase, non-private globals (doesn't start with _).

Vrací Uppercase names defined in *globals()* (variables from this module).

Typ návratové hodnoty *list*

`aleph.settings.substitute_globals` (*config_dict*)

Set global variables to values defined in *config_dict*.

Parametry `config_dict` (*dict*) – dictionary with data, which are used to set *globals*.

Poznámka: *config_dict* have to be dictionary, or it is ignored. Also all variables, that are not already in *globals*, or are not types defined in `_ALLOWED` (str, int, float) or starts with `_` are silently ignored.

1.2 Data structures

This module contains communication structures used in AMQP.

1.2.1 Structures

Author structure

class `aleph.datastructures.author.Author`

Informations about author (or person).

firstName

str

lastName

str

title

str

FormatEnum enum

class `aleph.datastructures.format_enum.FormatEnum`

Enum used as format in *EPublication*.

CD = 'CD-ROM'

DVD = 'DVD'

BROZ = 'bro\x5\xbe.'

MAPA = 'mapa'

VAZANA = 'v\x3\xa1z.'

ONLINE = 'online'

EPublication structure

class `aleph.datastructures.epublication.EPublication`

This structure is returned as result of users *SearchRequest*.

In case of *Search/Count* requests, this structure is filled with data from MARC XML record.

url

str – Url specified by publisher (THIS IS NOT INTERNAL URL!).

ISBN

list – List of ISBNs for the book.

cena

str – Price of the book.

vazba

str – Bidding of the book.

nazev

str – Name of the book.

format

str – Format of the book - see `FormatEnum`.

autori

list – List of `Author` objects.

castDil

str – Which part of the series of books is this.

anotace

str – Anotation. Max lenght: 500 chars..

podnazev

str – Subname of the book.

id_number

str – Identification number in aleph - starts.

originaly

list – List of (str) ISBN's of original books in case of translations.

nazevCasti

str – Name of part of the series.

datumVydani

str – Date of publication.

mistoVydani

str – City/country origin of the publication.

internal_url

str – Link to edeposit/kramerius system.

poradiVydani

str – Order of publication.

invalid_ISBN

list – List of INVALID ISBNs for this book.

zpracovatelZaznamu

str – Processor/manufacturer of record. with nkc - nkc20150003133.

nakladatelVydavatel

str – Publisher's name.

ISBNsouboruPublikaci

list – List of strings with ISBN of the book series.

static from_xml (*xml*)

Convert MARCXMLRecord object to *EPublication* namedtuple.

Parametry xml (*str/MARCXMLRecord*) – MARC XML which will be converted to *EPublication*. In case of *str*, <record> tag is required.

Vrací *EPublication* namedtuple with data about publication.

Typ návratové hodnoty structure

EPeriodical structure

class aleph.datastructures.eperiodical.**EPeriodical**

This structure is returned as result of users *SearchRequest*.

In case of *Search/Count* requests, this structure is filled with data from MARC XML record.

url

str – Url specified by publisher (THIS IS NOT INTERNAL URL!).

ISSN

list – List of ISSNs for the periodical.

invalid_ISSNs

list – List of INVALID ISSNs for this book.

nazev

str – Name of the periodical.

anotace

str – Anotation. Max lenght: 500 chars.

podnazev

str – Subname of the book.

id_number

str – Identification number in aleph.

mistoVydani

str – City/country origin of the publication.

datumVydani

str – Date of publication.

internal_url

str – Link to edeposit/kramerius system.

nakladatelVydavatel

str – Publisher's name.

ISSNSouboruPublikaci

list – ISSN links to other things.

static from_xml (*xml*)

Convert MARCXMLRecord object to *EPublication* namedtuple.

Parametry xml (*str/MARCXMLRecord*) – MARC XML which will be converted to *EPublication*. In case of *str*, <record> tag is required.

Vrací *EPublication* namedtuple with data about publication.

Typ návratové hodnoty structure

SemanticInfo structure

Definition of structures, which are used to hold informations about catalogization process.

class `aleph.datastructures.semanticinfo.SemanticInfo`

This structure is used to represent informations about export progress in Aleph.

It contains informations about state of the record, so it can be tracked from edeposit project.

See `toSemanticInfo()` for details of parsing of those attributes.

hasAcquisitionFields

bool – Was the record approved by acquisition?

acquisitionFields

list – Acquisition fields if it the record was signed.

ISBNAgencyFields

list – Was the record approved by ISBN agency? Contains list of signs if it the record was signed.

descriptiveCatFields

list – Did the record get thru name description (jmenný popis). Contains list of signs if it the record was signed.

descriptiveCatReviewFields

list – Did the record get thru name revision (jmenná revize). Contains list of signs if it the record was signed.

subjectCatFields

list – Did the record get thru subject description (věcný popis). Contains list of signs if it the record was signed.

subjectCatReviewFields

list – Did the record get thru subject revision (věcná revize). Contains list of signs if the record was signed.

isClosed

bool – Was the record closed? This sometimes happen when bad ISBN is given by creator of the record, but different is in the book.

isSummaryRecord

bool – Is the content of FMT == “SE”?

contentOfFMT

str, default “” – Content of FMT subrecord.

parsedSummaryRecordSysNumber

str – Same as `summaryRecordSysNumber` but without natural language details.

summaryRecordSysNumber

str – Identifier of the new record if `.isClosed` is True. Format of the string is not specified and can be different for each record.

static from_xml (*xml*)

Pick informations from `MARCXMLRecord` object and use it to build `SemanticInfo` structure.

Parametry xml (*str/MARCXMLRecord*) – MarcXML which will be converted to `SemanticInfo`.

In case of *str*, `<record>` tag is required.

Vrací `SemanticInfo`.

Typ návratové hodnoty `structure`

EPeriodicalSemanticInfo structure

Definition of structures, which are used to hold informations about catalogization process of periodical publications.

class `aleph.datastructures.eperiodical_semantic_info.EPeriodicalSemanticInfo`
 This structure is used to represent informations about export progress in Aleph.

It contains informations about state of the record, so it can be tracked from edeposit project.

hasAcquisitionFields

bool – Was the record aproved by acquisition?

acquisitionFields

list – Acquisition fields if it the record was signed.

isClosed

bool – Was the record closed? This sometimes happen when bad ISBN is given by creator of the record, but different is in the book.

isSummaryRecord

bool – Is the content of FMT == “SE”?

contentOfFMT

str, default “” – Content of FMT subrecord.

parsedSummaryRecordSysNumber

str – Same as *summaryRecordSysNumber* but without natural language details.

summaryRecordSysNumber

str – Identificator of the new record if *isClosed* is True. Format of the string is not specified and can be different for each record.

static from_xml (*xml*)

Pick informations from MARCXMLRecord object and use it to build *SemanticInfo* structure.

Parametry *xml* (*str/MARCXMLRecord*) – MarcXML which will be converted to *SemanticInfo*.
 In case of *str*, `<record>` tag is required.

Vrací *SemanticInfo*.

Typ návratové hodnoty *structure*

AlephRecord structure

Following structures are used to represent informations returned from Aleph.

API

class `aleph.datastructures.alephrecord.AlephRecord`
 This structure is returned as response to *SearchRequest* inside *SearchResult*.

base

str – Identity of base where this record is stored.

library

str – Library string, used for downloading documents from Aleph when you know proper *docNumber*.

docNumber

str – Identificator in Aleph. It is not that much unique as it could be, but with *library* string, you can fetch documents from Aleph if you know this.

xml

str – MARC XML source returned from Aleph. Quite complicated stuff.

parsed

namedtuple, default None – Parsed *xml* to *EPublication* structure in case of monographic / multimono publications, or *EPeriodical* in case of series.

semantic_info

namedtuple, default None – Export progress informations from *xml* attribute represented as *SemanticInfo* structure in case of monographic / multimono publications, or *EPeriodicalSemanticInfo* in case of series.

Poznámka: *semantic_info* and *parsed* attributes are parsed automatically from *xml* if not provided by user.

Request structures

Request structures, on which *aleph.reactToAMQPMessage()* reacts.

All structures defined here are simple dataholders, based on *namedtuple*.

class *aleph.datastructures.requests.CountRequest*

Put one of the Queries to *.query* property and result will be just the number of records, instead of records itself.

This helps to save some of Aleph resources (yeah, it is restricted to give too much queries by license).

query

Query object – *GenericQuery*, *ISBNQuery*, .. (*AnyQuery*) structure defined in *aleph* module.

Viz také:

aleph.reactToAMQPMessage() returns *aleph.datastructures.results.CountResult* as response.

class *aleph.datastructures.requests.SearchRequest*

Perform search in Aleph with given *query*.

query

Query object – *GenericQuery*, *ISBNQuery*, .. (*AnyQuery*) structure defined in *aleph* module.

Viz také:

aleph.reactToAMQPMessage() returns *aleph.datastructures.results.SearchResult* as response.

class *aleph.datastructures.requests.ISBNValidationRequest*

Validate given ISBN.

ISBN

str – ISBN, which will be validated.

Viz také:

aleph.reactToAMQPMessage() returns *aleph.datastructures.results.ISBNValidationResult* as response.

class *aleph.datastructures.requests.ExportRequest*

Request to export data to Aleph.

epublication

aleph.datastructures.epublication.EPublication structure, which will be exported to Aleph

Varování: *ISBN, nazev, Místo vydání, Měsíc a rok vydání, Pořadí vydání, Zpracovatel záznamu, vazba/forma, Formát (poze pro epublikace) and Nakladatel* has to be present, or `AssertionError` will be thrown.
 ISBN has to be valid, or request will be rejected with `ExportException`.

Viz také:

`aleph.reactToAMQPMessage()` returns `aleph.datastructures.results.ExportResult` as response.

Result classes

This module provides Result objects, that are sent back as answers to requests.

All classes defined here are just simple namedtuple data containers, without any other functionality.

class `aleph.datastructures.results.ISBNValidationResult`
 Response to `ISBNValidationRequest`.

is_valid
bool – True, if ISBN is valid.

class `aleph.datastructures.results.SearchResult`
 This is response structure, which is sent back when `SearchRequest` is received.

records
list – Array of `AlephRecord` structures.

class `aleph.datastructures.results.CountResult`
 This is returned back to client when he send `CountRequest`.

num_of_records
int – Number of records.

class `aleph.datastructures.results.ExportResult`
 Sent back as response to `ExportRequest`.

This class is blank at the moment, because there is no information, that can be sen't back.

ISBN
str – ISBN of accepted publication.

1.3 Aleph's lowlevel API

1.3.1 Propojeni do katalogu NK ČR

Kódy pro polí pro vyhledávání

- **wrd** - slova ze všech popisných údaju (název, autoři, rok vyd., nakladatel, edice, klíčová slova atd.)
- **wtl** - slova z názvových údajů (název, název části, edice, originál atd.)
- **wau** - slova z údajů o autorech
- **wpb** - slova z údajů o nakladateli
- **wpp** - slova z údajů o místě vydání

- **wyr** - rok vydání
- **wkw** - předmět (klíčová slova)
- **sbn** - ISBN/ISMN
- **ssn** - ISSN
- **icz** - identifikační číslo záznamu
- **cnb** - číslo ČNB
- **sg** - signatura

Jedná se o kódy formulářů, tak jak by je bylo možné vybrat na webových stránkách.

Vyhledání záznamů – podle ISBN

<http://aleph.nkp.cz/X?op=find&request=sbn=978-80-7367-397-0&base=nkc>

Vyhledání záznamů

Podle autora a názvu

[http://aleph.nkp.cz/X?op=find&request=wau=\(Milan+Hejny\)+and+wtl=\(Dite+matematika\)&base=nkc](http://aleph.nkp.cz/X?op=find&request=wau=(Milan+Hejny)+and+wtl=(Dite+matematika)&base=nkc)

(problémy s diakritikou, interpunkci, některými spojkami atd.)

[http://aleph.nkp.cz/X?op=find&request=wau=\(Milan+Hejn%C3%BD\)+and+wtl=\(D%C3%ADt%C4%9B+a+matematika\)&base=nkc](http://aleph.nkp.cz/X?op=find&request=wau=(Milan+Hejn%C3%BD)+and+wtl=(D%C3%ADt%C4%9B+a+matematika)&base=nkc)

Podle nakladatele

[http://aleph.nkp.cz/X?op=find&request=wau=\(Hejny\)+and+wtl=\(matematika\)+and+wpb=Portal&base=nkc](http://aleph.nkp.cz/X?op=find&request=wau=(Hejny)+and+wtl=(matematika)+and+wpb=Portal&base=nkc)

Nakladatel Grada

<http://aleph.nkp.cz/X?op=find&request=wpb=Grada&base=nkc>

Případně pouze knihy

[http://aleph.nkp.cz/X?op=find&request=wau=\(Hejny\)+and+wtl=\(matematika\)+and+wtp=bk&base=nkc](http://aleph.nkp.cz/X?op=find&request=wau=(Hejny)+and+wtl=(matematika)+and+wtp=bk&base=nkc)

a další možné kombinace...

Odpověď

Server vrátí odpověď, kde je uveden počet nalezených záznamů a kód množiny výsledků. Následně je možné stáhnout jeden nebo více vyhledaných záznamů v XML (formát OAI-XML)

```
http://aleph.nkp.cz/X?op=present&set_number=<kód množiny výsledků>&set_entry=1
https://aleph.nkp.cz/X?op=present&set_number=39EUFDPANCBL5134N9UPNMPHFDEXPR2LR32GMP99R85TACTN5U
```

resp.:

```
http://aleph.nkp.cz/X?op=present&set_number=<kód množiny výsledků>&set_entry=5,6,11
```

resp.:

```
http://aleph.nkp.cz/X?op=present&set_number=<kód množiny výsledků>&set_entry=1-25
http://aleph.nkp.cz/X?op=present&set_number=077285&set_entry=1-25
```

Průběh celé session

Dotaz na knihy vydavatele:

```
http://aleph.nkp.cz/X?op=find&request=wpb=Grada&base=nkc
```

Což vrátí:

```
<find>
<set_number>016513</set_number>
<no_records>000004998</no_records>
<no_entries>000002500</no_entries>
<session-id>9A5HGMD6SMHBBX6NFDGTS6JD1GNK9JUFQGQENM28Y6EDTL4RABD</session-id>
</find>
```

Nyní získáme kódy dokumentů (přes url http://aleph.nkp.cz/X?op=present&set_number=016513&set_entry=1-100 anebo ve formátu marcxml http://aleph.nkp.cz/X?op=ill_get_set&set_number=016513&start_point=1&no_docs=500)

Použijí url http://aleph.nkp.cz/X?op=ill_get_set&set_number=016513&start_point=1&no_docs=2

```
<ill-get-set>
<doc-number>002475050</doc-number>
<doc-number>002468257</doc-number>
<no-docs>000000002</no-docs>
<set-library>NKC01</set-library>
<session-id>F3A9P757D16J84CULSGBTYSMDMYUXDDGDD8R6EEJIIJK9YDV4H</session-id>
</ill-get-set>
```

Povšimněte si položek <doc-number>. Čísla z nich je možné nyní využít pro stažení samotných záznamů.

Stažení záznamů (formát MARCXML)

Funkce `ill_get_set` a `ill_get_doc`. Stažení záznamů (po jednom) – např.:

- http://aleph.nkp.cz/X?op=ill_get_doc&doc_number=001810391&library=nkc01

Stažení záznamů (formát MAR OAI)

Stažení záznamu podle systém. čísla (funkce `find_doc`):

```
http://aleph.nkp.cz/X?op=find_doc&doc_num=0018103916&base=nkc
```

nebo:

```
http://aleph.nkp.cz/X?op=find_doc&doc_number=001810391&base=NAK
```

Funkce `publish_avail`

Zjištění aktuální dostupnosti jednotek k danému záznamu (může být zadáno až 10 systém. čísel oddělených čárkami)

```
http://aleph.nkp.cz/X?op=publish_avail&doc_num=002107662&library=nkc01
```

resp.

http://aleph.nkp.cz/X?op=publish_avail&doc_num=002107662,002124258,002105616&library=nkc01

odpověď v češtině

http://aleph.nkp.cz/X?op=publish_avail&doc_num=002107662,002124258,002105616&library=nkc01&con_lng=cze

Nastavení v tab_expand

X-AVAIL	expand_doc_bib_avail	AVA=DG, ZL, ND, RD
X-AVAIL	expand_doc_del_fields	100##, 245##, 250##, 260##, 300##, AVA##

Standardně, je-li vyplněn status zprac. Jednotky, je jednotka nedostupná. To je možné změnit přidáním parametru AVA=..., kde se uvedou všechny statusy, které dostupnost neblokují.

Indices and tables

- [genindex](#)
- [modindex](#)
- [search](#)

a

- aleph, 3
- aleph.aleph, 9
- aleph.datastructures, 19
- aleph.datastructures.alephrecord, 23
- aleph.datastructures.author, 19
- aleph.datastructures.eperiodical, 21
- aleph.datastructures.eperiodical_semantic_info,
23
- aleph.datastructures.epublication, 19
- aleph.datastructures.format_enum, 19
- aleph.datastructures.requests, 24
- aleph.datastructures.results, 25
- aleph.datastructures.semanticinfo, 22
- aleph.export, 17
- aleph.settings, 18

A

acquisitionFields (atribut aleph.datastructures.eperiodical_semantic_info.EPeriodicalSemanticInfo), 23
 acquisitionFields (atribut aleph.datastructures.semanticinfo.SemanticInfo), 22
 aleph (modul), 3
 aleph.aleph (modul), 9
 aleph.datastructures (modul), 19
 aleph.datastructures.alephrecord (modul), 23
 aleph.datastructures.author (modul), 19
 aleph.datastructures.eperiodical (modul), 21
 aleph.datastructures.eperiodical_semantic_info (modul), 23
 aleph.datastructures.epublication (modul), 19
 aleph.datastructures.format_enum (modul), 19
 aleph.datastructures.requests (modul), 24
 aleph.datastructures.results (modul), 25
 aleph.datastructures.semanticinfo (modul), 22
 aleph.export (modul), 17
 aleph.settings (modul), 18
 ALEPH_DEFAULT_BASE (v modulu aleph.settings), 18
 ALEPH_EXPORT_URL (v modulu aleph.settings), 18
 ALEPH_URL (v modulu aleph.settings), 18
 AlephException, 11
 AlephRecord (třída v aleph.datastructures.alephrecord), 23
 anotace (atribut aleph.datastructures.eperiodical.EPeriodical), 21
 anotace (atribut aleph.datastructures.epublication.EPublication), 20
 Author (třída v aleph.datastructures.author), 19
 AuthorQuery (třída v aleph), 6
 autori (atribut aleph.datastructures.epublication.EPublication), 20

B

base (atribut aleph.aleph.DocumentID), 11
 base (atribut aleph.datastructures.alephrecord.AlephRecord), 19

23

BASE_PATH (v modulu aleph.settings), 18

BROZ (atribut aleph.datastructures.format_enum.FormatEnum),

19

C

castDil (atribut aleph.datastructures.epublication.EPublication), 20

CD (atribut aleph.datastructures.format_enum.FormatEnum), 19

cena (atribut aleph.datastructures.epublication.EPublication), 20

contentOfFMT (atribut aleph.datastructures.eperiodical_semantic_info.EPeriodicalSemanticInfo), 23

contentOfFMT (atribut aleph.datastructures.semanticinfo.SemanticInfo), 22

CountRequest (třída v aleph.datastructures.requests), 24

CountResult (třída v aleph.datastructures.results), 25

D

datumVydani (atribut aleph.datastructures.eperiodical.EPeriodical), 21

datumVydani (atribut aleph.datastructures.epublication.EPublication), 20

DEFAULT_LIBRARY (v modulu aleph.settings), 18

descriptiveCatFields (atribut aleph.datastructures.semanticinfo.SemanticInfo), 22

descriptiveCatReviewFields (atribut aleph.datastructures.semanticinfo.SemanticInfo), 22

docNumber (atribut aleph.datastructures.alephrecord.AlephRecord), 23

DocumentID (třída v aleph.aleph), 11

DocumentNotFoundException, 11

DocumentQuery (třída v aleph), 6

downloadMARCOAI() (v modulu aleph.aleph), 13

downloadMARCXML() (v modulu aleph.aleph), 12

downloadRecords() (v modulu aleph.aleph), 12

DVD (atribut aleph.datastructures.format_enum.FormatEnum), 19

E

EDEPOSIT_EXPORT_REFERERER (v modulu aleph.settings), 18

EDEPOSIT_EXPORT_SIGNATURE (v modulu aleph.settings), 18

EPeriodical (třída v aleph.datastructures.eperiodical), 21

EPeriodicalSemanticInfo (třída v aleph.datastructures.eperiodical_semantic_info), 23

epublication (atribut aleph.datastructures.requests.ExportRequest), 24

EPublication (třída v aleph.datastructures.epublication), 19

exportEPublication() (v modulu aleph.export), 17

ExportException, 17

ExportRejectedException, 17

ExportRequest (třída v aleph.datastructures.requests), 24

ExportResult (třída v aleph.datastructures.results), 25

F

firstName (atribut aleph.datastructures.author.Author), 19

format (atribut aleph.datastructures.epublication.EPublication), 20

FormatEnum (třída v aleph.datastructures.format_enum), 19

from_xml() (statická metoda aleph.datastructures.eperiodical.EPeriodical), 21

from_xml() (statická metoda aleph.datastructures.eperiodical_semantic_info.EPeriodicalSemanticInfo), 23

from_xml() (statická metoda aleph.datastructures.epublication.EPublication), 20

from_xml() (statická metoda aleph.datastructures.semanticinfo.SemanticInfo), 22

G

GenericQuery (třída v aleph), 6

get_all_constants() (v modulu aleph.settings), 18

get_POST_data() (metoda aleph.export.PostData), 17

getAuthorsBooksCount() (v modulu aleph.aleph), 16

getAuthorsBooksIDs() (v modulu aleph.aleph), 15

getAuthorsBooksXML() (v modulu aleph.aleph), 14

getBooksTitleCount() (v modulu aleph.aleph), 16

getBooksTitleIDs() (v modulu aleph.aleph), 15

getBooksTitleXML() (v modulu aleph.aleph), 14

getCountResult() (metoda aleph.DocumentQuery), 6

getDocumentIDs() (v modulu aleph.aleph), 12

getICZBooksCount() (v modulu aleph.aleph), 16

getICZBooksIDs() (v modulu aleph.aleph), 15

getICZBooksXML() (v modulu aleph.aleph), 14

getISBNCount() (v modulu aleph.aleph), 15

getISBNsIDs() (v modulu aleph.aleph), 14

getISBNsXML() (v modulu aleph.aleph), 13

getISSNsXML() (v modulu aleph.aleph), 13

getListOfBases() (v modulu aleph.aleph), 11

getPublishersBooksCount() (v modulu aleph.aleph), 16

getPublishersBooksIDs() (v modulu aleph.aleph), 15

getPublishersBooksXML() (v modulu aleph.aleph), 14

getSearchResult() (metoda aleph.DocumentQuery), 6

H

hasAcquisitionFields (atribut aleph.datastructures.eperiodical_semantic_info.EPeriodicalSemanticInfo), 23

hasAcquisitionFields (atribut aleph.datastructures.semanticinfo.SemanticInfo), 22

I

ICZQuery (třída v aleph), 7

id (atribut aleph.aleph.DocumentID), 11

id_number (atribut aleph.datastructures.eperiodical.EPeriodical), 21

id_number (atribut aleph.datastructures.epublication.EPublication), 20

internal_url (atribut aleph.datastructures.eperiodical.EPeriodical), 21

internal_url (atribut aleph.datastructures.epublication.EPublication), 20

invalid_ISBN (atribut aleph.datastructures.epublication.EPublication), 20

invalid_ISSNs (atribut aleph.datastructures.eperiodical.EPeriodical), 21

InvalidAlephBaseException, 11

InvalidAlephFieldException, 11

InvalidISBNException, 17

is_valid (atribut aleph.datastructures.results.ISBNValidationResult), 25

ISBN (atribut aleph.datastructures.epublication.EPublication), 20

ISBN (atribut aleph.datastructures.requests.ISBNValidationRequest), 24

ISBN (atribut aleph.datastructures.results.ExportResult), 25

ISBNAgencyFields (atribut aleph.datastructures.semanticinfo.SemanticInfo), 22

ISBNQuery (třída v aleph), 6

ISBNSouboruPublikaci (atribut aleph.datastructures.epublication.EPublication), 20

ISBNValidationRequest (třída v aleph.datastructures.requests), 24

ISBNValidationResult (třída v aleph.datastructures.results), 25

isClosed (atribut aleph.datastructures.eperiodical.SemanticInfo), 23	EPeriodicalSemanticInfo),
isClosed (atribut aleph.datastructures.semanticinfo.SemanticInfo), 22	parsed (atribut aleph.datastructures.alephrecord.AlephRecord), 24
ISSN (atribut aleph.datastructures.eperiodical.EPeriodical), 21	parsedSummaryRecordSysNumber (atribut aleph.datastructures.eperiodical_semantic_info.EPeriodicalSemar 23
ISSNSouboruPublikaci (atribut aleph.datastructures.eperiodical.EPeriodical), 21	parsedSummaryRecordSysNumber (atribut aleph.datastructures.semanticinfo.SemanticInfo), 22
isSummaryRecord (atribut aleph.datastructures.eperiodical_semantic_info.EPeriodicalSemanticInfo), 23	podnazev (atribut aleph.datastructures.eperiodical.EPeriodical), 21
isSummaryRecord (atribut aleph.datastructures.semanticinfo.SemanticInfo), 22	podnazev (atribut aleph.datastructures.epublication.EPublication), 20 poradiVydani (atribut aleph.datastructures.epublication.EPublication), 20
L	PostData (třída v aleph.export), 17 PublisherQuery (třída v aleph), 7
lastName (atribut aleph.datastructures.author.Author), 19	
library (atribut aleph.aleph.DocumentID), 11	Q
library (atribut aleph.datastructures.alephrecord.AlephRecord), 23	query (atribut aleph.datastructures.requests.CountRequest), 24
LibraryNotFoundException, 11	query (atribut aleph.datastructures.requests.SearchRequest), 24
M	
MAPA (atribut aleph.datastructures.format_enum.FormatEnum), 19	reactToAMQPMessage() (v modulu aleph), 7
mistoVydani (atribut aleph.datastructures.eperiodical.EPeriodical), 21	records (atribut aleph.datastructures.results.SearchResult), 25
mistoVydani (atribut aleph.datastructures.epublication.EPublication), 20	S
N	searchInAleph() (v modulu aleph.aleph), 11
nakladatelVydavatel (atribut aleph.datastructures.eperiodical.EPeriodical), 21	SearchRequest (třída v aleph.datastructures.requests), 24 SearchResult (třída v aleph.datastructures.results), 25
nakladatelVydavatel (atribut aleph.datastructures.epublication.EPublication), 20	semantic_info (atribut aleph.datastructures.alephrecord.AlephRecord), 24
nazev (atribut aleph.datastructures.eperiodical.EPeriodical), 21	SemanticInfo (třída v aleph.datastructures.semanticinfo), 22
nazev (atribut aleph.datastructures.epublication.EPublication), 20	subjectCatFields (atribut aleph.datastructures.semanticinfo.SemanticInfo), 22
nazevCasti (atribut aleph.datastructures.epublication.EPublication), 20	subjectCatReviewFields (atribut aleph.datastructures.semanticinfo.SemanticInfo), 22
num_of_records (atribut aleph.datastructures.results.CountResult), 25	substitute_globals() (v modulu aleph.settings), 19 summaryRecordSysNumber (atribut aleph.datastructures.eperiodical_semantic_info.EPeriodicalSemar 23
O	summaryRecordSysNumber (atribut aleph.datastructures.semanticinfo.SemanticInfo), 22
ONLINE (atribut aleph.datastructures.format_enum.FormatEnum), 19	
originally (atribut aleph.datastructures.epublication.EPublication), 20	T title (atribut aleph.datastructures.author.Author), 19

TitleQuery (třída v aleph), 7

U

url (atribut aleph.datastructures.eperiodical.EPeriodical),
21

url (atribut aleph.datastructures.epublication.EPublication),
19

V

VALID_ALEPH_FIELDS (v modulu aleph.aleph), 10

VAZANA (atribut aleph.datastructures.format_enum.FormatEnum),
19

vazba (atribut aleph.datastructures.epublication.EPublication),
20

X

xml (atribut aleph.datastructures.alephrecord.AlephRecord),
23

Z

zpracovatelZaznamu (atribut
aleph.datastructures.epublication.EPublication),
20