The BAS Repository

1 Architecture and software requirements

The BAS repository architecture is based on a file system and consists of the following components:

- a public sector containing the landing pages and the metadata (CMDI files ISO 24622-1,[1]) of stored corpora and corpus sessions. The PHP landing pages are generated dynamically from the CMDI files,
- a limited-access sector containing the primary data. This area can be accessed by authorized users after Shibboleth authentification,
- an OAI-PMH endpoint providing metadata in CMDI (ISO 24622-1) and Dublin Core format,

  http://www.phonetik.uni-muenchen.de/cgi-bin/BASRepository/oaipmh/oai.pl?verb=Identify

- a user search SQLite database and interface.

The repository software is written in Perl and PHP. It requires a web server which is capable to run CGI and PHP scripts, an SQLite database, as well as tools for xml validation (xmllint), metadata transformation (xsltproc), and checksum calculation (md5sum). We use the OAI-PMH2 XMLFile File-based Data Provider.

2 Data access

The **landing pages** of corpora and corpus sessions can be accessed without authentification:

- by the respective URL:

  https://clarin.phonetik.uni-muenchen.de/BASRepository/Public/Corpora/ALC/ALC.1.php

- by the handle PID pointing to this URL

  http://hdl.handle.net/11858/00-1779-0000-0006-BDA1-5

The **metadata** can be accessed by humans or machines without authentification:

- adding the part identifier *@format=cmdi, @format=dc*, or *@format=olac* to the handle PID

  http://hdl.handle.net/11858/00-1779-0000-0006-BDA1-5@format=cmdi

- using the Get parameter *?format=cmdi, ?format=dc*, or *?format=olac*,

  https://clarin.phonetik.uni-muenchen.de/BASRepository/Public/Corpora/ALC/ses1006.1.php?format=cmdi

  The Dublin core (dc) and OLAC format are generated from the CMDI format via xslt. OLAC is available for corpora only, not for single corpus sessions.

- by means of content negotiation. The the accept header has to contain the pattern *application/x-cmdi+xml*

  wget –header='Accept: application/x-cmdi+xml' http://hdl.handle.net/11858/00-1779-0000-0006-BDA1-5

- via the metadata URL field of the verbose handle,

[1] www.iso.org/obp/ui/#iso:std:iso:24622:-1:ed-1:v1:en

handle.gwdg.de:8080/pidservice/read/view?pid=11858/00-1779-0000-0006-BDA2-3&metadata=1

The **primary data** can be accessed after successful authentification by humans or machines in different ways:

- via the resource links on the dynamically generated landing pages,
- via the EPIC Handle PID system by a part identifier: *@partID= myResourceProxyId*,

  http://hdl.handle.net/11858/00-1779-0000-0006-BDA2-3@partId=m_0000000001

- using the Get parameter *?partID=myResourceProxyId*, whereat *myResourceProxyId* is the *id* attribute value of the corresponding *ResourceProxy* element in the CMDI file.

  https://clarin.phonetik.uni-muenchen.de/BASRepository/Public/Corpora/ALC/ses1006.1.php?partId=m_0000000001

3 Ingest procedure

The automatic ingest procedure of a new corpus is implemented in Perl and is directed by a single configuration file specifying repository locations, template files, and xslt stylesheets. The user only has to provide the location of the corpus CMDI file. The procedure consists of the following steps:

1. The corpus and each of its sessions is represented by a CMDI file. The files are validated against their profiles, and the resource links are tested.
2. For the corpus, its sessions and all primary data files repository IDs (RID) are derived from the CMDI files. An RID is shared by all versions of a resource. Thus, it serves to compare already stored and incoming data in order to trigger update steps if necessary (see below).
3. For the corpus and its sessions EPIC handle Verbose PIDs are requested using *curl*. Generally, a PID is assigned to each version of a corpus and a session.
4. The CMDI files are copied to the public repository sector and updated, i.e. PIDs are inserted and resource links are modified. The primary data is copied to the limited-access sector.
5. A repository content table (RCT) is generated that stores locations, IDs, checksums and isPartOf relations for all items of a corpus. This table serves for regular checksum consistency checks, resource updates, and can be used for repository rebuilds.
6. The repository search database and its interface are updated.
7. At the OAI-PMH endpoint, the Perl Storable object containing the metadata information is updated.

In case a version of an incoming corpus or session CMDI is already stored in the repository, it is tested whether in the new CMDI primary data is deleted, inserted or substituted (by checksum comparison with the corresponding RCT entry). If yes, a new version of the primary data and all its parent objects is inserted, new PIDs are required for the new corpus and session versions, and the RCT is updated accordingly.

4 PID: Granularity and Usage

EPIC Handle PIDs are assigned on the corpus and recording session level. One PID is assigned to each corpus and session version, thus each new ingest as well as each corpus or session update is marked by a new PID. Ressources that belong to these repository objects can be addressed by part identifiers as described in section 2. The PIDs can be accessed by humans and machines by reading out the *ID* element of the corpus CMDI files, and the *PID* element of the session CMDI files. The CMDI file access is introduced in section 2. Furthermore, the PIDs are displayed in the Metadata section of the repository object's landing page.