The diagram on the left describes the extension of the BHL-Europe System architecture to a global scale taking into account other regional BHL repository systems, global identifiers between systems, and replication procedures between them.

The technical architecture within BHL-Europe is based on the Open Archival Information System (OAIS) reference model. In addition to the core elements, BHL-Europe developed two modules that fulfill the roles Content Provider and Consumer. The provider role is covered by an Ingest module and the Consumer is represented by a Portal module.

The Ingest module handles harmonizing the metadata of each provider to our common metadata standard as defined by the OLEF scheme.

The Portal is the user interface for all BHL-Europe users. Search and retrieval will be done within this component.

The actual implementation of the OAIS system was done using well proven solutions like Fedora Commons, Apache Solr and Drupal 7. Together they form the architecture of BHL-Europe.

Fedora Commons offers an excellent integration of Apache Solr using the gSearch interface. It allows automatic updating of the Solr Index whenever new objects are ingested into the Fedora Commons repository. Through XSLT mappings it is possible to define which content is passed to Solr for indexing.

Drupal7 provides the portal in order to access the information within the BHL-Europe archive. By incorporating standard modules like apachesolr it is possible to directly interface with the Solr instance which is therefor used to search and discover items within the Fedora Commons repository. Any additional information for items is received by inquiring the Fedora Commons instance directly through its SOAP API.

One major advancement compared to classic search portals is the extensive use of external webservices during the ingest process as well as during searching. By utilizing those services (like Catalogue of Life, uBio Taxon Finder, Virtual International Authority File, ...) it is not necessary anymore to worry about regular imports but instead received data is always up to date. By enriching the metadata during ingest and expanding the search terms it is possible to leverage the available information and harmonize the results.

In order to keep an acceptable response time all requests and responses to webservices are cached using Drupals built in caching API. Thus caching is handled automatically using established technologies.

Examples of improving search results are: automatic searching for synonyms, discovery of scientific names within OCR texts, searching for alternative...