The abc of ABCD :
the Reference Manual

Version 2.0

Egbert de Smet

The abc of ABCD : the Reference Manual: Version 2.0

Egbert de Smet

Publication date November 1, 2017

Abstract

This document aims at providing all relevant background information and instructions on how to use the integrated library and documentation management system 'ABCD'. Both the basic operations and the advanced management of the software are discussed, as are useful and necessary topics concerning the ISIS-software on which ABCD is based. In this new edition the new features of version 2.0 are added : the use of different flavors of CISIS, of Unicode,and the digital library capabilities as well as some additional modules (ODDS, LDAP, DSpaceBridge) and utilities (UTF8-conversion, loanobjects creation...) .

Table of Contents

List of Figures

List of Tables

Chapter 1. Introduction

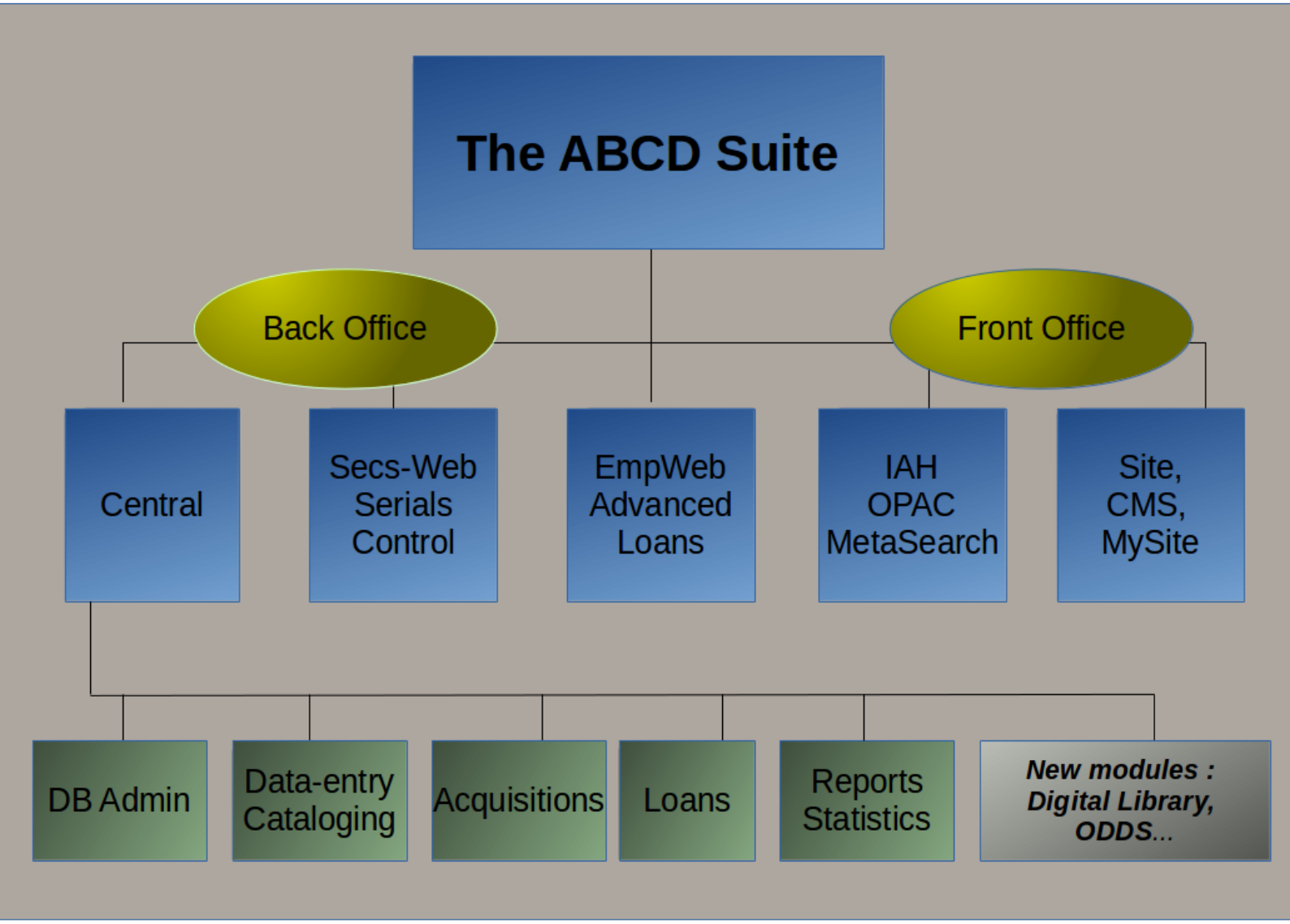
1. Background information

1.1. General introduction to ABCD as a software suite

ABCD is the acronym for a software suite for the automation of libraries and documentation centres. In Spanish this is, in full : '***A***utomatisación de **Bi**bliotécas y **C**entros de **D**ocumentación', which keeps the same acronym valid also for French (**A**utomation des **Bi**bliothèques et **C**entres de ***D***ocumentacion) or Portugese (**A**utomatização das **Bi**bliotecas e dos ***C***entros de **D**ocumentação). Even in other non-latin languages, with some slight but quite acceptable variations, - e.g. Dutch : '**A**utomatisering van **Bi**bliotheken en **C**entra voor **D**ocumentatie' - the acronym can still be maintained.

The name itself already expresses the ambition of the software suite : not only providing automation functions for the 'classic' libraries but also other information providers such as documentation centres. Flexibility and versatility are at the forefront of the criteria on which the software is developed. This flexibility e.g. is illustrated by the fact that in principle, but also practically, any bibliographic structure can be managed by the software, or even created by itself. Even non-bibliographic structures can be created, as long as the information is mainly 'textual' information, as this is the limitation put by the underlying database technology, which is the (CDS/)ISIS textual database. Good understanding of some basic ISIS-related concepts and techniques, e.g. the Formatting Language, is crucial for full mastering of the ABCD-software. For this reason some sections of this Manual will also deal with the underlying ISIS-technology.

ABCD is called a 'suite' of softwares for library and documentation centres automation because it exists of some relatively independent modules, which can fully co-operate but also can exist without each other. In fact some existing advanced softwares, mostly having already shown their potential in demanding environments in BIREME-applications (within the Virtual Health Library context), were adopted and adapted into ABCD - that is why the original names such as iAH, SeCS (both developed by BIREME) and EmpWeb (Empréstimos en Web) developed originally by KALIO ltda. of Uruguay and amply tested in Valparaiso at the University) are maintained. These main parts are shown, with their hierarchical relationships, at the second level in the following picture and subsequently discussed briefly :



1. **the ABCD Central**

   The 'Central' module of ABCD comprises modules for Database Administration (creation of databases, editing of database-structures, database utilities), Cataloging, Acquisitions, Circulation/Loans and Statistics. A thesaurus management module is also being prepared as part of the cataloging module for a specific thesaurus-structure database with consistency control of the hierarchical levels. As part of this 'central module' we would also like to mention import- and export services, printing and database-tools like blocking/unblocking and 'global changes' to fields in records. This 'Central' part in fact represents the 'back-office' part of ABCD, end-users will not be confronted with this but what they will see and be offered is fully defined in this central management part of the software !

   Any ISIS-database structure can be defined and managed, with currently records of 1Mb maximum size and 4Gb max, databases (but these restrictions will be made obsolete by the NBP-based next generation of ISIS and ABCD). As compared to 'normal' ISIS-technology ,60-character (as compared to 30-character) indexing keys are used, there are much stronger authority control features available (picklists based on tables or authority databases such as thesauri) at the data-entry stage with flexible validation formats) and all interaction is based on WWW-technology of course, allowing e.g. HTML-coded text-strings for full-text indexing, hyperlinks to help-pages etc.

It is perfectly possible to fully automate a smaller library with mostly internal users with all necessary functions, only using this Central part, as e.g. an advanced searching option is built-in, so that all functions are covered with a minimum of technological complexity (i.e. only ISIS and PHP).

2. t**he ABCD OPAC** (iAH)

   The public search interface (OPAC) is an adapted version of BIREME's general 'advanced interface for Health information' (iAH). It allows meta-searches on not only the local catalogs but also many other information resources.

   The iAH interface developed by BIREME is currently being upgraded to iAHx, ensuring it will align perfectly with modern Information Retrieval concepts and techniques (e.g. clustering, relevance ranking based on Lucene indexing).

3. **the ABCD Site**

   The search function is offered as part of an 'end-users' portal page, presenting the own catalog(s) in a much wider information context by providing access to other information resource.s (e.g. Google, Medline...) and communication (announcements, alerts), also paving the way for 'Web 2.0'-like functions.

   The Site Administrator actually is a specific Content Management System which allows designing the structure and components of the portal page of ABCD.

4. **the ABCD Serials Control System** (SeCS)

   This module offers an advanced management tool for serials/journals (classical and/or electronic) of any publication type (referring to periodicity). Serials as such but also issues of a serial and all types of publication patterns can be managed by this module. BIREME uses this technology e.g. for its products 'Portal of Scientific Journals' (see : http://portal.revistas.bvs.br/main.php?home=true&lang=en) and SCAD (see : http://scad.bvs.br/php/index.php?lang=en) which is the Brazilian union catalog of over 12.000 journals (with millions of issues) of more than 50 libraries.

5. **the ABCD Advanced Loans module** (EmpWeb)

   This module offers advanced loans management with some more extra features for larger and more complex organisations. It offers a 'MyLibrary' function to end-users through the OPAC and is based on 'web-services' technology. It can be used to replace the integrated loans module of ABCD in case of a need to cope with multi-branch/policy and very high transactions volume situations.

The 'suite' idea reflects the fact that ABCD has relatively independent parts - as is the case with office automation suites (e.g. Open Office, Microsoft Office) - but with obvious links to co-operate. The Statistics module e.g., as part of the Circulation/Loans module, can work on any ISIS-database, while the iAH-OPAC also can offer advanced web-based retrieval in any (set of) ISIS-databases, not only ABCD-maintained ones. The Serials Control System (SeCS) manages ISIS-databases for serials within or outside the ABCD-context. But together, we believe, these parts constitute a very powerful suite of tools, and as an integrated part we hope 'the sum is more than just the parts added up' !

1.2. The ISIS software family (history and overview)

In this paragraph we want to briefly introduce the wider 'ISIS software family' to which ABCD belongs. As with all 'families', members share a lot of characteristics but not all.

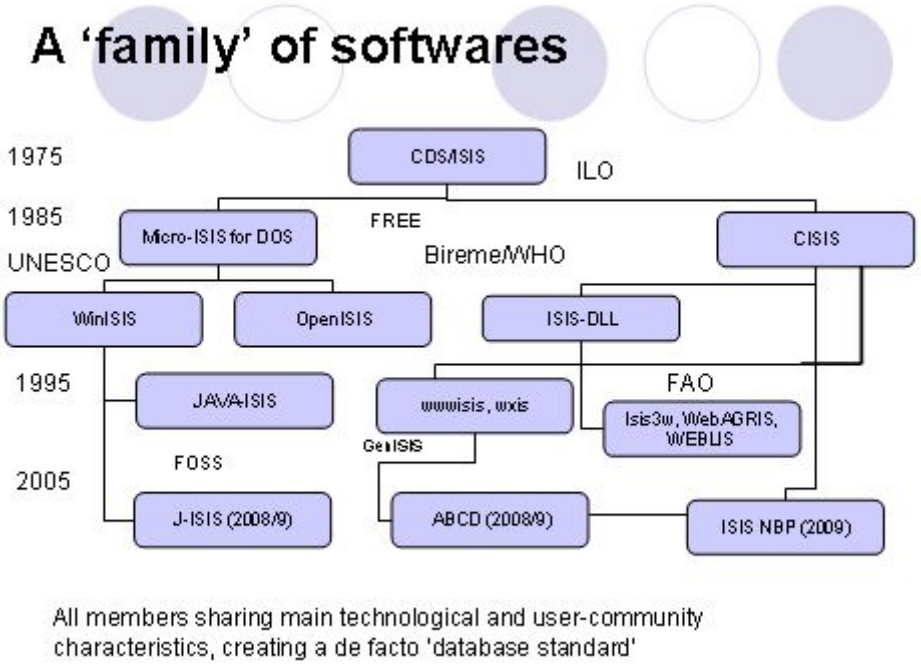
The common characteristics of the ISIS family relate to the way how information (of textual nature) is stored and managed by putting it in repeatable fields of variable length with the possibility of subdividing fields into subfields. Fields are in fact couples of a field-ID (a 'tag') combined with a field-value (a text, or in newer ISIS generation, any object, like e.g. 'binary large objects' or blobs).

In addition to technological common characteristics, most if not all ISIS family members share also 'social' characteristics, e.g.

- being mainly used in Developing Countries or 'the South', with e.g. a very strong presence in Latin-America, but also - more than can be 'measured' in all kinds of small, often deprived non-connected (no Internet) information centres in Africa and Asia.

- being promoted by many United Nation members and projects, of course first of all in UNESCO-environments, but - as shown by the BIREME example - also WHO and FAO (the AGRIS and ASFISIS systems of FAO can be given as examples here, but also the origin of the WEBLIS library system). The United Nations IFAP and 'Knowledge Society' programmes should not underestimate how much *real* impact comes from the UNESCO-promoted information tools like ISIS, IDAMS, Greenstone etc. - sometimes even indicating that the impact can be the reverse of given financial input or publicity.

The following illustration summarizes the full family up to now.



All members sharing main technological and user-community characteristics, creating a de facto 'database standard'

One could summarize the history by claiming the 'family' now has 4 generations while the 5th generation is being prepared :

- The first generation : CDS/ISIS and Micro-ISIS

- The second generation : enriched ISIS/Pascal interfaces, CISIS-tools

- The third generation : graphical, multimedia and multi-database : WinISIS, ISISDLL

- The fourth generation : WWW-enabled versions (wwwisis, isis3w, openisis…) .

In view of some major technological changes introduced in the newest generation as of 2008 one should perhaps consider the newest ISIS-members (J-ISIS and ISIS/NBP) as representing yet another new 5th generation.

Some highlights of each generation is given below.

1. 1975 - The first generation

   a. 1975 :

   CDS/ISIS at the International Labour Organisation (ILO) Centralised Documentation System merged with Integrated Set of Information Services Running on VAX OS on mainframes

   b. 1985

Micro-ISIS G. Del Bigio joins UNESCO and creates PC-DOS based version and integrates separated functions into one general customizable, multilingual menu-based interface with full documentation as version 2.3 Version 3.0 – 3.8 : networked multi-user, ISIS/Pascal UNIX-version for Intel-based UNIX OS World-wide distribution and huge success in Developing Countries

2. 1985 - The second generation

   - ISIS/Pascal programmed add-ons (e.g. Heurisko, ADEM, IRIS and ODIN, LAMP) create rich tools; e.g. IRBIS (Russia) for libraries, FAO uses ISIS for its AGRIS-system and ODIN/IRIS extensions for its ASFISIS-system
   - Bireme/OPS (WHO Brazil) creates CISIS-tools suite for command-line database management, uses it for its huge health information databases on the Internet; these are multi-platform (run on Unix/Linux and DOS)

3. 1995 - The third generation

   - UNESCO produces Windows version : WinISIS, with many graphical, multi-media and multi-database features
   - Full library automation systems can be and are developed, e.g. PURNA (India)
   - other libraries start using ISIS for full library automation, e.g. SNAL (Tanzania) uses networked ODIN/IRIS based library system for its university library
   - Bireme distributes a web-server version of ISIS as ‘wwwisis’ running on both DOS/Windows and UNIX/Linux; many applications are developed JavaISIS (Italy) and isis3w (Poland) added to the family

4. 2005 – The fourth generation

   - Advanced web-based tools spearhead further developments : GenISIS (France) allows easy creation of web-based search interfaces
   - WEBLIS (Poland/FAO) is a full-fledged advanced web-based library automation system
   - Bireme develops WXIS and adds XML to ISIS
   - WXIS-based library systems are developed in Latin-America (e.g. OpenMarcoPolo)
   - OpenISIS (Germany) creates first fully Open Source version (webserver, PHP-library) but goes its own way (Malete, Selene)

5. 2008 - The fifth generation

   - UNESCO developes a completely new Java-based graphical interface 'J-ISIS" using not only JAVA-technology but also the embedded Berkeley DB for the storage layer. This project is a fully FOSS-oriented project.
   - BIREME developes ABCD and - at the same time - a fully new technology for its future ISIS-products : ISIS/NBP. ABCD is meant to be the first applcation to be migrated into NBP.

     NBP or 'Network Based Platform' is the new ISIS technology with as the main characteristics :

     - flexible archtecture in which 'ISIS-cells' will communicate through known protocols with several platforms and interfaces; ISIS-cells will also allow to use different storage models as these will be contained within the cells but they behave in the same standardized way towards the external technology used;
     - ISIS databases will no longer have out-dated limitations re database-, record- and field-sizes;
     - ISIS databases will be UNICODE compatible
     - Indexing will be done by using other FOSS full-text indexers such as Lucene (from Apache Software Foundation).

ISIS is being used by ten-thousands of users, mostly in the Developing Countries where it is promoted by UNESCO and BIREME (for mostly Latin America). In Latin America ISIS is very strongly represented in libraries and documentation centres (it has a 'dominant' position even here), in Africa and Sout-East Asia there are an unknown but high number of users, many of them often non-connected to the internet and therefore still using older technology and with relatively poor ICT-skills. This creates a special challenge to the support of the users-community.

At the 3rd World Congress on ISIS (Rio de Janeiro, Brazil, September 2008) the Users Community decided to make ISIS fully 'FOSS' and co-ordinated by an 'International Co-ordination Committee on ISIS' (ICCI), see : http://portal.unesco.org/ci/en/ev.php-URL_ID=27760&URL_DO=DO_TOPIC&URL_SECTION=201.html

Summarizing the long history of ISIS, one could say that ISIS combines very sound basic 'textual database' principles, a strong tradition and a world-wide but insufficiently co-ordinated users' community with still modern state-of-the-art technological development.

1.3. From 'free' to 'FOSS'

Tip

You might be interested to read the full article on this topic, published at : '*Innovation*', no. 36 June 2008, p. 39-47.

CDS/ISIS as a software has been 'free' and 'open' since its early days, long before 'FOSS' (Free and Open Source Software) became a known software model (or should it be put in the reverse way : long before 'commercial closed software' became widely practised').

1.3.1. ISIS as 'open' software

Whereas ISIS, from the DOS-version produced and distributed by UNESCO as from 1985, always has been ‘free’ – i.e. without cost but with a restriction to the not-for-profit sectors only – the software was not ‘open’ in the strict meaning of the concept as nowadays known as ‘Open Source Software’ with its different definitions (see http://www.opensource.org/docs/osd) and licenses (e.g. (L)GPL, BSD, Creative Commons..).

But in 3 meanings there were, already from this beginning - and therefore long before the FOSS movement began to become really visible –, elements of being ‘open’ in addition to being free(ware) :

1. the standards were open and published. In the ‘CDS/ISIS Reference Manual’, written by its founding father Gianpaolo Del Bigio (working for ILO then UNESCO), the technical details were published in the annexes, allowing others to program their own versions of ISIS using the same compatible standards. E.g. in Slovakia Marek Smihla had programmed executables (e.g. ADEM for data-entry) which ran independently from the ISIS-executables from UNESCO and could write and read ISIS-records. Bireme in Sao Paulo, Brazil, did something similar : they programmed writing, reading and indexing tools with lots of advanced features (e.g. joining databases, linking them as relations etc…) in the C-language (therefore CISIS) which are still the basis for their other ISIS-related software : the DLL and the webservers (WWWISIS, WXIS) and which now have expanded capacity, e.g. 4 Gb max. database size, 1 Mb record size, 60 character-index keys. Co-operation was then set up with UNESCO, e.g. allowing the ‘CDS/ISIS for Windows’ to become a mix of UNESCO-programmed and Bireme-programmed modules.

2. an open, adjustable interface : the software itself was presented as a very flexible environment, with three main features which were used heavily all over the world not only to change its ‘interface’ but also the functions and features.

a. An open menu-structure : Micro-CDS/ISIS was fully based on menus which could be produced and changed by using the software itself, including the definition of ‘actions’ to be invoked by each menu option and allowing hierarchical sub-menus as well as dropping/adding options.

b. An open message system : all messages were/are based on small ISIS-databases which can be edited (each language having its own message-database) and expanded. This not only allowed (often together with the previous feature of open menu’s) creation of rather different conformations of the software – taking into account also colors and screen-features which could be changed – but also expansion and introduction of parameters (which could then be ‘read’ as messages) for additional software running inside ISIS (see further : ISIS/Pascal add-ons), as amply used e.g. by the cataloguing interface ‘ODIN’ and OPAC ‘IRIS’ (by the author of this article).

c. A programming tool ‘ISIS/Pascal’ which acted as an ‘API’ (with published calls for functions and their parameters) inside CDS/ISIS. ISIS/Pascal programmes, varying from a few lines to thousands of lines for sophisticated applications, could be included into the program either as ‘format exits’ (to expand the functions of the already very rich Formatting Language) or as ‘menu exits’ to expand the functions of the menus, allowing almost independent interfaces to ‘take over’ the CDS/ISIS environment in the creation and manipulation of its databases. One feature illustrating the ‘openness’ was the possibility of adding a parameter in the ‘SYSPAR.PAR’ initialization file to automatically invoke a menu and its option, therefore allowing the menu-interface to be skipped and immediately presenting the new ISIS/Pascal interface. In this way full OPAC (e.g. IRIS using a welcome-screen which could be invoked by a time-out mechanism after a previous session was left) and CD-ROM search modules (HEURISKO is an example) were written, loan-systems for libraries and thesaurus-management tools were produced.

d. Last but not least : the ‘open character’ of the Formatting Language. The Formatting Language is a grammar used to define in a detailed way how elements of the database-data, taken from repeatable fields and subfields, also from other records in the same or other databases (therefore resembling relational approaches) and with navigation links, will be ‘processed’ in some output (for display, sorting, printing, exporting). It was largely expanded with graphical features in the Windows version (RichText but also images and extra text- and image-boxes). Together these strong ‘data-processing’ and ‘presentation’ features of the Formatting Language have allowed the production of rather new ‘identities’ of the software, e.g. as a Library Management software with OPAC and Loans System (e.g. PURNA from India). In current applications, based on web-technology, the Formatting Language is still gracefully used to produce HTML-elements (e.g. links but also tables), even if more dedicated tools for that, e.g. PHP, are now added to the power of the own ISIS Formatting Language.

1.3.2. ISIS as full open source software

Already in 2001 UNESCO decided to embark on this relatively new approach of not only providing the software for free but also making the source codes in principle ‘open’, i.e. publicly available (see : http://portal.unesco.org/ci/en/ev.php-URL_ID=13803&URL_DO=DO_TOPIC&URL_SECTION=201.html). This has finally lead to a framework of its wider ‘Free and Open Source Portal’ approach promoting the idea and adding other softwares, e.g. Greenstone, into their ‘basket’ of supported and promoted softwares for better professional development also in the Southern and transitional countries. UNESCO’s FOSS Portal can be found at : http://www.unesco.org/cgi-bin/webworld/portal_freesoftware/cgi/page.cgi?d=1, with interesting links to discussions of the FOSS history, licenses and case studies. In reality however the source codes for existing ISIS-software are to be requested from UNESCO, but the new softwares will be fully available on public websites.

At Bireme/OPS/WHO a similar decision was taken in 2006/7. No longer would the institute charge a small fee for their software (as was the case before, e.g. 150 USD for official registration as a user with support rights) and therefore make it ‘free’, but also the sources have been and are still being prepared for publication of all their software, including the basic CISIS-modules. Their new ISIS-generation software, called ‘ISIS-NBP’ (Network Based Platform) will follow FOSS-methods (including a ‘community’ with possibilities to contribute, discuss and download sources at the URL http://reddes.bireme.br) to show their firm commitment to FOSS. As the newest full-fledged application, ABCD will be fully published as open source, even if the original development is still centrally managed by Bireme and its own programmers, as the project is now also supported by the Flemish Interuniversity Council (VLIR) with specific requirements to present it as a full competitor to other library systems (including other FOSS–softwares like KOHA and NewGenLib) and to this end needs some more central control for specific purposes.

The advantage of becoming fully open source – for all software - lies in the fact that users, certainly (programming) skilled ones, can fully check on the internal mechanisms and propose/make changes if so desired. One example: WinISIS has a slightly different way of sorting values taken by the ‘VAL’-function (i.e. removing padding 0’s first) which is not a bug as such and therefore does not ‘need’ to be corrected by the software provider; with access to the source codes one could change this however.

As is always the case with open source software, it would be best not to make such changes without consulting/informing the ‘developers’ community’.

1.4. Aims of ABCD

ABCD aims at providing an integrated library management tool covering all main functions in a library, i.e. acquisitions, bibliographic databases management, users management, loans management, serials control, end-user searching on local and external bibliographic databases and library portal.

it is not the first time in the ISIS-history and -environment that such effort has been undertaken. Open MarcoPolo, Clabel and - as a more advanced effort - WEBLIS are predecessors to ABCD in this sense.

- ABCD as a generic, flexible bibliographic tool

As the name itself suggests, ABCD however aims not only at providing a solution for libraries, but for documentation centres as well. These typically have slightly different needs, e.g. have more specialized collections, higher needs re contents disclosure (e.g. by providing abstracts, using thesauri etc.) and requiring more flexibility in the bibliographic structures. For this reason ABCD not only has tried to include full-text features but was *principally* conceived to offer a very open solution, allowing any fields structure to be created and maintained within the same software. By the database technology of ISIS itself, which is quite flexible and non-restrictive, bibliographic structures can be created without a need to 'normalise' all elements into a series of tables or relations (as is the case with relational database technology) and in most cases all bibliographic elements can be contained into one single database - only for optimization purposes ISIS would expect some semi-relational approaches to be implemented.

As a library system, however, ABCD comes pre-configured for some major bibliographic standards, i.e. MARC21, CEPAL and AGRIS. But we repeat : the same mechanisms, interface and forms can be used to create and maintain *any* structure, whether bibliographic or not.

So, to put the aims a bit more precise : ABCD aims at providing a very generic/generalizable tool for managing libraries and documentation centres.

- ABCD as a librarian-oriented tool

Another specific aim of ABCD is to offer a tool for librarians, rather than ICT technicians. This is achieved by taking library and information science principles (rather than computer or programming principles) as the starting point, even in the design of the databases themselves. Typically a bibliographic record is one real entity in an ISIS database, not a complicated series of elements 'queried' or 'joined' together from many tables (as in relational systems), however preserving criteria like efficiency (in space usage, speed of operation..). Each entity subsequently can be thoroughly 'moulded' by the librarians themselves with the use of the ISIS Formatting Language (FL), which allows dealing with all elements of an entity (e.g. a substring from a subfield of an occurrence of one specific field at micro-detail level) without real programming - even if the FL allows some degree of programming logics like loops and nested conditions - for the creation of any output format. This output can be anything like a sort key, an indexing key, a screen format or - as is the case in e.g. ABCD - ISIS-data embedded in web-pages or any other grammar such as XML. Lots of teaching experiences with ISIS show that librarians are perfectly capable of understanding and using all this, reaching advanced results without any real programming.

- ABCD as a tool for developing countries

  ABCD aims at providing librarians and information workers in developing countries a very powerful tool, which however takes into account some specific realities, such as :

  - low availability of ICT skills : as with previous ISIS-based solutions, librarians are - in principle - enabled to solve their problems by avoiding unnecessary software architectures while still allowing flexibility within the software (e.g. through the Formatting Language);
  - low availability of bandwidth and connectivity : by using modern web-techniques such as AJAX and JavaScript, data-traffic in between client and server is kept minimal, allowing the local computer (at the 'client-side') to process the data as much as possible without always referring to the server; also the graphical design is kept rather sober for the same reason.

1.5. Actors and partners of ABCD

ABCD, as with all major software projects, is a conglomerate effort of several actors and partners.

At the following URL a list of the main actors and partners is maintained :

http://reddes.bvsaude.org/projects/abcd/wiki/HallFame?version=20

The main input, obviously, comes from the Brazilian BIREME institute (see http://www.bireme.br), which has availed all of its ISIS-based technology to be combined into one 'culmination' product which is indeed ABCD. In

fact the original idea stems from its actual Director, Mr. Abel Packer, who has generously availed also worktime of his programmers and software managers.

A special mentioning certainly is appropriate for Mrs. Guilda Ascencio, Venezuela, who was the main programmer of the ABCD central part with its modules, based on her own 'Orbital Documental' software, in which she had proved that very advanced applications, combining library and other documentation management issues, could be built using ISIS and web-technology.

[!!] Programmers' generously availed by BIREME's director, mr. Abel Packer, have contributed since years to the development of ABCD iAH, Site and SeCS. These contributions represent uncountable and invaluable contributions to the ABCD software - and moreover they will continue developing the software as it will now also be used for BIREME's own projects, adding significantly to the future 'sustainability'.

Both the author of this book and mr. Ernesto Spinak, co-ordinator of the BIREME-based team, have acted as coordinators of the ABCD development project, trying to assemble the many pieces of the puzzle - and to make sure the final picture of the puzzle not only is more or less correct but also somehow attractive.

Two more institutional partners have to be mentioned as well :

- UNESCO : as explained above in the section about ISIS history, it is clear that UNESCO has an enormous merit in developing and promoting ISIS. ABCD will become part of the set of UNESCO-promoted ISIS products, but through a Memory of Understanding in between UNESCO and BIREME close technical supervision by BIREME will be assured.

- VLIR/UOS : the 'Development Co-operation' section of the Flemish Interuniversity Council (VLIR, Belgium, see http://www.vliruos.be), through a project '**D**evelopment **O**f and **C**apacity **B**uilding in **I**SIS-**B**ased **L**ibrary **A**utomation **S**ystems' (DOCBIBLAS) which is promoted by the Belgian co-author of this manual, has selected ABCD as the library automation solution it wants to promote with its partner university libraries in the South (Latin-America, Africa and South-East Asia).

2. ABCD technology

2.1. ISIS databases

ISIS databases are files in which information is contained in sequentially numbered records (MFN's or Master File Numbers) with values (mostly textual) stored in fields with a 'tag' (or numerical identifier) and subfields (with a one-character identifier). Subfields, fields and records all are variable-length and 'variable occurrence' varying from 0 (not present) to any higher number of occurrences, with a maximum depending on the ISIS-technology used but in the newest generation (in J-ISIS or Alpha-ISIS) without limit.

Records are structurally described in a 'header' for each record itself, instead of the usual table-header in relational databases. By doing so ISIS reflects more the concept of each record being a 'document' in its own right with its own document structure, like e.g. books, articles or web-pages indeed. Therefore we prefer to call ISIS a 'documentary database' in which documents are stored as a record with variable structure and length. This avoids complicated structures of 'normalized' relational structures, which are very efficient in storing highly structured data but less so for semi-structured textual data.

This means that the records themselves can be quite polymorph, meaning structurally different with any combinations of fields. In principle ISIS can handle bibliographic records along with user-data and transactional data (e.g. loans) in one single database, but because of 'semi-relational' capabilities (fast retrieval of any part of a record in any ISIS-database at run-time, i.e. by the Formatting Language creating the output without the need for these 'relations' to be pre-defined) typically ISIS-applications will use some few databases, e.g. in ABCD only 3 or 4 databases (one for bibliographic entities, one for users, one for transactions and possibly one for items) can allow to run a full library.

In the 'classic' ISIS technology [1]all variable-length records (with (sub-)fields containing the values) are stored in a 'Master' file (.MST) and record-positions are kept track of in the 'Cross-Reference' (.XRF) file, which can be seen

[1] 'classic' refers the technology of ISIS since its introduction in the 1970's before the introduction of J-ISIS and Alpha-ISIS.

as a 'first-order' normal index on the records in the database. New or even just edited records are always appended at the end of the Master file and references in the XRF are updated accordingly, necessitating some 'compacting' at times to get rid of deleted and/or inactive (versions of) records.[2]

All values specified by a 'Field Selection Table' (which uses the Formatting language, therefore allowing very flexible and powerful definition of selected elements), are included into a B-tree 'Inverted File', which can be seen as a 'dictionary' of terms with the exact 'address' (record, fieldtag, occurrence, position within occurrence) attached to them. This allows very efficient retrieval, including full-text based, of any element defined as being 'retrievable'. ISIS was one of the first databases to offer full-text, which became only popular decades later. This 'Inverted File' (or IF) has several components (with nodes .N01/.N02 and leaves .L01/L02 files) for efficient organization - because in certain applications with intensive indexing the IF can be even larger than the database file itself !

So typically ISIS-databases exist of some 10 files : a MST with XRF, the B-Tree Inverted File files and some definition tables for the fields, the data-entry form and the indexation.

Due to the memory types used in the 'XRF'-file, which is a fixed-format table acting as a list of all MFNs with their ID and location into the MST, the maximum number of records which can be hold in one ISIS-database is slightly over 16 million. On the other hand within this limit the software performs remarkably well without almost any noticeable performance loss at higher numbers.

All this is changing with the new database technologies introduced in 2009 with e.g. J-ISIS : Berkeley DB uses a different storage in separate files with the definitions incorporated into the main data-files. But basically the concept of 'tag-value' pairs (an identifier and a content), on which a powerful Formatting Language and field-based plus full-text indexing is applied, remain the core of ISIS-databases.

2.2. CISIS

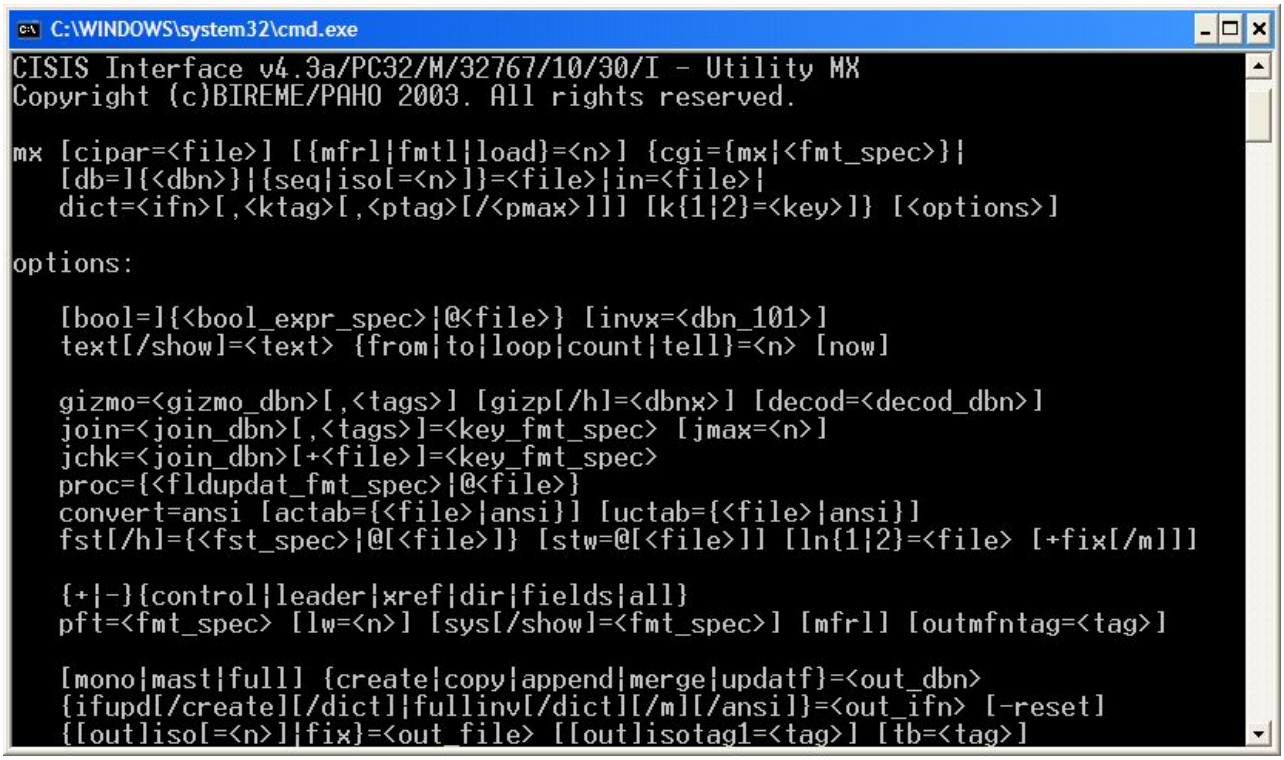
CISIS is the software developed by BIREME to handle ISIS databases from the Command Line in UNIX/Linux or DOS/Windows. This sofware has been written in the C-programming language and hence the name of this ISIS family member. CISIS mainly exists of a series of 'utilities', i.e. command-driven executables which perform all types of functions in ISIS-databases, like creating records, updating and searching them, updating the Inverted File, import and export and many other functions, sometimes unique in the 'ISIS Family', e.g. joining records from different databases according to common keys, indexing and searching from different Inverted Files for one database.

Actually CISIS as a set of utilities contains more than 25 different tools or executables. As this is not a manual on CISIS, we will not deal with all of them, but some are worth being mentioned, certainly also because we will use them for some off-line functions of ABCD.

2.2.1. The Master / Xross-reference tool : mx

The mx tool is the main CISIS utility, it could easily be baptised as 'CDS/ISIS for the command line', meaning most things which can be done with (M)asterfiles and (X)rf-files - therefore 'mx' indeed - with ISIS can also be done with MX. Just to give an idea we give the list of parameters mx accepts (as this list is given when invoking the command in a command-line environment such as the CMD-window in Windows or a terminal-window in UNIX/Linux. As one will see, too many parameters are available, meaning mx is an enormously powerful tool for ISIS-database management, but it deserves a manual and training in its own right !

[2]This behaviour, necessary because of the variable length of records, makes ISIS less suited for very dynamic databases, such as transactional applications (loans e.g.).



A glance at the many parameters show that mx can not only search ISIS-databases (bool=) but apply on-the-fly GIZMO (string-substitutions) and ANSI-conversion (ansi=), join fields of records from different databases but identified by their IF-entry (join= and jchk=), apply data-entry processes (proc=) and inverted file operations.

As CISIS comes in several varieties, according to the capacity of the databases and Inverted File keys intended, we need to specify that for ABCD we will only use the '16/60' variety of mx and other CISIS-tools. This can be verified from the information mx gives when invoked without any parameter as illustrated :

The most relevant uses of mx in this context of ABCD are :

1. import of ISO-records into an ISIS-database, e.g. the command :

   ```
   mx iso=myISOrecords.iso create=mydb now -all tell=100
   ```

   will read the file myISOrecords.iso and create an ISIS database 'mydb' without waiting for any user-input ('now'ait) and without showing any information on the screen (-all) but showing progress after every 100 records imported.

   ## Note

   In ABCD we use this to import a larger quantity of ISO-records into a database, as a high number and therefore long processing time would invoke the time-out of the web-server to stop the process.

2. index an ISIS-database, e.g. the command :

   ```
   mx mydb ifupd/create=mydb fst=@mydb.fst stw=@mydb.stw now -all
   ```

   will create an 'Inverted File' named 'mydb' using the mydb database with the indexing specifications given in the FST 'mydb.fst' and omitting the stopwords listed in mydb.stw, again without interactive mode or output (now -all).

Note

In ABCD we use this to create an index off-line in case - as is often the case - the database is r

2.2.2. Inverted File tools : mz, ifupd, ifkeys, ifload, ifmerge

These are more specialized tools to generate/update the ISIS Inverted File with its B-Tree technology and parts (leaves and nodes) from the command line with some more optimized speed and more options. E.g. MFN-ranges can be defined, keys can be taken from the previously created LK (link) files (ifload) or nodes-files (ifmerge) of the B-Tree, which can be balanced etc.

We don't normally need to use this with ABCD, but knowing the possibilities exist, especially in the case of very large databases, is certainly useful.

2.2.3. other CISIS-tools

Other tools to be briefly mentioned only are e.g. :

1. retag : this tool will change the tags of the fields according to a given specification - which can have instructions on many fields in one run

2. mfcrunch and ifcrunch : to convert the ISIS-files (resp. MST and the IF-files) from DOS/Windows to Unix and v.v.

3. mkxrf : to re-create the XRF-file for a given database, in case this is lost or got corrupted - the tool will analyze the MST-file and assign XRF-records into the XRF.

4. ctlmfn : to edit the values of the 'control-record' of the database, in which the maxMFN and other very technical values for the database are stored - for experts only !

2.3. ISIS Formatting Language

The ISIS Formatting Language (FL) is one of the most important parts of the software because it gives ISIS-managers the possibility to exactly define what ISIS will produce out of the databases at many stages of the software, e.g.

- what ISIS will show on the screen, i.e. 'present (defined in the Print Format Table or PFT)
- what ISIS will use for the creation of indexing keys (defined in the 3rd column of the Field Select Table or FST)
- what ISIS will use for sorting the records
- what ISIS will use as exported values (defined in the reformatting FST)
- what ISIS will use as values to validate input in fields (given in the validation tables).

2.3.1. The FL for presenting values

This is by far the most important function of the Formatting Language : specifying which data exactly need to be taken and how they will be 'displayed' or 'printed' (to the screen, to a printer, to a file, to a webpage...).

Separate documents exist to deal with this extensive language, e.g. the dedicated chapter in the ISIS Reference Manual, published by UNESCO (June 2004, chapter 8, p. 94-122).

Basically there are three types of statements in the ISIS FL :

1. values from fields, given as : Vx, where 'V' denotes the value (or 'contents') of a field with tag 'x', Vx^a is the value of the subfield a (^a) of field x and (Vx/) is the series of all occurrences of field X separated by a

'new-line' (/) since the parenthese embrace a 'repeatable group' of statements to be applied to all occurrences (repeatable fields are a strong special feature of ISIS).

2. literals or quotes strings, which can be 'unconditional' (single quotes), |conditional| (pipes indicate the string will only be produced if the related field is present) and "repeatable" (double quotes will only produce the string at the first occurrence of a repeatable field).
   ISIS-applications on the web, such as ABCD, create web-pages with HTML-tags using this method of adding literals to field-values, e.g.

   '<table><tr><td>' Vx '</td><td>' Vy '</td></tr></Table>'

   will display resp. the fields x and y in two columns of a table in HTML. Note that all HTML-codes are quoted (as unconditionals) and the values taken from the fields in the database are inserted by referring to them with the V-statement.

3. commands, which can be of different types, e.g. :

   - mode commands : mhl/u (mode heading lowercase/uppercase), mdl (mode data upper/lowercase) or mpl/u (mode proofreading upper/lowercase)

   - (in Windows-environments) : commands defining screen attributes (colors, fonts, boxes) or links (requesting the operating system to open other data, e.g. multimedia data referred to in a record), e.g.

     LINK('click here for tull-text', OPENFILE Vx) will request - when the user clicks on the hyperlinked text 'click here for full-text', Windows to open the file of which the name is in Vx, with the Windows-application associated to the extension of that file.

   - the REF-command, which can retrieve data from other records (in the same or another database when expressly referenced to), allowing semi-relations setups in ISIS-applications (but with the advantage that the relation is followed only at run-time when requested). e.g.

     REF(['users']) L(['users']V2),V1) will retrieve the value from field 1 in the database 'users' if the L(ookup) function has found the value of field 2 (in the actual database) in the index of the users-database, so that the MFN of the record can be identified.

   - conditional routing statements : e.g. 'IF...THEN... (ELSE....)FI' or even the 'SELECT [case1 case2...] ELSE-CASE... ENDSEL construct can be used to apply formatting statements only to database values which comply with given conditions.

   - in the CISIS-environment extra FL statements are available, the most important one being a command which will actually PROCess a record to alter the contents of the fields. The general syntax is :

     proc(x|y...) where x or y can be any of the following : 'Dxxx' (to delete field with tag xxx) - |Axx#|value|#| (to Add value into field xx)

   - functions, mostly for string-operations (e.g. substr, size, val) or numerical (e.g. rmin, rmax, rsum...)

   Full documentation on the Formatting Language is available, e.g. the 'CISIS Formatting Language' published by BIREME.

2.3.2. The FL for definition of indexing keys

The same formatting language, but of course without any appearance-related effects, can be used to exactly define which values should go into the Inverted File of ISIS. This will be defined in the third column of the 'Field Select Table' where the extraction format using the FL is to be used. See also the discussion of the FST definition in the chapter on database definition and management of this manual.

Since the full formatting language - except graphical elements - is available, the REF-function e.g. can be used to take into the Inverted File values different from the actual field contents, even from another database. This can e.g. be used to substitute codes for their full explanation or v.v.

2.3.3. The FL for definition of sorting keys

The same reasoning can be applied for the definition of keys which ISIS will use to sort records : again the actual sorting values can be processed values derived from the actual field values, by using the FL.

2.3.4. The FL for conversion during import/export

During import/export of records, most ISIS-applications will allow the use of a 'reformatting' FST, which has in its third column the exact definition of what to export/import, and in the first column (the 'IDentifier') the tag to be assigned to this value.

2.3.5. The FL for validation statements

The Formatting Language can also be used to create error messages in case defined conditions are (not) met. These conditions will be checked when passing data entered into a data-entry form into the record for storage. ABCD provides this technique by default as explained in the section on record validation. An example again can clarify this easily :

```
if a(Vx) then 'This field is mandatory, please check again !' fi
```

This statement will produce, on the screen, the message 'This field is mandatory, please check again' if the value of the field with tag x does not exist or is A(bsent).

More sophisticated statements can be used for more advanced quality/consistency checking, e.g. using a 'SELECT' construct, or even checking the value in another database (with the earlier discussed 'REF'-function) to see whether it is a valid entry.

2.4. ISIS Script

ISIS Script is a scripting language developed by BIREME in order to make stronger functions available to the ISIS webserver 'WWWISIS' for creation of pages with elements from ISIS-databases. ISIS Script in fact was one of the main elements in the stepping-up from WWWISIS to 'WXIS' which is the underlying web-server for ABCD.

ISIS Script scripts are stored as files with an extention .XIS. ABCD uses more than 100 such scripts, most of them in the php/dataentry/wxis folder but also iAH (the OPAC) makes extensive use of such scripts.

Obviously we cannot discuss the whole power of the ISIS Script language here. As a longuage it uses XML-like statements, e.g. in between the tags <pft> and </pft> a print format can be given and this format can be displayed by putting it in between <display> and </display> tags. All WXIS parameters can be defined within the <parm> and </parm> tags and fields can be defined with values, e.g.

```
<field action="replace" tag="6000">ValueOfField6000</field>
```

will put the string 'ValueOfField6000' into the field with tag 6000 (such high-value tags, in fact all tags above 999, are mostly used within ISIS-applications for temporary internal values which are not really stored in ISIS-records but rather 'virtual records'.

ISIS Script allows more flexible manipulation of data-elements, taken from ISIS-databases, in web-pages. In combination with PHP (see the dedicated section on PHP), which is a language for creation of web-pages powerful results are possible and this certainly adds to the general advanced functionality of ABCD.

Of course more details on the ISIS Script language can be found in the dedicated documentation.

2.5. J-ISIS

J-ISIS or J(ava)-ISIS is the current new technology in the ISIS-family, based on Java.

Longer ago a first attempt to create a java-based ISIS version was done by an Italian team. The result was a working solution to consult ISIS-databases remotely using java, although not very highly performant. This effort has stopped development after some years and is no longer maintained or available.

As from 2005 the UNESCO software-expert, mr. Jean-Claude Dauphin, mostly responsible for the IDAMS software maintained by UNESCO (for statistical management), started developing a fully new ISIS-version, no longer based on the until then unique 'MST/XRF' approach, but storing the data in a schemeless key-value database 'Berkeley DB' (see e.g. http://www.oracle.com/technetwork/products/berkeleydb/overview/index-085366.html), available still as FOSS from Oracle. For the search-engine and indexing the proven technology of Lucene was used from the Apache Software Foundation.

The real ISIS-technology of the concepts of variable-length records, fields and subfields with the ISIS Formatting Language (PFT) used not only in the output-presentation but also in the Field Selection Table (to define the exact string to be taken into the index) is still present and actually makes J-ISIS still being ISIS. J-ISIS by the way is the first version which contains a PFT-parser for grammar-checks and assistance.

The use of Berkeley DB means that no longer any limits are imposed upon record lengths (in traditional ISIS up to 32Kb with an extension possibility in CISIS up to 1Mb) or database-sizes.

The use of Lucene indexing technology means that not only all previously indexing techniques remain available but also 'ranking' now is added, making J-ISIS more suitable for e.g. full-text applications. A 'digital library' demo-database, based on the concept of using the TIKA-library for text-extraction of document formats by simply loading a document into a text-field, is included with the distribution package of J-ISIS.

At least once a year a new update is made available, before through kenai.com, nowadays (as from April 2017) at the URL https://github.com/J-ISIS/J-ISIS .

The 'new generation' ABCD (version 3.x) will be based on J-ISIS and is currently being developed by a team at 'Universidad de Ciencias Informáticas' (UCI) in Havana, Cuba.

2.6. PHP

PHP is a 'Hypertext Preprocessing' language, which means it is a programming language for web-pages. As one of the successful 'FOSS' products it is nowadays very popular and wide used, often in combination with Apache and MySQL databases. This has even lead to packages such as 'EasyPHP' and 'WAMP' (Windows, Apache, MySQL and PHP) which allow to install these often-combined softwares into one package.

As usual there are some criticisms on PHP as a language, but a fact is that it is very popular and getting more powerful with each release. ABCD uses e.g. also 'controls' or ready-available modules for specific functions, which are freely available.

ABCD 2.0 is compatible with the current versions of PHP, i.e. 5.x and 7.x. No longer - as in ABCD 1.x - the parameter 'short_open_tag' needs to be switched on, but a few non-standard 'extensions' need to be activated (in php.ini) : gd2, xsl, yaz, xmlrpc, and if the related functions are used : ldap, mysqli (for EmpWeb) and mbstring (for Unicode).

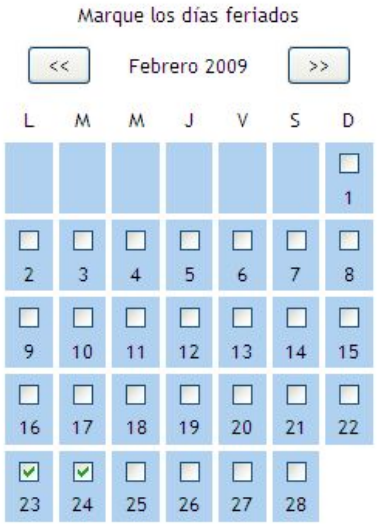
2.7. JavaScript

The official name of JavaScript is 'ECMA Script' but JavaScript is the popular name of a technology which is nowadays used in many web-pages : relatively small programmes embedded into the HTML-codes of the pages. Contrary to the name the language is not really linked to the JAVA Programming Language. JavaScript nowadays is supported by all up-to-date web-browsers and does not need any extra software or configuration. However it remains to be an option which can also be switched off (in Firefox e.g. : Tools|Options|Content, where both JavaScript and Java can be disabled), so make sure that the JavaScript option is enabled for the use of ABCD.

ABCD uses JavaScript 'scripts' inside its pages in very many instances, one reason being that by doing so the local computer can process data without a need for high traffic in between the server and the client (which is important in slow connectivity conditions).

As an example of a simple JavaScript we can refer to the script 'lrtrim.js' (in the ABCD-folder \ABCD\www \htdocs\php\dataentry\js\) which is called upon from several ABCD-PHP pages. The script trims white-spaces at the right or at the left side from strings. This can be easily done locally, no need for sending the string to the server together with the request to trim it and then having it returned from the server. Therefore the script is loaded into an ABCD page and executed locally.

Also generally available JavaScript existing modules are being used, e.g. for the calendar function in the Loans module or for the 'HTML Editor' (FCKEditor.js). Under here the calendar example is shown, based on the JavaScript 'popcalendar.js' which can be found e.g. in the folder php/loans/js of the ABCD home-folder (/ABCD/www/htdocs). This little tool displays any month of the calendar and allows marking the holidays to take them into account when calculating the loans period !



Most JavaScript functions however are not visible on the screen, but perform useful functions within the web-pages of ABCD. So even if tools like the above mentioned (the HTML editor or the calendar) are considered unnecessary, still it is important to keep the option to run JavaScript within your browser 'on'. As with Java, this option e.g in Firefox can be checked in the Tools|Options|Content tab (in Internet Explorer one has to activate 'Enable for Active Scripting' in the Scripting section of the security zone 'Internet' under Tools|Options|security).

2.8. JAVA, Groovy and Jetty

JAVA is at the same time a programming language (like e.g. 'C++') and a 'runtime compiler', which means that programs written in JAVA need a 'RunTime Environment' (RTE) version of JAVA which will compile the program for the given Operating System and CPU combination at run-time (i.e. when the user executes the program). By doing so the JAVA programs are completely 'multi-platform' (Windows, UNIX, Linux, OS/X...) because such RTE's exist for all platforms and are freely available for installation. So make sure your computer has its own JAVA RTE installed ! Both Sun (the real Java promoter) and Microsoft offer free versions of Java (e.g. at http://java.com/en/download). JAVA is not only free but also 'Open Source' and therefore can be reported as being fully 'FOSS', as is ABCD.

ABCD uses JAVA only for the 'advanced' loans module, which comes as an extra option (see the chapter on the Circulation module). This advanced circulation management module is intended only for larger institutions with more complex circulation rules and multiple branches with their own loans policies or with user-databases in other formats (e.g. SQL). Also more interactive 'MyLibrary' -style functions can be offered. In order to allow such more complicated software-combinations, ABCD calls upon JAVA to provide web-services and links with other database-models.

Groovy is an object-oriented programming language for the Java Platform which can be used as a scripting language for the Java Platform.

The advanced ABCD Loans module (EmpWeb) also uses Jetty-technology, which is aHTTP Server and Servlet container written in Java.

Jetty can be used as:

- a stand-alone traditional web server for static and dynamic content
- a dynamic content server behind a dedicated HTTP server such as Apache using mod_proxy

- an embedded component within a java application

2.9. MySQL

MySQL is a relational database developed as FOSS but with a 'dual license' scheme, allowing both commercial and free applications. Currently MySQL has been taken over by Sun Microsystems, a strong defender of FOSS software, e.g. JAVA. Recently Sun Microsystems has been taken over by Oracle, so the future is not so clear.

As a database MySQL has become incredibly popular because of its ease of use and combined packing with e.g. Apache and PHP for easy deployment of database-driven websites.
Examples of such pre-packaged combinations of Apache/PHP with MySQL are : EasyPHP (http://www.easyphp.org) and WAMP for Windows or XAMP for Linux (http://www.wampserver.com). Both are Open Source and free to use (GPL license).

Critics claim its 'relational' qualities are still lagging behind - even if improved a lot since the earlier days - as compared to e.g. PosGreSQL or of course the major relational databases like Oracle or IBM DBII. Some other library automation packages are completely using MySQL for the databases, the best known being KOHA (albeit that KOHA currently envisages adding/changing to another type of database, i.e. 'Zebra', exactly to avoid limitations of MySQL for library purposes.

The 'SQL' part of the name means 'Standard Query Language', denoting a standard grammar for retrieving data out of relations (related tables), heavily relying however on its relational structure. For this reason e.g. ISIS is not using SQL as it does not store its data into tables with fixed cells and structures.

MySQL will only be used in ABCD within the 'Advanced Loans' module, which is a non-standard extra (see the chapter on the loans module in this manual). There it will be used to store the transactions of the loans system, as these are administrative data which can be more efficiently handled by this type of database as compared to ISIS with all its - in this case unnecessary - flexibility and text-oriented features.

2.10. YAZ

YAZ is a freely available software for embedding the Z39.50 protocol in applications.

Z39.50 is used as a protocol to retrieve data from other catalogues, mostly in MARC-format.

ABCD uses YAZ for its 'Z39.50' function in the cataloging module.

2.11. Apache

Apache is the name of the webserver software very frequently used in 'open source' webservers. In fact we are talking about a software called 'HTTPD', which is only one product of the powerful 'Apache Software Foundation', which also provides other interesting products such as e.g. Lucene indexing (also going to be used in the next releases of ABCD), TomCat (a Java Servlet and Server Pages server) and the Derby DB.

Apache as a webserver seems to be the most widely used in the actual Internet, which is one of the (few) examples where FOSS dominates over the commercial solutions offered. All information on the Apache web-server and download files can be found at the URL : http://www.apache.org.

In many cases the Apache webserver software will already be installed on the server where ABCD will reside, as is probably also the case with PHP (and MySQL). For this reason ABCD came as a package both inclusive of Apache and PHP and another one without these, but in ABCD v2.0 only the non-inclusive package is available, leaving the Apache and PHP installation apart - good installers like WAMP and XAMP exist anyway also as FOSS. In the case of an existing Apache webserver, expertise on Apache should be available in order to integrate ABCD with the existing Apache-based applications. E.g. a virtual server for ABCD could be set up with 'aliases' specifically for the ABCD system (htdocs home) and cgi (scripts folder). In the case of the full package, as it came for ABCD 1.x, the latest 'stable' Apache httpd-version was included, pre-configured to work with ABCD as 'localhost' (which means : the PC itself runs both the client and the server). A small script launched the httpd (or Apache) service based on that configuration, so that installation and configuration efforts in principle could be kept to a strict minimum. Since nowadays too many different versions (32/64 bits, thread-safe or non-safe, MS VC10,12,14...) exist we think it is better to leave the installation of Apache and PHP to specialized packages such as the above mentioned WAMP and XAMP or, in the case of Linux, the 'server-versions' of the Linux-distributions which come

with their own Apache and PHP pre-installed. In case of additional configuration still to be necessary, the user should be fully aware of the fact that Apache, as a Linux-based software, is case-sensitive for its parameters and file names (with path information) !
In the case of webservers, we should mention 'IIS' (Internet Information Services) of Microsoft, free software but not open, which is the webserver coming with Windows. Differences are mostly in the way how it should/can be configured and managed, rather than performance, security etc... The terminology is a bit different (e.g. 'aliases' are called 'virtual folders' and there is no easily approachable ASCII-configuration file as with Apache and its 'httpd.conf'.
ABCD runs perfectly with IIS, as with other web-server software (e.g. Xitami), but this manual does not support the implementation on IIS. Dedicated manuals on configuring IIS for ABCD exist.

3. ABCD installation

3.1. Available installation versions

ABCD version 1.0 came originally in three main versions :

1. non-assisted installation package for Windows : this is a ZIP-file containing all necessary files, which simply need to be unzipped into the root of (one of your) harddisk(s), e.g. C:\. Since it encompasses both Apache and PHP, after simply unzipping it should normally work ! Here Apache comes with its own configuration file (httpd.conf) where port 9090 is activated to allow running next to possible other running Apache installations. Only for some specific uses, e.g. the use of Z39.50 which needs additional PHP-modules (YAZ), it will be necessary to do some editing (of e.g. php.ini).

   ABCD installations which will use the Advanced Loans module (EmpWeb) need to additionally unpack the most recent EmpWeb....zip package, where an extra main subfolder in \ABCD will be created and some additional files will be added to the ABCD Central directory.

2. assisted installation for Windows : this is a self-installing executable, which will first check whether Apache and PHP are already installed on the system. If so these installations will be skipped, if not they will be added to the basic ABCD installation. This requires mainly following the dialogs and instructions of the installer itself. At the end a similar directory-folder as with the package under 1. will be the result.

3. non-assisted installation package for Linux : this is a .tar.gz archive for Linux systems which should be unpacked into the Linux file-system, depending on its organisation (definition on where such applications can be put). If the correct access-rights are granted (with the appropriate Linux-commands such as chown and chmod) ABCD can be installed, like in Windows, under the file-system root '/'. In Linux systems the assumption is that Apache and PHP are installed separately (e.g. with the dedicated tool like apt-get or Synaptic), so the package only contains the proper ABCD files in the 'www'-directory.

The new version 2.0 distribution is simplified : no longer ABCD comes with its own conifguration of Apache with PHP for Windows, since nowadays very good installation packages for these environments exist (WAMP, XAMP) but also these packages arein a better position to keep the different versions of Apache (ASF, Bauhaus ??) in thread-safe or not compiled with one of the many MS Visual C versions (9, 10...14) streamlined with the correct verisons of PHP, again 32- or 64-bit versions etc.

So the installation for Windows is now very similar to the one for Linux : Apache and PHP are supposed to have been pre-installed and -configured. ABCD only needs a 'virtual host' configuration file to be added into the Apache-server in order to run from that pre-installed version. In e.g. WAMP the file - coming with the installation - httpd-hosts-abcd.conf simply needs to be added into the 'alias'-directory and Apache restarted.

3.2. Installation issues

This section deals with the installation issues for ABCD. Since ABCD has several totally different components, installation by definition encompasses quite some potential pitfalls. Three main reasons can be given for the installation to be complex :

1. ABCD is a combination of several software technologies : ISIS-databases, ISIS-scripts and ISIS-formats, a webserver, PHP-scripting, plus (in the case of the advanced Loans module) some JAVA and MySQL parts;

2. being web-based, which means a web-server has to be installed and special measures have to be taken about access rights and security : in principle the whole world - with access to the WWW - can interfere.

3. ABCD will be installed in quite different situations, varying from a simple stand-alone (even non-networked) PC upto servers in big networks with a webserver and often also PHP-scripting services already pre-installed.

Previously the installation packages came in two types :

1. a full package, containing all ABCD-proper files plus the Apache webserver and PHP-scripting engine.

   In this situation an archive (.zip) needs to be unpacked into a root-folder of the file system (which can be any operating system in which Apache/PHP and ISIS can run). After unpacking there will be a dedicated folder for Apache, another one for PHP, a cgi-folder (to contain the web-accessible executables) and a 'documents' folder (in Apache called ' htdocs') which acts as the homepage of the ABCD-application.

   - Apache comes with a pre-defined configuration file (httpd.conf in the conf subfolder of the Apache folder) which defines the following specific parameters :

Apache parameter	explanation
ServerRoot "/ABCD/apache"	the directory from where Apache runs
Listen 9090	the port used by ABCD, the default http-port being 80, but in order to avoid interference with other existing http-applications, if so desired, a different port can be used, e.g. 9090. In case of using a different port-number, some adjustments will have to be made in the ABCD_start.bat script and in some OPAC-URL's.
PHPIniDir "/ABCD/php"	The folder from where PHP is running
DocumentRoot "/ABCD/www/htdocs"	The root-folder for all the files which are part of the application itself, so the ' homepage'
ScriptAlias /cgi-bin/ "/ABCD/www/cgi-bin/"	The folder in which Apache will executables allow to run from instructions in the web-pages

Note

Make sure the 'cgi' module of PHP is installed into Apache, which is no longer the case (as before) in newer Apache installations. The command to install this module in Linux is :

```
sudo a2enmod cgi
```

- PHP comes with a predefined configuration in php.ini.

PHP settings and php.ini

Since ABCD uses PHP throughout with some additional PHP modules (YAZ, XSLTProcessor...) Pears should be installed within the PHP-installation and some extra modules need to be copied into the PHP 'extensions' folder : php_yaz.dll, yaz.dll, yaz3.dll (these two serve the Z39.50 function of ABCD cataloging), iconv.dll, libxm2l.dll, libxslt.dll (for the XSLT Processor). The PHP-extensions folder needs to be present in the system's path environment variable (in Windows e.g. : go to 'My Computer (right-click) | Properties | Advanced | Environment Variables | System variables and edit the Path variable by adding, if not present : ';C:\ABCD\php\ext'). Also make sure your php.ini (in \ABCD\php) has the extensions mentioned here commented out (i.e. remove the leading ';' to activate the extension).

extension=iconv.dll

extension=iconv.dll

extension=libxml2.dll

extension=libxslt.dll

extension=yaz3.dll

extension=php_yaz.dll

Be careful with possible other php.ini files existing, e.g. in \Windows or \PHP as these might disturb your ABCD-PHP. A PHP-test option is available with ABCD at the URL : http://localthos:9090/info.php. We are specifically interested in the following section below, where XSL and YAZ should be mentioned as running - if not check your path-environment variable and all paths again, as well as the 'extensions' section of your php.ini !

xsl

XSL	enabled
libxslt Version	1.1.23
libxslt compiled against libxml Version	2.6.32
EXSLT	enabled
libexslt Version	0.8.13

yaz

YAZ Support	enabled
PHP/YAZ Version	1.0.11
YAZ Version	3.0.24
Compiled with YAZ version	3.0.6

The php.ini file contains a few more settings which need to be checked for ABCD to run correctly :

- register_globals = On (default = Off)
- extension_dir = "/ABCD/php/ext" (or adjust to the real path to your ABCD installation)
- default_charset = "iso-8859-1" (default = not active) or "utf8" if Unicode is to be used
- extension_dir = "/ABCD/php/ext" => defines the extensions directory
- extension=yaz3.dll and extension=php_yaz.dll are listed in the => are added in the 'Dynamic Extensions' section in order to allow the YAZ-module for Z39,50 to work

Note

As from ABCD 2.0 it is no longer required to keep the setting for 'short_open_tag' to 'On' (which is contra-indicated anyway). All 'short open tags' '<?' have been changed to '<?php' .

2. an ABCD-only package, requiring Apache (or another web-server) and PHP already being installed.

In this case the assumption is that at least some expertise is available to understand the existing web-server installation and PHP configuration. Using ' aliases' for the ABCD-installation and cgi-folder, which can be put in a virtual host configuration file, ABCD can be installed anywhere inside or outside the existing home-folder for the web-server. So only the cgi-folder and htdocs-folder is included into this package. System managers

should refer to the Apache and PHP manuals in case they are not sure about how to proceed with this type of installation.

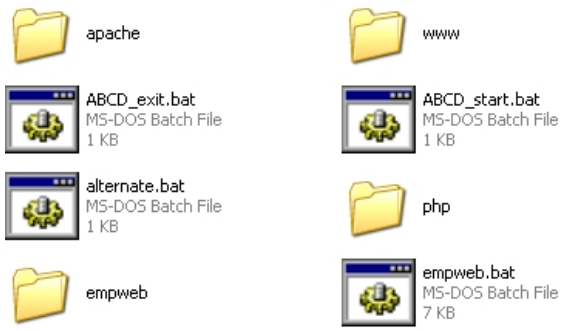
Alternatively one could also use prepackaged installations like EasyPHP or WAMP (for Windows) / XAMP (for UNIX/Linux). Again in this case Apache and PHP (and MySQL) will be automatically installed and the ABCD cgi-bin and htdocs folders have to be moved into the existing folder-structures (of Apache) and php.ini has to be edited.

As from ABCD 2.0 only the second type of distributions will be available, leaving the installation of Apache and PHP to other, more specialised packages such as WAMP or XAMP. Only a specific 'virtual host' configuration file for ABCD needs to be added to the Apache configuration (e.g. in WAMP : by putting it into the 'alias' directory) and some PHP settings need to be checked in php.ini to activate additional extensions e.g. for gd2, libxml, xsl, and if the related functions are used : ldap, yaz, mysqli (for Empweb) and mbstring (for Unicode).

A dedicated installation tool will be created as part of the ABCD-software, but in essence still doing the same as described above, only after collecting some parameters for installation (like which disk to use, which port etc.).

3.3. Directory structure and access rights

After installation of ABCD the following folder structure will be created (in this case EmpWeb is included) :



As can be seen, 3 (or 4 if EmpWeb is included) sub-folders have been created in the main folder /ABCD. In the case of installation of the optional Advanced Loans folder one more folder containing basic technology for ABCD will be added : the Java Development Kit (JDK). The standard folders are resp. :

1. apache [ABCD 1.x only]

   The Apache folder contains the Apache web-server software, which is in fact only of several important softwares developed by the Apache Software Foundation. By default Apache webserver is installed in another base-folder (e.g. in Windows : C:\Program Files\Apache Software Foundation\Apache2.2) and network-managers will probably have installed Apache on their server(s) according to their own preferences, but when installed from the 'full ABCD-package' Apache will run - with its configuration file httpd.conf adjusted for this situation - from \ABCD\Apache.

2. php [ABCD 1.x only]

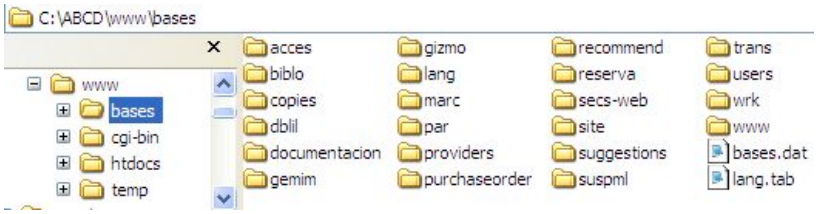
   THe PHP folder contains the PHP scripting software. Again, as with Apache, in many instances this software will be installed in its own right, e.g. in C:\PHP, or often also as part of a combined package containing Apache, MySQL and PHP, e.g. with EasyPHP or WAMP-server. When installed as part of ABCD however PHP will run from here with the necessary adjustments done in the main PHP-configuration file php.ini.

3. www

   The www folder contains the whole ABCD system, which is subdivided in 4 folders :

a. bases

The bases folder contains the databases of your ABCD installation, which one dedicated subfolder (with many subfolders in its turn) for each database. When an additional database is copied or created using ABCD, the system will create such a dedicated extra subfolder here. A typical list of database-folders in the /bases folder looks as follows :



[!!] As can be seen, many databases exist (but not as many as there are tables in a relational setup, since ISIS does not practice 'normalisation' into related tables), some of them - e.g. marc, biblo, dblil - are models coming with the installation of ABCD, others - in this case e.g. 'gemim' - are created by ABCD in the author's installation only, while finally others are serving specific modules of the library system, e.g. 'providers' and 'purchaseorder', are used for the acquisitions module, 'suggestions', 'suspml', trans and users are used for the Loans Module. 'recommend' and 'reserva' meanwhile are legacy folders to older ABCD-versions pre 1.0. So each ABCD-bases folder will be different according to the actual databases used.

Some essential configuration files to be located in this folder are :

- bases.dat : the list of databases available to this Central-installation (i.e. a database-name, a column-separator '|' and a description
- lang.tab : a list of languages used as key-value pairs for code-full language, e.g. en=english
- abcd.def : the main system-wide configuration file, see the configuration section
- loans.dat : if this file exists, ABCD Central will use the copies/loanobjects directly from the catalog-database, not from the dedicated copies/loanobjects databases;
- acquisitions.dat : a file listing the catalog databases to be used for acquisition of copies, e.g.

  marc|Marc

  biblo|Cepal

Note that since more than one bases-directory can be defined (in db_path.dat), also different files abcd.def (with e.g. different LEGEND1 and LEGEND2 parameters to identify the base-directory on the screen in the footer) can be used.

A special database is the database 'acces' which holds the users (with their login data) and their access-rights (authority level) to the databases.

In the 'www'-folder ABCD keeps some small special files, e.g. the 'prolog' and 'epilog' html-codes which will be invoked resp. before and after the main page contents of each ABCD-page produced by an ISIS-PFT. This is where system manager could - if so desired - add code (e.g. JavaScript) they want to be executed at each page.

The 'LANG'-folder is also quite special : it contains, for each language used, the tables with messages used for each of the Central modules. E.g. the 'lang.tab' file contains the information on the actually officially supported 4 languages : pt=Portuguese fr=French en=English es=Spanish. This list of languages for each language is based on the basic language-authority file lang.tab in the bases-folder itself (where it resides along with the authority list of databases available : bases.dat).

Note

[!!] The '00' lang subfolder contains the tables serving as 'master' for the other languages. Whenever a message is not found by ABCD in the language selected, it will refer to these tables and use the messages contained there, to avoid missing messages in any language. This way one can also start translating into a new language without having to finish the full job before using ABCD, as the missing messages will be taken from the language '00'.

Finally also the folder 'par' is, like 'lang', not a database folder but it holds the .par files for each database known to ABCD. A .par file actually is a small text-file (so it can be edited by any TXT-editor like Notepad) with on each line the full path reference to parts of the database concerned. E.g. a typical .par file for ABCD looks like this :

“marc.*=%path_database%marc/data/marc.* prologoact.pft=%path_database%www/prologoact.pft prologo.pft=%path_database%www/prologo.pft epilogoact.pft=%path_database%www/epilogoact.pft epilogo.pft=%path_database%www/epilogo.pft autoridades.pft=%path_database%marc/pfts/en/autoridades.pft”

Each element gets, after the equation sign, its path in the file-system. As can be seen, variables taken from the Operating System's Environment can be used, in this case %path_database%, which is substituted by the real pathname as defined in the main configuration file config.php (see infra).

[!!] While normally all elements referred to here belong to the database in question, elements of other databases should also be added if they are used in 'REF'-statements of the formats used in this database, since ISIS will have to know where to locate such external database element if called from a format - and will look for its path here !

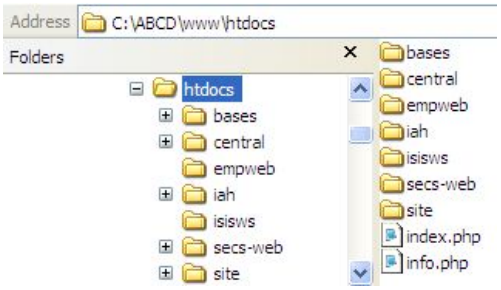
b. cgi-bin

The cgi-bin folder contains the executables which ABCD will call from its web-pages and which therefore should be authorized to run by the webserver (Apache) using the CGI-protocol. In the case of ABCD the main executable is the wxis.exe ISIS-server, which does the main part of the job. Some other CISIS-tools are however also included for specific tasks.

The wxis-modules subfolder here contains scripts (with .xis extension) for the wxis-server, while the 'gizmo' folder contains some small ISIS-databases which define strings to be substituted by another one, e.g. for changes due to different environments used (DOS/ASCII, Windows/ANSI, WWW/XML.

c. htdocs

The htdocs (we use the traditional Apache 'hypertext documents' folder name) is the 'home-folder' of the web-site served by the ABCD-Apache server. So therefore it contains all the software elements (except the basic external technology such as Apache and PHP) specifically produced for ABCD :



Two initial scripts are present within this homepage folder : index.php (which is the default home-page indeed, allowing the URL of ABCD only to refer to the server-part) and the [!!] 'what.php' script for including the footer info.

An optional file 'db_path.dat' can be located here to point to different (each on one line) database-folders.

Since ABCD is a 'suite' of different functions, each one has its own homepage, i.e. the 'index.html' file located in the appropriate subfolder.

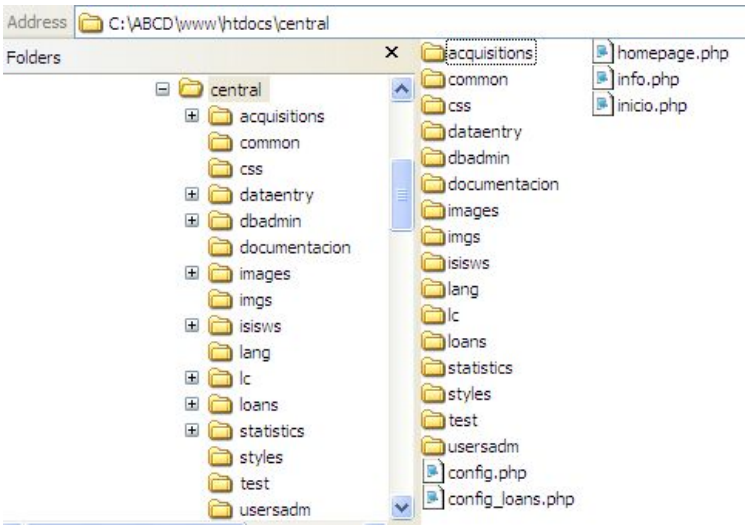
The main folders of the ABCD-system are briefly described below here :

i. bases

Here for each database (in a dedicated subfolder) external files linked to from the records in the database, e.g. full-text PDF's or images, will be stored E.g. the user images can be stored here in a subfolder 'users', so the photos of the user will be shown whenever a loans-system user is presented. [!!] Don't mix this folder up with the 'bases' folder where the actual databases reside !

ii. central

This is indeed, as suggested by the name, the 'central' part of the system where most of the database administration and many core-activities of the software are included. We will therefore deal with the important subfolders contained in here :



Some initial scripts are located at this level : homepage.php and inicio.php are the starting pages, which read into memory the main configuration parameters defined in config.php (or config.loans.php for the Loans module). For the 'mySite' functionality, additional initial scripts will be found here too : iniciomysite.php, homepagemysite.php and availibility.php. These scripts now (as from ABCD2.0) contain both the code for EmpWeb (using SQL-queries) and Central Loans (using ISIS-QL), so e.g. 'empwebavailability.php' is no longer used since the code is included in 'availability.php'.

The basic configuration files **CONFIG.PHP, SYSTEM_CONF.PHP and DATABASE_CONF.PHP,** which are also found here, will be discussed in more detail in their dedicated sections on the ABCD-configuration.

The following folders here deal with one specific function or module of ABCD by storing the PHP-scripts with lots of additional elements (images and style-sheets for the webpages etc.) : acquisitions, dataentry, dbadmin, loans, statistics and usersadm.

The names of the folders are sufficiently self-explanatory in these cases. Here we would only like to underline the presence of a module 'database administration' which allows creation of any ISIS-structure

to deal with any type of textual data, allowing ABCD to be more flexible than most other systems and more than just a library system.

Special folders here dedicated to special functions in ABCD here are the following :

- common : in here there are some crucial php-scripts which are needed by all modules, e.g. 'header' and 'footer', but also 'wxis-llamar.php' (which allows using either the cgi-method of calling executables (safer) or direct executable calls from PHP (faster). The instituational_info.php script defines the name of the resonsible institution of the ABCD-installation, which will be called upon in many pages.
- documentacion : obviously this folder contains scripts to deal with the online-help functions of ABCD.
- images : contains small images used in many pages (mostly .png and .gif)
- css : contains the Cascading Style Sheets used in this central part of ABCD
- styles : contains the basic main stylesheet 'basic.css'
- lang : contains for each module a script to facilitate language switching or reverting to the default language
- [deprecated] test : contains some scripts testing the ABCD-installation and access to the cgi-executable.

iii. empweb

This folder is non-standard and contains the subfolders and software parts used by the Advanced Loans module of ABCD, which is not further detailed in this manual.

iv. iah

iAH is the original name of the **a**dvanced web-**i**nterface for '**H**ealth Information' of BIREME which acts as the OPAC of ABCD but also as a meta-search engine on other defined-as-relevant sources.

v. isisws

This folder contains scripts for the SOAP-related functions of ABCD.

vi. secs-web

This module allows ABCD to offer advanced serials management tools within the web-environment : **Se**rials **C**ontrol **S**ystem.

vii. site

Finally the 'Site' module combines advanced OPAC searching (with meta-search possibilities) with a 'portal' service, offering the search option within an environment of other networked information resources and communication with users. The structure and the contents of this portal can be edited online with a built-in ABCD Content Management System.

d. EmpWeb (only if installation of EmpWeb was added !)

This folder contains most but not all files necessary to run EmpWeb, e.g. the Java Jetty server and the scripts. EmpWeb however additionally needs also added scripts in ABCD Central (this allows the Advanced Loans to be compatible with the built-in Loans system of ABCD) and - since it uses an SQL-database for storing the transactions - an installation of one of the common SQL-databases (MySQL, PostGres, Oracle...), which needs to be done separately - use the installation instructions for the SQL-solution chosen. A separate manual on EmpWeb is available.

Chapter 2. ABCD Modules

1. Introduction and general configuration

This chapter deals with the main functions [!!] of the 'Central' module of the ABCD system. As an integrated 'library automation software' the system offers tools for database management (both for bibliographic/document databases and administrative databases such as users, acquisitions and loans), data-entry, statistics, circulation, serials control and searching functions (OPAC in a 'portal' environment).

These functions are presented in different parts of a suite, which are relatively independent from each other but not fully. Parts are accessed by their own URL. Within one part several modules can exist which also cooperate. For example the pre-cataloging information produced for acquisitions will be re-used in the copies-database for the inventory and loan-objects database for the circulation module, which in its turn uses bibliographic information from the catalogs. Statistics can be applied to any ISIS-database, not only the Circulation databases, so this function will also re-appear at several instances within the software. The OPAC-technology can run on any ISIS-database, not only the own ABCD-catalogs, so it will be described as a relatively independent tool, as will be the case with the Serials Control.

Important

How to access the suite parts directly ?

- The first six modules together constitute the 'Central' part of the ABCD-suite. It can be accessed by the [!!] URL http://[serverURL]:9090. The 'index.php' part is optional if the web-server (Apache) has been told that index.php is one of the default pages in the folder (as is e.g. also 'index.html').
- The combined OPAC-with-portal (Site) can be accessed by http://[serverURL]/site/index.php, with the administrator page for this Site being http://[serverURL]/site/admin/index.php.
- The Serials Control part should be accessed by the URL http://[serverURL]/secs-web/index.php.
- EmpWeb can be accessed, if installed, by the URL http://[serverURL]/empweb/ (note the trailing forward-slash here !)

For all these parts or modules ABCD provides publicly available start login data, which need to be read from the 'leiame.txt' or 'readme.txt' files coming with the installation package. It is the responsibility of the system manager to take these login data out of the system and replace them by local - and locally controlled - login data.

ABCD manages the control of who can access the system and with which privileges through a system (introduced with version 1.0) of 'profiles'. Profiles are sets of allowed modules, databases and forms. ABCD users (not library patrons) then will be assigned to one of the defined profiles (see 'Users Administration' in the following section).

1.1. Multilinguality configuration

ABCD is fully conceived as a multi-lingual software, allowing the creation and use of any language. All language-sensitive elements, such as screen messages and help-files, but for databases e.g. also display-formats, field-definition and -selection tables etc. are stored in separate subfolders for each language. These subfolders are named according to language-codes which are defined in the file 'lang.tab' of the 'bases'-directory. An example of such lang.tab file, in which two additional (non-Latin alphabet) languages have been added to the four original default languages, is listed here :

pt=portuguese

es=spanish

en=english

fr=french

am=amharic

si=sinhalese

With this list all messages (the 'lang'-database) and database-structures and -printformats will (need to) have sub-folders pt, es, en, fr, am and si. The list of languages has to be repeated in each lang-subfolder, if so desired translating each of the language values (not 'keys' as the abbreviation codes need to remain constant !) can be translated into the underlying language and - if 'Unicode' is switched on (see infra) - even written in non-Latin alphabets. Beware of non-ASCII characters such as 'ñ' (e.g. in español) since these are coded differently in ANSI (Windows) vs. Unicode.

Since ABCD can deal with multiple database-folders (see infra), it will be necessary to define the path to the 'lang' database for the messages with the variable '$msg_path'. This defaults to the directory defined by $db_path in config.php but can also be defined independently to any folder in the system.

A special 'virtual' language with code '00' is used in ABCD to hold all messages from which the language translations will be derived. These key-value pairs are mostly defined in English. The keys are the ones referred to in the PHP-scripts whenever displaying a language-dependent text on the screens, but will be searched for in the related table in the active language, unless it cannot be found there : then the '00' language acts as the reserve fall-back option. This means that if a message appears not in your own language but in English (supposing your language is not English), then that key still needs to be translated (added) to the corresponding table for your language and the message displayed is taken from the '00'-language.

Later in this manual a dedicated section will discuss how to create a mew language in ABCD with a Unicode non-Latin example.

1.2. The main configuration files for ABCD Central

The main configuration of ABCD Central is based on the following files :

1. config.php
2. system_conf.php and abcd.def
3. database_conf.php and dr_path.def

1.2.1. CONFIG.PHP

This file contains the local variables which are managed by the system administrator.

Currently the list of variables is as follows :

- Date_default_timezone_set (the allowed values can be found at the URL http://php.net/manual/es/timezones.php
- $ Open_new_window : if set to 'Y' after logging in a new window with the main ABCD-interface will be opened in addition to the login screen
- $ Context_menu : if set to 'N' the Central interface in the new window will not contain navigation elements (e.g. by 'right-clicking') - this is safer for keeping the ABCD work apart from other browser-activities and force the operator to use the interface elements rather than browser-elements e.g. to 'go back'
- $config_date_format : defines the format of the dates used; to be given as DD for days, MM for months and YY for the year-display, each time separated by slashes '/'.

Note

If a field is defined as 'ISO'-date, the format will automatically be yyyymmdd; this can be automatically derived from a preceding normal date-field defined with $config_date_format

- $Inventory_numeric : if set to 'Y'es leading zero's (at the left) will be omitted when reading the inventory number (or barcode) to make it a real numerical value
- $max_inventory_length : a number defining the fixed number of positions in the barcode or inventory numbers; missing positions will become leading zero's. This parameter is taken into account in the processes related to the assignment of the inventory numbers of the database copies
- $max_cn_length : same as $max_inventory_length but for the Control Number field : missing positions up to the number defined here will be filled with zero's
- $app_path : the name of the folder acting as 'Central' with all Central scripts; best kept as 'central'; if renamed some scripts might fail.
- $log : if set to 'Y'es 'and a subfolder 'log' exists in the database-directory, all calls to the Isis-executable 'wxis' will be logged in one file per day named according to the date and with the following information :

  First line : ** Friday 30th of November 2012 07:34:35 AM Operator: abcd Identifies the date, time, and operator executing the transaction Second line: /central/dataentry/inicio_main.php/ABCD/www/htdocs/central/dataentry/wxis/login.xis IsisScript = / ABCD / www / htdocs / central / dataentry / wxis / login.xis ? Base = access & cipar = c: /bases_abcd/bases/par/acces.par & Login = abcd & password = adm & path_db = c: / bases_abcd / bases / & cttype = s

  where the following elements are identified: name of the php script that is running, name of the wxis script being invoked, parameters sent to IsisScript for the corresponding execution

  ## Note

  In the future this log will also be made database-dependant.

- $img_path : derault path where the images and digital documents linked to the records of the databases will be stored. This default directory can be modified using the dr_path.def file that provides more specific information about the storage of digital documents for a specific database (see infra).
- $msg_path : path to the lang-database for the messages of the interface; this path can be in one of the defined database-folders but also outside any of them
- $lang : default language used when opening Central
- $lang_db : default language for the database administration module (can be different from $lang)
- $change_password : whether 'Y'es or 'N'ot operators are allowed to change their passwords (in the login-screen)
- $adm_login : an optionally built-in login-name to (temporarily) allow entering the Central system even if the access to the databases is not possible, e.g. for debugging/testing reasons; for security reasons : remove this login when not needed.
- $adm_password : an optionally built-in password for the login with $adm_login;

  ## Note

  remember this login/password acts on the full-administrator level and therefore is at a high-risk level !

- $dirtree=1: show (1) or hide (0) the icon or menu-entry that gives access to the exploration of the bases-folder; some network- or server-managers will not allow such function on their system !
- $MD5= When set to 1 (on) passwords will be encrypted with the MD5 encryption
- $fix_file_name = defines an array with substitutions of characters to be applied in file-names for uploaded files (in order to avoid complicated e.g. diacritical characters; the default values are :

array('Š'=>'S', 'š'=>'s', 'Ž'=>'Z', 'ž'=>'z', 'À'=>'A', 'Á'=>'A', 'Â'=>'A', 'Ã'=>'A', 'Ä'=>'A', 'Å'=>'A', 'Æ'=>'A', 'Ç'=>'C', 'È'=>'E', 'É'=>'E', 'Ê'=>'E', 'Ë'=>'E', 'Ì'=>'I', 'Í'=>'I', 'Î'=>'I', 'Ï'=>'I', 'Ñ'=>'N', 'Ò'=>'O', 'Ó'=>'O', 'Ô'=>'O', 'Õ'=>'O', 'Ö'=>'O', 'Ø'=>'O', 'Ù'=>'U', 'Ú'=>'U', 'Û'=>'U', 'Ü'=>'U', 'Ý'=>'Y', 'Þ'=>'B', 'ß'=>'Ss', 'à'=>'a', 'á'=>'a', 'â'=>'a', 'ã'=>'a', 'ä'=>'a', 'å'=>'a', 'æ'=>'a', 'ç'=>'c', 'è'=>'e', 'é'=>'e', 'ê'=>'e', 'ë'=>'e', 'ì'=>'i', 'í'=>'i', 'î'=>'i', 'ï'=>'i', 'ð'=>'o', 'ñ'=>'n', 'ò'=>'o', 'ó'=>'o', 'ô'=>'o', 'õ'=>'o', 'ö'=>'o', 'ø'=>'o', 'ù'=>'u', 'ú'=>'u', 'û'=>'u', 'ý'=>'y', 'þ'=>'b', 'ÿ'=>'y',' '=>'_');

Note

the last element substitures a space for an underline in file-names

- $show_acces= When set to Y(es) the 'acces'-database with operators and their passwords (yes or not encrypted depending on $MD5) will be shown in the list of available database (if included there).
- $EmpWeb= When set to 1 (on) the loans-system will use the EmpWeb Advanced Loans module and mechanisms e.g. in 'MySite' or availability-checks.
- a new section can be optionally added to CONFIG.PHP dealing with the variables needed for the LDAP-authentification function, with example data given :

```
$use_ldap=false;

$ldap_host = "ldap://zflexldap.com";

$ldap_dn = "cn=ro_admin,ou=sysadmins,dc=zflexsoftware,dc=com";

$ldap_search_context = "ou=guests,dc=zflexsoftware,dc=com";

$ldap_port = "389";

$ldap_pass = "zflexpass";
```

- finally also an optional section can be added to assist the configuration of EmpWeb (if used), more specifically to pre-define some variables and re-define some ports :

```
ProxyPass /empweb/ http://127.0.0.1:8080/empweb/ // to direct calls to empweb to port 8080 instead of 9090

ProxyPassReverse / http://127.0.0.1:8080/ // reverse of previous

$empwebservicequerylocation = "http://localhost:8086/ewengine/services/EmpwebQueryService";

$empwebservicetranslocation = "http://localhost:8086/ewengine/services/EmpwebTransactionService";

$empwebserviceobjectsdb = "objetos";

$empwebserviceusersdb = "*";
```

Note

DEPRECATED VARIABLES : $institution_name and $institution_URL are no longer used but replaced by variables in abcd.def (see next section).

1.2.2. system-variables defined in abcd.def

In the main bases-folder (in Windows e.g. C:\ABCD\www\bases, in Linux : /var/opt/ABCD/bases) a file 'abcd.def' defines system-wide variables. The main current variables here are :

- LEGEND1= followed by the text to display at the second line of the Central footer

  > **Note**
  >
  > The 1st line of the footer is defined in the code of the script 'htdocs/central/common/footer.php. In this script also the following variables are arranged, so the administrator could envisage re-arranging them if so wanted by changing the (simple) script.

- LEGEND2= follwed by the text to display at the header of Central (next to the ABCD logo)
- URL1= followed by the text to display as the third footer line as a link
- URL2= followed by the text to display as the fourth footer line as a link
- FRAME_1H= a value (typically 100) defining the number of pixels for the height of the upper segment (mostly dark-blue background with ABCD logo) in the Central data-entry screens
- FRAME_2H= a avlue (typically less than 100) defininng the number of pixels for the height of the toolbar (with search-box) segment of the Central data-entry screens.

  > **Note**
  >
  > With the variables FRAME_1H and FRAME_2H the administrator can adjust the interface to different resolution screens, however only at a system-wide level. Lower-resolution screens will require lower values.

- MULTIPLE_DB_FORMATS=
- UNICODE=
- DIRTREE_EXT=
- LOGO=[filename with logo picture] : reference to the picture file to be used as logo in the upper part of the Central screen. The default logoabcd.jpg 

can be replaced by any localized image here. Make sure the image is small enough (e.g. max. 3 Kb while the default is only 1,2Kb). Tools for re-sizing pictures are freely available for both Windows and Linux.

  > **Note**
  >
  > The logo for ABCD iAH and Site need to be defined differently. E.g. for the OPAC (iAH) the image is defined in the file htdocs/iah/iah.def.php with the variable 'LOGO IMAGE=' ,

1.2.3. database-variables defined in dr_path.def

In this file 'database-related paths definition', to be located in the base-folder for each database, some variables can be defined which re-define characteristics of the database. These parameters are ALL optional ! Currently the list of variables at this level is :

- ROOT= defines the default path for this database where files attached to the record, e.g. images, PDFs etc. will be stored, e.g. ROOT=/var/opt/ABCD/bases/marc/dr/
- IMPORTPDF= defines with Y or N whether documents (PDFs) can be imported into this database. if 'Y'es the PDF-icon will be shown in the Central toolbar
- barcode= defines with 'Y' or 'N' whether or not barcodes are used in this database; this parameter defines whether the barcode icon will be shown in the Central toolbar

- tesaurus= defines the name of the thesaurus-database to be used with this database, e.g. tesaurus=agrovoc

  Note

  the Spanish spelling of 'tesaurus' in this case

- prefix_search_tesaurus= defines the prefix with which the descriptors in the database are indexed, e.g. MA_
- COLLECTION= defines the path were digital library collections will be stored for this database. Such directory could be defined under the database-folder itself (e.g. /var/opt/ABCD/bases/dubcore/collection/) or as a collective directory for all databases in the system (e.g. /var/opt/ABCD/bases/collections).
- UNICODE= defines with '1' or '0' whether the database uses utf8 or unicode encoded content. In fact any value greater than 0 can be used, e.g. 'Y' will simply indicate 'yes, use unicode', '1' but also '2' etc. will also indicate 'yes use unicode' but at the same time identify Unicode character-subsets to be used, e.g. '1' includes Amharic, '2' includes Sinhalese, '3' includes Arabic etc. These codes are relevant for the OPAC where the alphabets will be shown in clickable icons.

1.2.4. Defining different database-directories : db_path.dat

The file 'db_path.dat' in the bases-folder allows to define several directories (or folders) to be used as bases-folder, not only the current one (defined in CONFIG.PHP). The lines in this optional file contain the path, followed by a column-separator '|' and a description as shown in the selection-menu.

E.g. to define 2 database-folders, one with operational and one with test-databases, the file db_path.dat would be (example for Windows) :

```
C:/ABCD/www/bases/|Operational

D:/databases/|Test-databases
```

These two options will appear in a list in the login-screen of ABCD Central and the $db_path variable will be substituted by the selected one.

1.3. Login configuration of ABCD Central

The most important configuration file for ABCD Central is the file 'config.php' in the /www/htdocs/central folder. We discussed some general parameters earlier on (in the section about installation) as they deal with installation paths and default language to be used, but at the end some parameters are introduced which provide a recovery solution for the case when the general System Administrator login data have been lost (meaning the system cannot be entered by anyone anymore !).

These parameters are :

```
//USE THIS LOGIN AND PASSWORD IN CASE OF CORRUPTION OF THE OPERATORS
DATABASE OR IF YOU DELETED, BY ERROR, THE SYSTEM ADMINISTRATOR

$adm_login=""; $adm_password="";

//USE THIS PARAMETER TO ENABLE/DISABLE THE MD5 PASSWORD ENCRYPTION
(0=OFF 1=ON)

$MD5=1;
```

By manually editing (e.g. with Notepad or another flat-ext-ASCII editor) the $adm_login and $adm_password parameters one can create temporary login data, which will allow to create real logins again (using the database administration tools for the USERS-database). It is strongly advised to immediately thereafter again remove the login-data from this file config.php - for obvious security reasons.

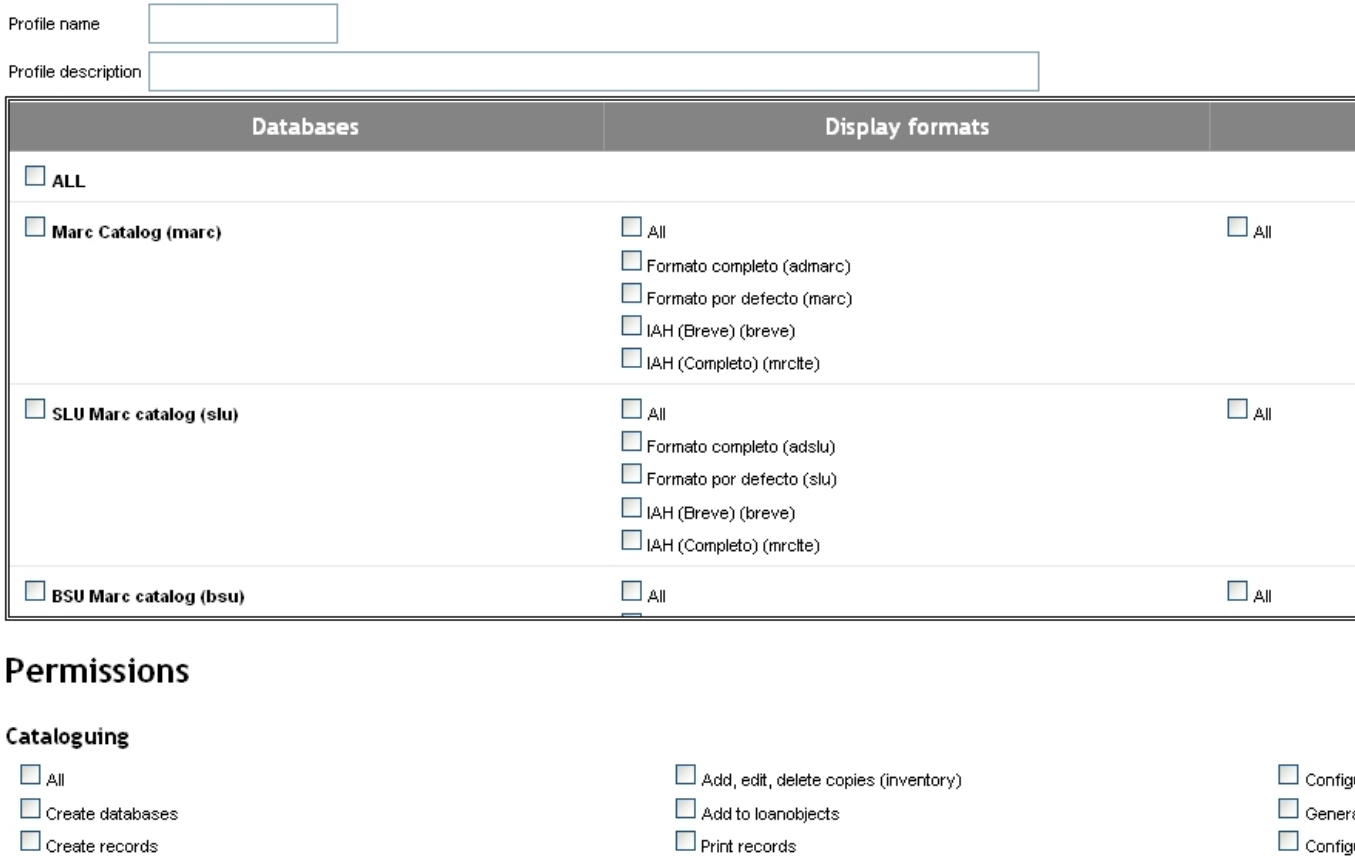
The $MD5 parameter will invoke password encryption (using the MD5 algorithm) when put to '1' or not if put at '0'.

1.4. Administration of the ABCD user profiles.

From the main Central menu option 'Users Administration' one can enter the 'Create/Edit profiles' option in addition to create/edit/delete system users.

Some profiles come with the ABCD installation as examples, e.g. :

- System Administrator
- Database Administrator
- Database Operator
- Operator LIS



As can be seen from this screen, the administrator has to enter the following data :

1. profile name : any (short) name for the new profile to be created
2. profile description : any description describing the profile
3. for each database listed : whether or not access is granted and which data-entry forms of that database can be used by this profile. If all, 'All' can be checked for simplicity.
4. Permissions : for each module (Cataloging - Circulation - Acquisitions) the menu-option (or function) to which this profile should have access should be activated (in the small box).

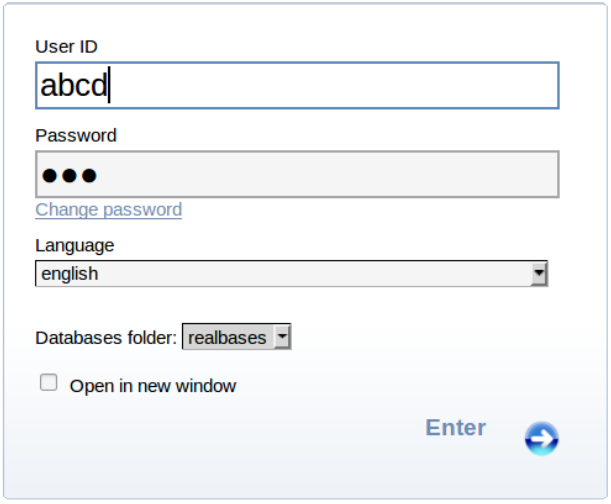
Note

The list of profiles is kept by the Operating System in the file 'profiles.lst' in the bases/par/profiles and for each profiles the characteristics (databases allowed, modules allowed are kept in a file named after the profile (without extension). Privileges not granted in the profile are assumed to be

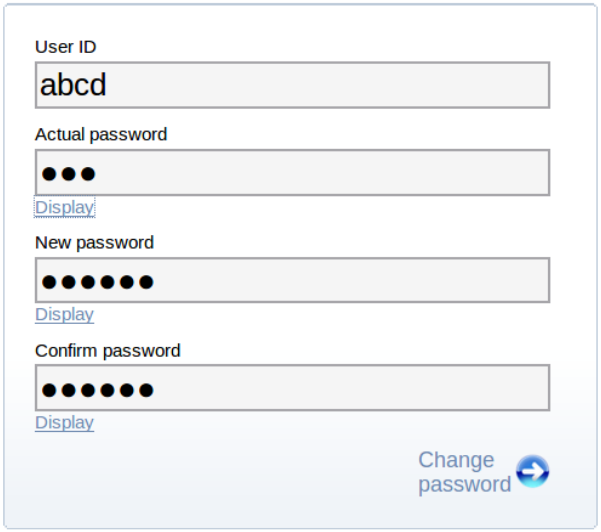
'not' allowed. The privileges have brief but mostly self-explanatory names, e.g. 'delrec' for deleting records, expimp for exporting and importing records etc.

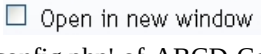
1.5. Logging in into the system

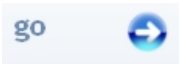
After profiles have been created (or adopted from the default ones) and have been assigned to system users, any of the system users' logins can be used to enter the system. Unlike in previous preliminary versions of ABCD, it is obviously no longer necessary to select an authorization level when logging in, since this is now coupled with the login itself. Therefore in addition to login-name and -password only the language to be used is to be selected (this list is taken from the file 'lang.tab' in the bases-folder of ABCD (default : /ABCD/www/bases).



As from ABCD 1.5 and 2.0 on the login-screen also provides a link to change the password, which gives a dialog to enter user_ID, old password and the new password twice, as illustrated here :



This login-screen has one option 'Open in new window'  as a 'tickbox' for which the default value is defined in the general configuration file 'config.php' of ABCD-Central with the parameter-name '$open_new_window' which can be set to "Y" (meaning after login ABCD will use a separate window, which avoids interference with other tabs or wrong use of the 'BACK' button of the browser) or "N" to simply open the ABCD window in the same area; the other parameter linked to this behaviour in 'config.php' is '$context_menu' which also can be set to "Y" or "N" to resp. allow or not the window-menu invoked by right-clicking on the page - again this can avoid wrong use of the 'BACK' button as this going back should preferably be performed through ABCD-interface buttons, not the ones of the browser who has no control on certain necessary ABCD-navigation issues.

By clicking on the arrow ('GO') 

the user will enter the main Central menu - adjusted to the profile granted. All options of this menu are discussed in the next sections of this manual. This includes the administration of users and linking them to a profile, since the users-database can be seen as just another ABCD-database for which general Database Management tools of ABCD are to be used.

1.6. Using LDAP authentification

The use of an LDAP authentification server is a new feature in ABCD 2.0 . The acronym stands for "Lightweight Directory Access Protocol" and in practice for ABCD means that you can use a dedicated, mostly institution-wide server which holds records of all authorized users. E.g in a university a 'registration office' could manage such a centrally co-ordinated list of students, which the library system consults whenever a user wants to login. This means that no longer the library itself is responsible for the up-to-date keeping of this list. E.g. when a student would be 'expelled' from the university, no longer the library needs to know this immediately in order to prevent such user - often quite frustrated ! - to enter the library(-system) and abuse it.

This LDAP functionality can be activated for the following ABCD-parts :

- Central login for operators
- Site : login for end-users in MySite
- Secs-Web : login into the Serials Control module.

The logics of using LDAP in ABCD is as follows :

- in config.php the parameter is switched ON or OFF (actually : true or false), so the use of LDAP is completely optional;
- whenever an operator (librarian) wants to log in into ABCD Central, or an end-user wants to log in into the 'MySite' module, the login-data entered are first checked against the LDAP-server; if that one responds 'clear' the login is granted, if not the login is rejected;
- if the LDAP-server does not know the login (which is different from 'rejects the login'), the login is subsequently checked against the local login-database ('acces' for operators, 'users' for patrons);

A pre-requirement for LDAP to work in ABCD is to have the extra PHP-library installed for LDAP :

sudo apt-get install php7.0-ldap *[substitute 7.0 by '5' if using PHP 5.x]*

and restart your Apache with a command like in Linux : *sudo service apache2 restart*

Other newly required files (as compared to versions pre-ABCD2.0) are :

- the ldap.php script to be copied into htdocs/central/common directory
- a dedicated IsisScript 'loginLDAP.xis' to be copied into the directory htdocs/central/dataentry/wxis
- new versions of inicio.php and inicio_mysite.php (done already by default in ABCD 2.0) with the following changes :
  - at the beginning (e.g. after the line require_once ("../config.php");) add a new line with the following instruction : require_once ("ldap.php");
  - at the end of the script change the part 'VerificarUsuario();' as follows, so as to use the new authentification if LDAP is used and the old one if not :

```
if (isset($arrHttp["login"])){ global $use_ldap; if($use_ldap) VerificarUsuarioLDAP();
else VerificarUsuario();
```

    - three new functions added : VerificarUsuarioLDAP(),LeerRegistroLDAP(),LoginNLDAP(), and Session($llave).

- a new CSS-element defined in the CSS htdocs/central/css/layout.css at the very end before the closing '}': .dl-horizontal { float: left; width: 45px; overflow: hidden; clear: left; text-align: left; text-overflow: ellipsis; white-space: nowrap; border-radius:150px; display: block; margin: 0 auto; margin-left: -50px;

- for the ABCD Loans module : a new version of the script htdocs/central/circulation/usuario_prestamos_presentar.php with

    - in the initial 'include' statements section, include *../common/ldap.php*

    - on line 627 added code : *if($use_ldap){ if(!Exist($arrHttp["usuario "])) { ProduceOutput("<h4>".$msgstr["ldapExi"]."</h4>",""); die; } }*

- for reference purpose we also note the additional changes for the Secs-Web module if LDAP is to be used :

    - in htdocs\secs-web\index.php add : *require_once(BVS_DIR. "/htdocs/central/config.php")* after *require_once("./common/ini/config.ini.php");* so as to check whether or not to use LDAP in Secs-Web

    - new version of class htdocs\secs-web\common\class\session.class.php : a new function $misession->checkLoginLDAP()is implemented which will be used if LDAP is found to be active in config.php (previous listitem), instead of the default $misession->checkLogin() function.

    - in htdocs\secs-web\common\class\session.class.php the SessionManager has added instructions at the very beginning to include the LDAP script and note its ON/OFF setting :

      require_once(BVS_DIR. "/htdocs/central/common/ldap.php");

      require_once(BVS_DIR. "/htdocs/central/config.php");

The LDAP configuration is to be defined as follows in CONFIG.PHP of the Central directory :

- $use_ldap : true or false, sets or unsets the use of LDAP;

- $ldap_host : the name or server-IP of the LDAP-server;

- $ldap_dn : the LDAP domain data in comma-delimited format;

- $ldap_search_context : additional data used by the LDAP-server to specify the use-context;

- $ldap_port : the port used by the LDAP-server;

- $ldap_pass : the password needed to access the LDAP-server;

In practice you will get all these standard-data from the LDAP-server manager.

An example configuration in the ABCD-Central config.php script for a free test-LDAP server is given below :

```
$use_ldap=false;

$ldap_host = "ldap://zflexldap.com";

$ldap_dn = "cn=ro_admin,ou=sysadmins,dc=zflexsoftware,dc=com";

$ldap_search_context = "ou=guests,dc=zflexsoftware,dc=com";

$ldap_port = "389";

$ldap_pass = "zflexpass";
```

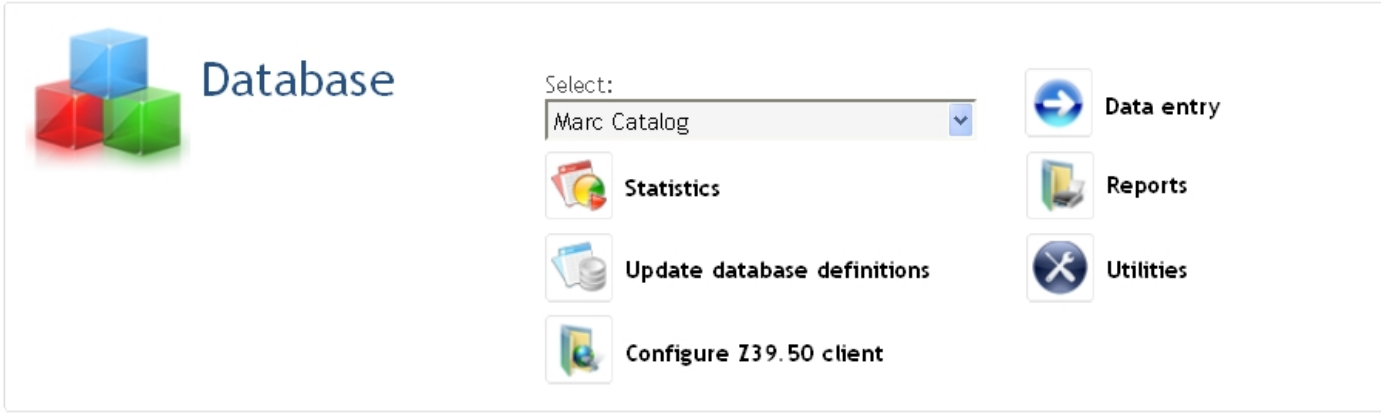
This example configuration is taken from a free LDAP server which can also be used for testing, see the URL : http://www.zflexldapadministrator.com . Sample users available are 'guest1' (password : guest1password), guest2 (password : guest2password) and guest3 (password : guest3password).

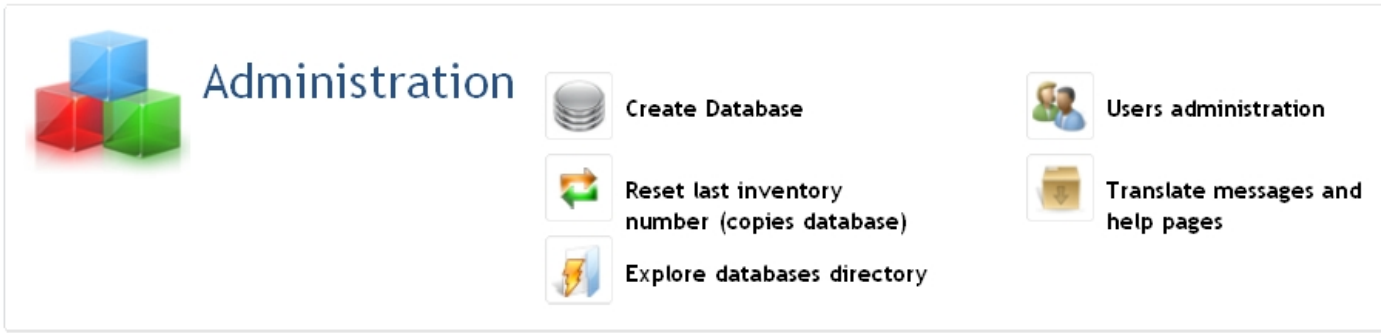
Please note that for your own ABCD LDAP you will probably work with the administrator in your organization or institution to define the correct configuration and do the testing.

2. Central module : database management

In this section we discuss briefly the main techniques of one of the most powerful functions of ABCD : creating new databases and modifying database structures. Since ISIS-databases don't require sophisticated 'normalized' relational structures and still can cope with elements in many-to-many relationships (like authors with publications), ABCD can be used to deal with any such 'locally' created database relatively easily. We recommend ABCD for environments where several such applications, like e.g. Institutional repositories, cultural heritage collections, vocabularies and ontologies or even just 'snippets' (loose textual information units), are likely to be created and used.

We discuss in the following sections each of the options given in the main menu of ABCD Database management :





Note

This menu is created in the PHP-script 'homepage.php' in the folder '\ABCD\www\htdocs\php\' where each login level gets its own function to create the menu (e.g. function MenuAdministrador() for the system administrator), so if it is necessary to change the sequence of the functions of this menu, this file has to be edited by someone who understands the HTML-coding inside.

We prefer to discuss the options in this main menu in a slightly different sequence (which can be obtained also in the menu by editing the above mentioned 'homepage.php' script), because before doing anything else the Users Administration should be at least performed once to define a local System Administrator and probably (quite) some other system users.

[!!] In view of the importance of the 'Data Entry' menu option here a dedicated section of this manual will be devoted to it following the discussion of the other Central functions. Also the procedures to follow to create copies and loan-objects for the inventory resp. loan-databases will be explained there as they are part of the Data Entry function.

2.1. Users administration

The Users administration option of the main ABCD Database Administration menu is a specific case of database-management, using mostly the general techniques discussed in this section, but for a specific database 'USERS' in which only the System Administrator can create profiles and ('register') new users or edit them.

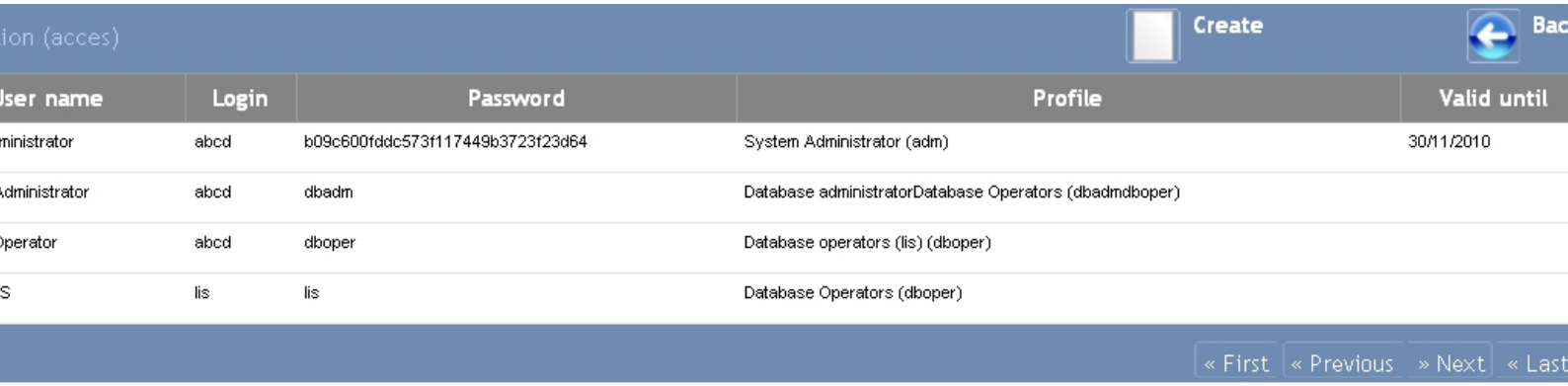
Note

IMPORTANT ! Before doing anything else, ABCD should get, by using this Users Administration option, a new, local System Administrator with his/her own login data ! The default login 'abcd/adm' will be widely known as it is published, so doesn't give *any* security indeed *!*

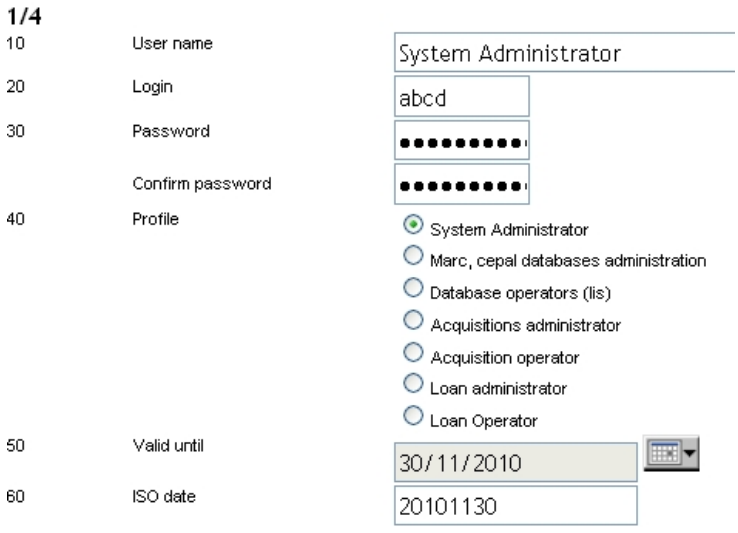
The screen following selection of the 'Users Administration' on the main Central menu will show 2 options :

- Users administration
- Create/edit profiles

The second option of managing the profiles is discussed elsewhere. The option on managing users is presented by first showing the existing users (there should be at least one 'System Administrator' user !) and giving the options to either edit these, delete or add (create) a new user.



When clicking on the 'record edit' icon (first one of the three presented for each user : ) the record with the user data will be shown in an interactive edit-form :



This edit form has the following parts :

1. the user name, which can be a full name
2. the login to be used in the login screen, mostly a shorter name
3. the password for this user
4. The profiles which have been created and which can be assigned to this user. A set of demo-profiles is included with the ABCD-installation package.
5. the 'expiry' date for the current user, in the 'normal' date-format (as defined in config.php) and the mandatory ISO-date format which will be created automatically by the software itself.
6. After editing the new user the record should be saved and then will be immediately allowed to use the system.

2.2. Creating a new database in ABCD

After selection of the 'Create Database' menu option, the following 3 elements need to be specified :

In the first box the software asks for the 'name of the database', which will be the real internal file name of the new database. These names no longer are confined to the old-style '6 characters' name of CDS/ISIS or WinISIS, but short names are still preferable. The name as presented to the users will be specified in the 2nd box : the 'description'.

Tip

Database names and descriptions can be approached directly in the file 'bases.tab' in the folder \ABCD \www\bases. In this file each database, provided to users, has one line with each two values : the 'name' and the 'description', separated by a pipe ('|') .

The 3rd box will always provide the options 'new database' - meaning creating a database from scratch - and 'WinISIS database' - meaning copying an existing structure of a (Win)ISIS database or in fact any ISIS-database with a FDT, FST and PFT. Then also the existing databases will be provided as models to be used as the basis from which to create the new database. We only deal with the first 2 options, as copying from an existing ABCD-database is quite straight-forward (ABCD simply creates the database by copying all necessary files into their appropriate folders and adding the new database to the list of existing databases).

The creation of a new database 'from scratch', meaning : not based on an existing model but starting from a zero-basis, involves understanding quite some ISIS-techniques, esp. the Formatting Language, because this will be used not only in the creation of the presentation format of the new database, but also in several ABCD-specific attributes of the fields (in both the FDT and data-entry worksheet) and the FST for indexing.

2.2.1. Creation of a new database from scratch

2.2.1.1. Editing the FDT

[

[!!] Since version 1.0 of ABCD two interfaces are provided for editing the FDT : one 'full' and one 'abbreviated'. The abbreviated form will not show the subfields unless the field itself is selected by its link in the first column - they will then appear in the subsequent form where all the details of the subfields can be edited, e.g. the width of the columns as 'no. of columns' in case the subfields are to be presented in a table rather than separate entry-fields (which is the default type of entry : Text/Textarea). This abbreviated FDT-editor is quite practical - and faster - in case of large complicated (i.e. using many subfields) structures such as MARC (even when only using 'minimal' and 'national' cataloging levels the number of possible elements is very high, p.e. there are 14 different record types with each their own 'polymorphic' field 8 and corresponding picklists, also for the MARC-indicators... resulting in more than 140 picklist-tables per language). For other, simpler structures the full FDT-editor can be used.

[!!] In the case the full FDT-editor is selected, or when in the abbreviated editor a field is presented in a detailed format, the link at the first column can be used to show the field in a 'vertical' detailed way, so presenting just the selected field in a normal form. This then again is more practical to deal with the individual elements to be defined.

[!!] The FDT-editor screen is probably the most complicated one of ABCD, as it presents an empty FDT, but since in ABCD this FDT also defines the worksheet for data-entry (or cataloging), unlike in other ISIS-softwares where a separate but simple 'FMT' (data entry worksheet) is defined, and since in addition ABCD uses quite some more advanced data-entry features such as picklists and validations, this step is rather demanding.

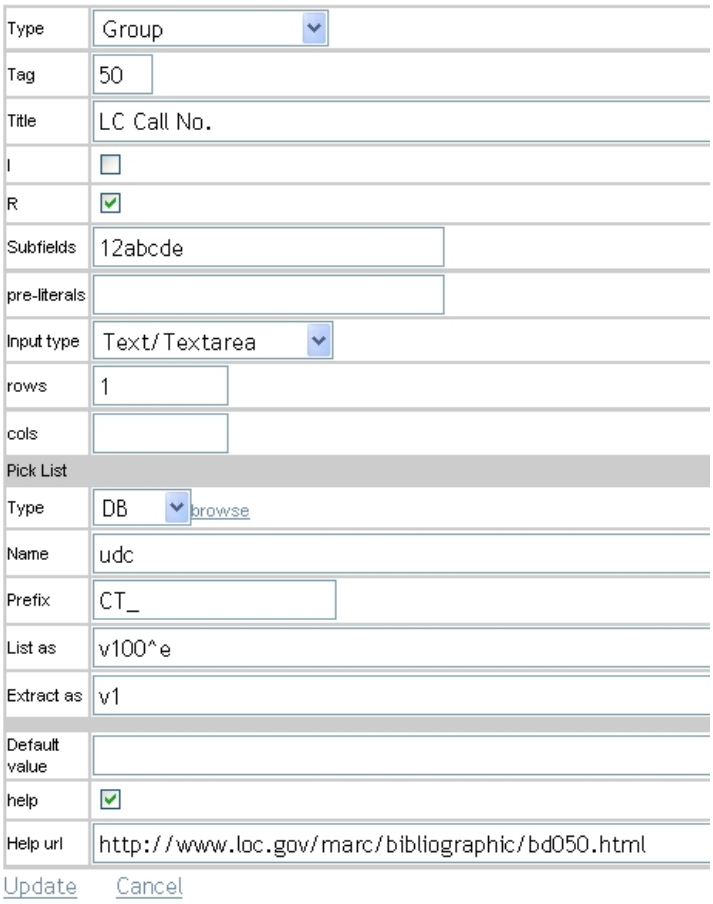
Tip

In order to 'edit' the form, double-click inside a cell of the table ! Simple-clicking will only select the row but not make the cell editable or invoke the menu attached to the cell.

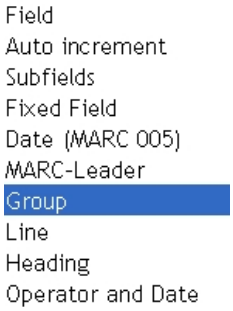
We will deal with each 'column' of the table now, but for a simple test it could be sufficient to simply only use the first 11 columns and the 2 last ones, the remaining part being dedicated to the optional definition of picklists :

2.2.1.1.1. A. Defining the fields

1. The first column : this is only a number, assigned by the system. It can be used however, if so desired, to open the row in a separate window to present all columns as separate boxes to interact with, by clicking on the hyperlink of the number itself. Such an empty row presentation looks like this :



2. The second column is about the 'type' of the field, which can be one of the following types :



   a. Field : the basic unit within a record, which should be used in case the element is NOT one of the following types : a subfield, a fixed field, a MARC-fixed field or -leader, or a 'group' which is a repeated field with subfields.

   b. [!!] Auto-increment : this is a special field which the system itself will normally manage, by taking the number saved in the small file 'control_number.cn' in the data-folder of the database, and adding 1 for each subsequent record using this field. When for some special reason this automatic numbering needs to be manually changed, the 'assign'-link will allow to do so.

   c. [!!] A subfield : when previously a field was created with values for subfields given in the 'subfields'-column (see infra), ABCD expects subsequently ALL subfields to be described immediately following the field to which they belong. The subfield-identifier then has to be put into the column for 'subfields' and the sub-field-name as the field-name. Indicators as used in MARC-structures should be treated as subfields but with numerical identifiers.

   d. A fixed field : allows to create a simple field with a fixed length

e. [!!] Date (MARC 005) : this is the special date-field with tag 005 as used in MARC.

f. [!!] MARC-Leader : the fixed-structure MARC leader field. Dedicated support for all positions of this special field is given.

g. 'Group' : this is a subfielded field which is repeatable. As with a normal subfielded field, it should be followed immediately with the subfields belonging to the group, but each series of subfields will be repeatable. A typical example of a group is the 'author'-field, as an author definition contains mostly several 'parts' (or subfields) such as name, first name, role etc.. and documents in principle can have more than one author, therefore this field should be repeatable. The 'table' data-entry element is quite suitable to represent such a complicated field in a data-entry form : each row of the table will be a repeat of the field and the columns represent the subfields. [!!] Sizes of the columns can be set as the 'row' -parameter of the first subfield definition row. The number of rows in the table will define how many occurrences will be shown, but ABCD will always add one empty row to allow creation of additional occurrences.

h. A 'line' is just a graphical element to separate fields in the worksheet for data-entry. It doesn't need any further specifications.

i. A 'heading' is a short text which can define a 'section' in the data-entry worksheet in order to 'group' fields together; ABCD will automatically provide hyperlinks-within-the-form to navigate directly to any of the defined headings. In MARC a typical 'header' could be e.g. 'primary entries' or 'secundary entries'.

j. Operator and Date : this field will be automaticall filled in by ABCD with the name of the logged-in operator who edits the records and the time-stamp of creation.

3. The third column is used to define the 'tag' or numerical identifier of the field, as required by ISO-2709. These numbers range from 1 to 999. ABCD (as does CISIS) uses many fields with values higher than 1000 for internal, mostly temporary uses. Field-tags can be arbitrary (e.g. 1, 2, 3...) but often should comply with existing standards, e.g. MARC21 uses '245' for the main title field. It is the designer's (you..) responsibility to decide on a proper list of field-tags.

4. Column no. 4 allows to identify the field with a 'name' or 'title' in order to explain the meaning of the field-tag. Here any - preferably short - indication can be used in the actual language.

   ## Note

   ABCD, unlike WinISIS and other ISIS-variants, allows creation of FDT for each language used, so field names can be language-dependent !

5. Column no. 5 allows to select one - and only one ! - field in a database to be used as the 'I'dentifier field on which the lists (see e.g. the 'A-Z selection tool') will be based. This is not the same as the 'primary key' field in relational databases, but indeed defines the field marked as being 'I' - since this column only allows to 'activate' or 'de-activate' - and hence being used to sort the records for direct selection.

6. Column no. 6, as the previous one, only allows 'activating' or 'de-activating', in this case to denote whether a field is repeatable or not. This is an important decision to be taken, according to the designers view on the database structure, but different reasonings can be applied. E.g. in a simple structure the 'title'-field could be made 'repeatable' to also contain all types of titles (e.g. sub-titles, translated and original titles...) in order to confront the users with only one 'title'-field. MARC wants all types of titles to be in different fields, but still prefers the title-proper field to be repeatable ! We suggest to make fields repeatable in case of any doubt, as it is easier to use a repeatable field only once, rather than displaying repeats properly in a field defined as non-repeatable (in that case the field definition has to be changed in the FDT ánd the PFT's need to be adjusted !).

7. Column no. 7 allows to define the single-characters (0-9 or a-z, case-insensitive) which will identify the subfields, if any. Remember that if subfields are given here, the subsequent lines or rows SHOULD deal with each of them individually, otherwise a logical error will be given. [!!] In the table-row defining the specific subfield, this column should contain the subfield-identifier.

8. Column 8 allows to *optionally* define characters such as punctuation (,:: etc.) which will be converted - in the same sequence, so be careful and make sure there is consistency ! - into the subfield identifiers. This allows

data-entry staff to use the punctuation instead of the rather less-obivous sub-field identifiers, but remember that ABCD also allows to deal with each subfield individually without having to bother about the identifiers (see the section on 'data-entry').

9. [!!] Column 9 allows to define the type of HTML-input component the data-entry form will provide, where there are 15 possibilities :

   a. Text/Textarea will present a text-box of variable length. The number indicated in the 'rows' column defines the number of lines which will be presented in this box.

   b. Text (fixed length) will present a text-box of fixed length. The number indicated in the 'columns' column defines the number of characters which will be allowed to enter in this box.

   c. Table will present a table in which, in the rows, occurrences of the field can be entered, and in the columns, subfields of that occurrence can be entered. The numbers in resp. the 'rows' and 'columns' columns define, as can be expected, the number of occurrences and subfields which will be captured for this field.

   d. The 'Password' option gives a text-box in which the box will be filled with * for each character typed, to hide the contents of this special field. If the MD5 option is activated in the config.php file, the passwords will be encrypted.

   e. Date is the option to capture a date, with the assistance of a JavaScript control to offer date selection from a calendar.

   f. [!!] ISO-date is the ISO-formatted (yyyymmdd) date field, mostly filled-in automatically by ABCD for internal use.

   g. Select simple will allow selection of only one element from a list of pre-defined options.

   h. Select multiple will allow selection of more than one element from a list of pre-defined options.

   i. Checkbox is the option to allow one or more boxes to be 'ticked' (activated) in order to select them.

   j. Radio is the option to allow only one round button to be activated in order to select the related option.

   k. HTML area is the option to present to the user a full HTML-editor (JavaScript control, in ABCD we use FCKEditor) for editing, in WYSIWYG-mode, text with HTML-codes.

   l. External HTML is the option to create, as in 10, a text-with-HTML-codes, but this will be stored not in the database but as an external file, with a link in the ISIS-record to this file.

   m. Upload file is the option to present a JavaScript control for loading files to the server and create a link accordingly.

   n. [!!] Read-only is a field which will not be editable after it has been entered as it is read-only.

   o. [!!] Hidden : a field which will not be shown after having been entered.

10.Column 10 allows to define the number of 'rows' the data-input HTML-component will offer. Depending on the exact type selected in the previous column, this means the number of text-lines which can be displayed in the box, or the number of occurrences of a field (in a table to contain a 'group') will be allowed etc.

11.As with the previous column, this column allows to define a number, but this time for the number of 'columns' in the HTML-input component, which can be e.g. the number of characters (in a fixed-length textbox) or the number of sub-fields (in a table), or more generally the 'width' of the box.

In principle, after having defined these 11 columns and if no need exists to define 'picklist', what is remaining is to simply define, if wanted, a 'default' value for this field in the last-but-one column, and to indicate if a help-page for the current field is to be made available. [!!] These help-pages are to be referred to as URL's (local files or online webpages).

At the end of the table options are provided to save the table, but also to test and validate it

Here the 'Test' and 'Validate' options will resp. display the resulting form for getting an idea about the result and display the table in a different window with a message indicating wether any logical or grammatical errors are present in the table. It goes without saying that such errors need to be corrected first before 'saving' or 'updating' the FDT with the last option presented here.

The 'List' option provides a listing of the table in a separate window, e.g. allowing printing it or saving it as a separate file.

2.2.1.1.2. B. The definition of picklists

We continue here discussing the columns of the FDT, this time with the columns 12-20, which (except for the last 2) deal with the definition of 'pick-lists' to be presented in the data-entry form in order to support terminology control, authority control or simply to facilitate the data entry by providing the options available.

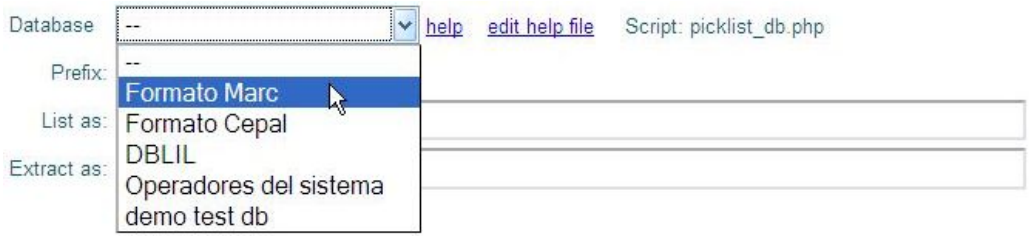
12.Type of picklist : here we define the type of the control list to be used, with the following options : [!!] Database, Table or Thesaurus.

    A database is actually an ISIS-database with its Inverted File, so providing an almost unlimited number of possibilities, but being rather a complicated solution. [!!] A simple pick-list (table) will be based on an ASCII (or TXT-)file containing on each line one option. The 'Thesaurus'-option is in fact an (ISIS-)database again, but this time using a specific field structure with references to the different (and standardized) thesaurus hierarchical relationships, such as 'synonym', 'broader term', 'narrower term', 'scope note', 'use for' or 'used for'. Such thesaurus-database normally provides ways of 'navigating' to the related terms and therefore offers even a higher level of support for data-entry, by providing in such a scientific way descriptors of a given scientific field or topic.

13.Name : here the name of either the database or the file on which the control list is based, is to be put. This can also be taken from the 'browse' option in the 15th column (see infra).

14.Prefix : here the short prefix should be put in case a database is producing the picklist, as that list will be produced from the Inverted File of that database and that often will be divided in 'sections' by the use of a prefix - this is the one to be put here to allow partial presentation of the Inverted File. If e.g. the control list is used to facilitate the entry of publisher names (as many come back often), probably the database publishers are indexed with a prefix such as 'PU=', then putting this prefix here will only display the section of the IF with that prefix.

15.'Browse' : this is a link which, when clicked, will open the following separate window, which allows defining some information on the picklist-database in separate boxes, first of all the name of the database which can be selected form the ones already available.



16. Display format (or 'List as') denotes the PFT which defines how the values in the list will be displayed with the Formatting Language. [!!] Here either an 'inline' PFT can be given, e.g. 'v11', or a reference to an external format can be given as '@myformat.pft'. This external format has to be written following a pre-defined pattern in order to be correctly interpreted.

    See the example here used for the authority files of the MARC database : @autoridades.pft:

    select e3

    case 1: v1

    case 100: v100^a,`$$$`v100^a

    case 110: v110^a,`$$$`v110

    case 111: v111^a,`$$$`v111

    case 245: v245^a,`$$$`f(mfn,1,0)

    case 260: v260^a," : "v260^b

    case 270: v270

    case 340: v340

    ...

    endsel

    ## Note

    [!!] In case the PFT contains pipes (|) it CAN NOT be put inline into the FDT but has to be put in an external PFT referred to from this cell (this is because the pipes are also used as separators for the column values of the FDT table as stored in ASCII-format).

17. Extract as : defines, again with the Formatting Language, how the contents of the field needs to be exactly extracted from the field values in the record to which the entry in the list (as an Inverted File posting) points. If this value is omitted, the values will be kept in the format defined as 'display format' in the previous column. If the display format is a pre-defined format (@xxxx) and follows the instruction to separate the display format from the extraction format by $$$, this part should be left empty.

18. Default value : here the default value can be put which could serve for fields which often have, in the specific case of the database, the same value, which then will already be presented automatically.

19. Help : this is a tick-box (active or not) to indicate whether a help-file for this field should be presented in the worksheet. The help-pages are stored in the folder bases/*dbn*/ayudas, where *dbn* represents the name of the database.

2.2.1.1.2.1. The FST Definition

After having defined the list of fields (and picklists for them), ISIS expects the manager, who is creating a new database, to define not only which fields will be indexed, but also exactly how they should be indexed. That is what the **F**ield **S**elect **T**able (FST) is meant to do.

There are excellent documents available as 'help-pages' within ABCD on this complicated ISIS-technique (and included as annex with this manual), so here we only present the main purpose of the three FST-columns.

The FST contains 3 columns :

1. The identifier

   This is a tag (a number) which will be used as the field from where the index term was taken, e.g. to allow field-limitations in searching. Mostly this tag will correspond to the actual field from where the value was taken, but it can also be a 'virtual' field (e.g. to group several titles in one 'title-field' to simplify the structure for the search). For example, one could create the very popular 'ALL FIELDS' search (Google-users don't know anything else !) by indexing all significant fields with one and the same IDentifier, e.g. '999' to allow a 'non-fielded' search.

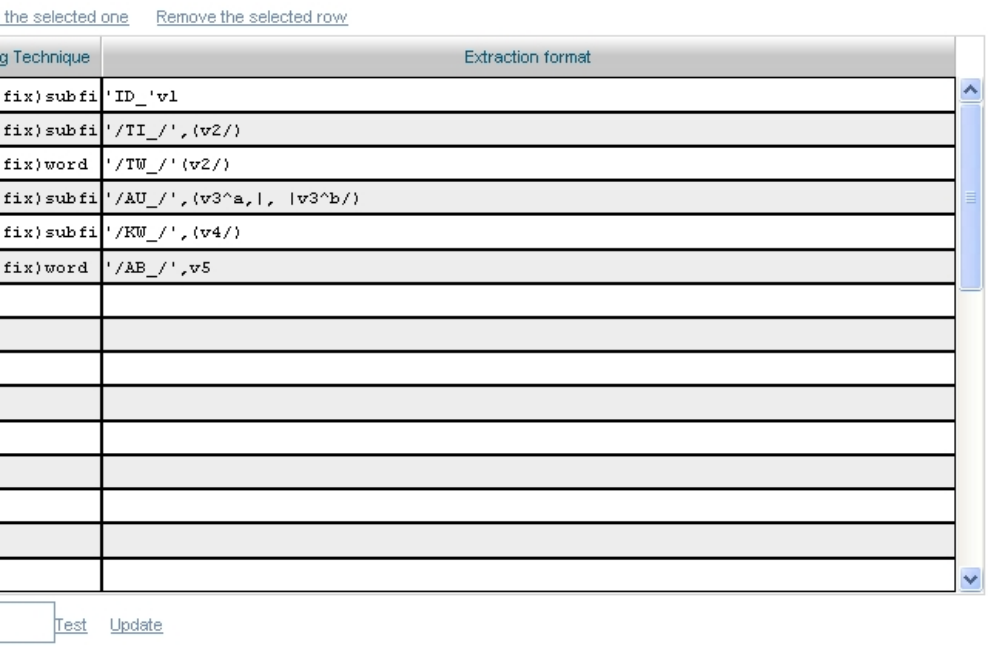
2. The indexing technique

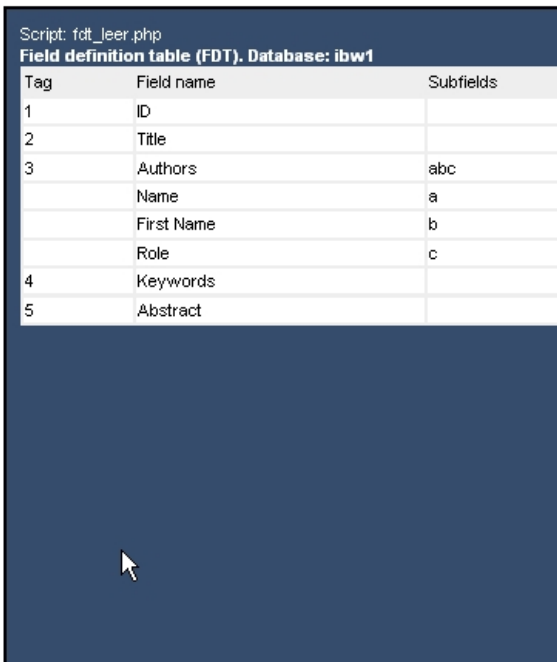
   ISIS avails 9 techniques for indexing, but basically these can be reduced to two main options : the full field (abbreviated to the first 60 characters in ABCD) - called 'by line' - or a full-text indexing - called 'by word'. Indexing techniques from 5-9 are optimized for indexing using a 'prefix' (a short tag preceding the values to group the values in the same alphabetical section of the overall index or inverted file).

3. The extraction format

   Here the actual format to produce the sting to be indexed is specified using the ISIS Formatting Language. All features of the Formatting Language (except presentation features) can be used, including REFerring to other databases.

The interface of ABCD makes the creation of such an FST as easy as is possible (but it isn't easy really, because of the advanced possibilities available!), by not only providing the FST in 3 editable columns, but also, as a reference, the FDT from which fields can be used with their tags, and also indicating whether they have subfields and are repeatable or not.



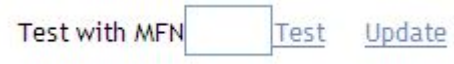


As can be seen in [!!] this example, which uses very simple extraction formats, we always prefer prefixes to be used, e.g. 'ID_'. In view of some built-in options of the iAH OPAC interface for ABCD, we recommend the use of 3-character prefixes ending with an underscore ('_').

As the Formatting Language (ISIS' most powerful feature) can be used here in its full power (without the graphical presentation features), values can be processed before then enter the dictionary, e.g. 'N:', f(mfn,1,0) produces the recordnumber or MFN, formatted (f) as a string, but more complicated examples can be given, e.g. .

- a combination of several fields or subfields with added punctuation
- formats using the REF-function to refer to external databases to take values from there after locating the MFN with the L-function - doing so e.g. codes can be converted into values for the dictionary.

After having edited the FST, it can be tested with any record of your database to check whether the actual values which will be indexed indeed comply with what was intended.



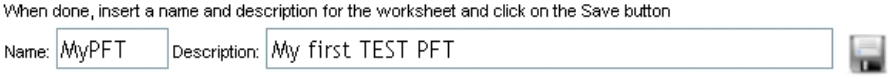
2.2.1.1.2.2. The worksheet editor

As in any ISIS-application, ABCD allows the creation of many different worksheets for different purposes, e.g. to deal with only a smaller subset of fields, or with the fields in a different sequence etc.

For that reason ABCD offers in this option an easy tool to select which fields to present in the (new or edited) worksheet and in which sequence to present them :

In case an existing worksheet needs to be edited, it can be selected (in the upper part of the interface) from a list - where possibly also an existing worksheet can be deleted is so desired. The main part of the worksheet presents the list of available fields with possibility to copy any field () or all fields () to the worksheet list at the right side.

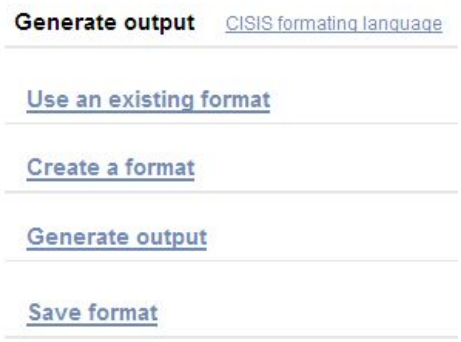
Finally the worksheet can be saved into the system with an internal name (short and lower-case only) and a description for easy identification.



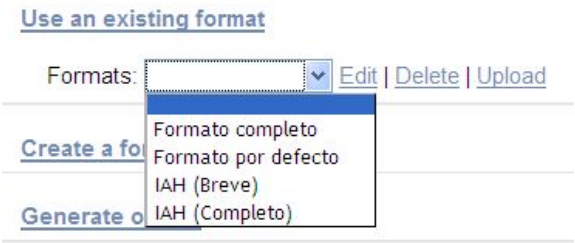
2.2.1.1.2.3. The PFT Editor

Since the ISIS Formatting Language is a very powerful tool with many possibilities, a dedicated document on the (C)ISIS Formatting Language is added as annex to this Manual on ABCD, but it can also be consulted from a link provided in this PFT-interface of ABCD. Here we just briefly explain how the ABCD interface allows easy creation of either HTML-formats (for presentation on the web-pages) or 'delimited' formats (for export to a comma-delimited file).

The PFT-Editor has 4 parts :



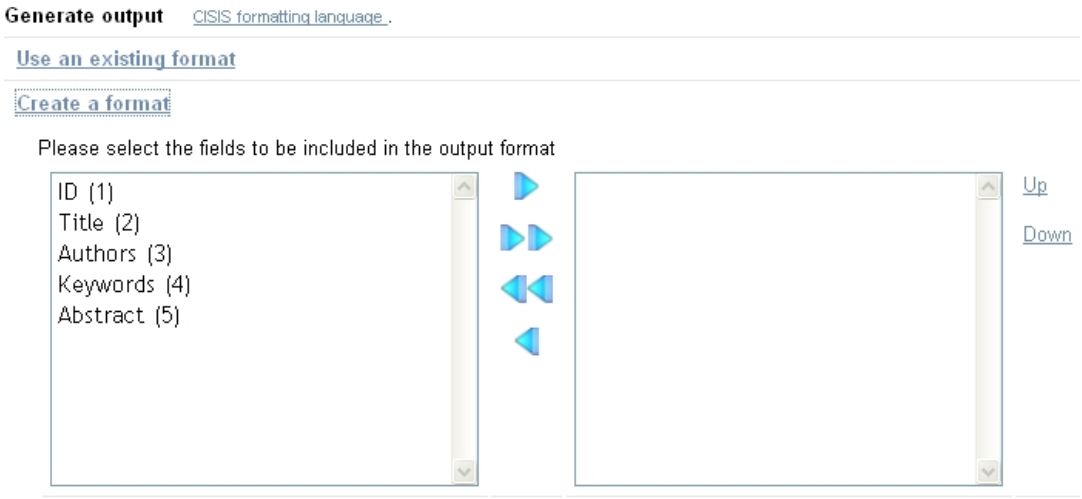
1. Use an existing format : a list of existing PFT's will be available to select from. It can be also deleted or uploaded from an external file if not yet available. The format then will be presented with an editor to make changes into it.



2. Create a format : as with other table editors in ABCD, first a list of available fields is presented, which can be copied either individually or altogether into the format and then re-ordered.

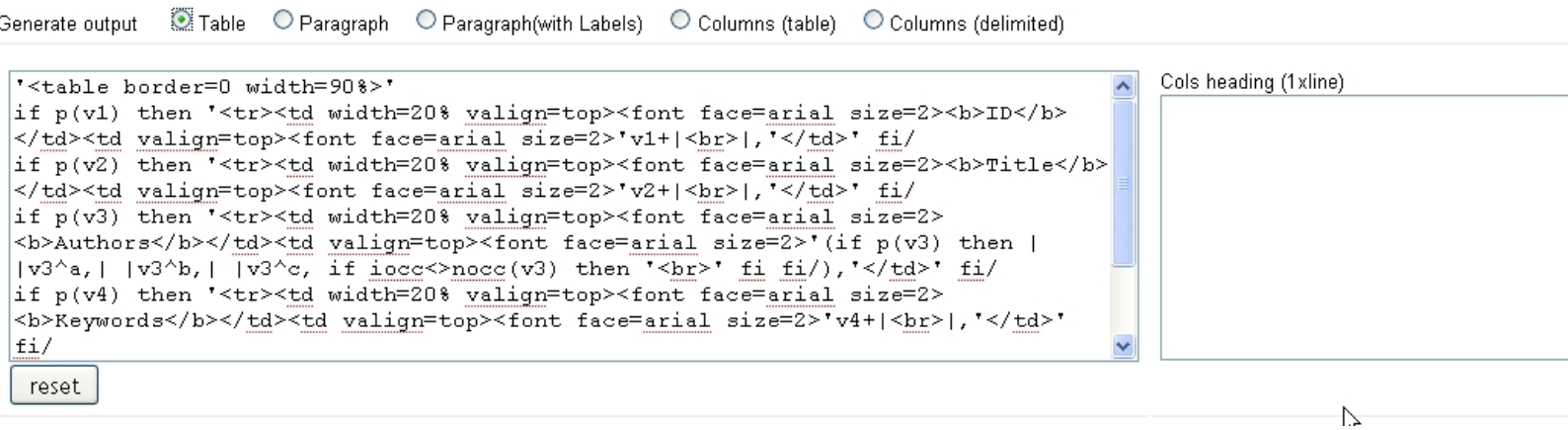
Note

Windows list-selection tricks can be used here : Ctrl-click to add an entry into your selection or Shift-click to select all options upto the cursor position.

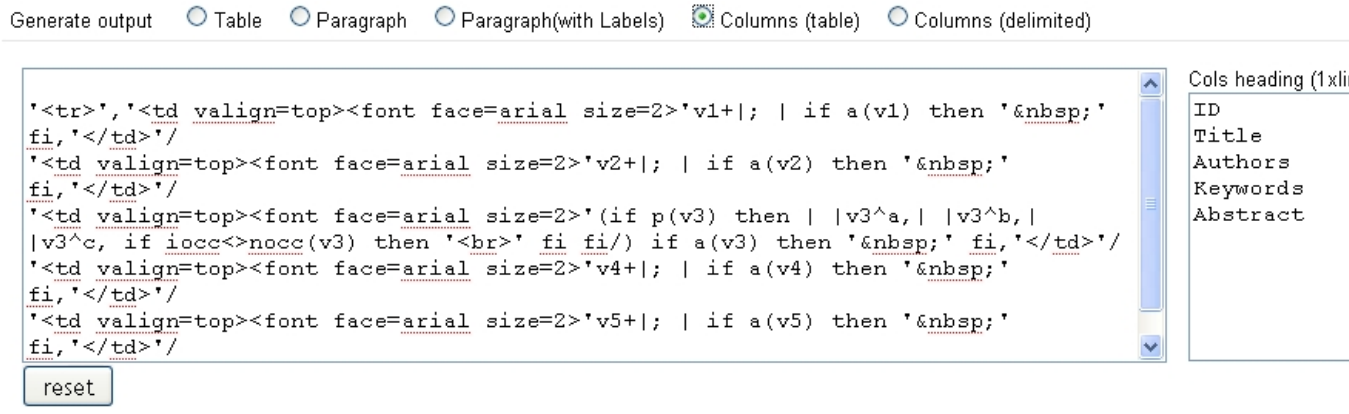


3. Generate output

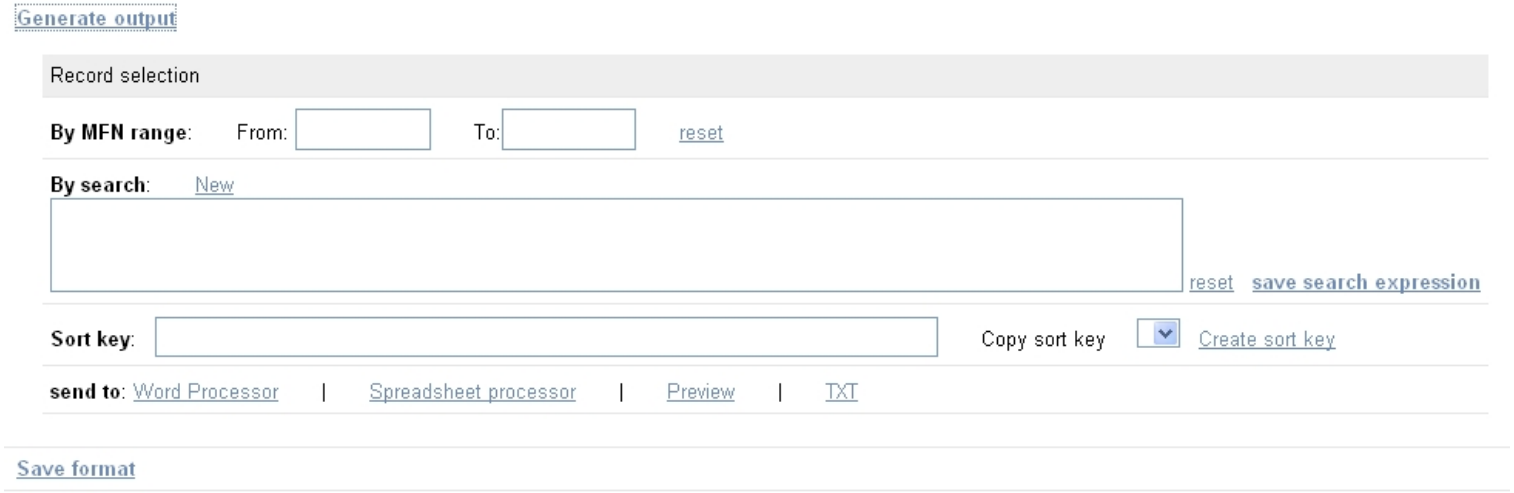
   As mentioned and shown above, generating output to test the PFT can be one of three pre-defined standard ways of presenting data from your database : either a 'table-formatted' web-page (in colums) or 'paragraph-formatted' webpage (no columns), or - alternatively for quite different purposes - a delimited format for export to other software. ABCD will immediately generate the necessary code, combining HTML-tags as quoted 'literals' with values from the fields (Vx).



[!!] In case a 'column'-based format ('delimited' will allow export to softwares such as spreadsheets and statistical analysis tools) is selected, in the right square the (sequende of the) headers of the columns can be defined, by default they will be the field-names already defined.



This format then can indeed be 'tested' immediately on either a range or records or a selection defined by a search-statement.



This will result, when 'previewed' within the interface as opposed to 'sent to a document or worksheet' (this last option is ideal when outputting data as 'delimited'), into a display format like the following :

ID	1
Title	My first title
Authors	Guns Raf de Smet Egbert
Keywords	ABCD Oefening CNTDI
Abstract	ISIS is being used by ten-thousands of us and **BIREME** (for mostly Latin America). I umentation centres (it has a `dominant` p high number of users, many of them ofter and with relatively poor ICT-skills. This cre At the 3rd World Congress on ISIS (Rio de make ISIS fully `FOSS` and co-ordinated I portal.unesco.org/ci/en/ev.php-URL_ID=2 Summarizing the long history of ISIS, one ples, a strong tradition and a world-wide t of-the-art technological development.

Note

In this example subfield codes are still visible, but with simply adding a 'mode' statement like 'mhl' into the format ISIS will hide them.

The other functions of defining/editing ABCD-database definitions will be discussed below when dealing with the'Update database definition' section.

2.2.2. Copying an existing WinISIS database

For creating an ABCD-database from your existing WinISIS database, here are the steps to be followed :

1. Export your existing records into an ISO-export file using WinISIS (or another ISIS-tool allowing ISO-export); remember where you have saved this .ISO export file, normally it will reside in the WORK-folder of your WinISIS installation. No specific parameters need to be set, unless of course you would only want to use a subset of the records in that database (by using MFN-range minimum and maximum or a search result) or you need to 'convert' (reformat) the records before entering them into ABCD by the use of a 'reformatting FST'.

2. Assign in ABCD, after having selected the 'Import from WinISIS' option, a name and a description - as with a new database, see supra. Then select your WinISIS database using the list in the 'Create from' part of the dialog.

3. Select the FDT belonging to that database and click on 'Upload' in order to have the FDT loaded into the ABCD environment of the new database.

4. Select the FST belonging to that database and click on 'Upload' in order to have the FST loaded into the ABCD environment of the new database.

5. Select the PFT belonging to that database and click on 'Upload' in order to have the PFT loaded into the ABCD environment of the new database.

Warning

Most WinISIS databases use a default PFT (with the name of the database) which contains typical Windows (as opposed to HTML) codes, such as e.g. 'BOX', 'FS' etc. These will result in a 'grammatical' error when later opening this in ABCD, so it is better to avoid this by selecting a PFT without such

typical Windows-elements ! If not available, remember to re-create a HTML-based format within ABCD to replace the default PFT for your new database.

6. Click on the 'Create Database' option in order for ABCD to start writing the necessary folders and files for your new database. A message about successful creation (or not, in case of problems) will be displayed on your screen. Also you will be reminded of the fact that without assigning this database as accessible to at least one user, you won't be able to use this database.

7. Now you can open the new database, as it has become part of the list, in the main database management window.

8. As the database can be opened but with 0 records, the first thing to do is to import the ISO-records created in the first step of this series. To this end, click on the 'Utils' icon in the main toolbar of this data-entry screen (as described in the section dedicated to this) and select the option 'ISO import'. This procedure further, as can be expected, involves the selection of the source ISO-file, which then should be 'uploaded'.

9. Now, a bit strange, the ISO-file is ready for being effectively imported into the database. For this, click on the 'Utils' icon again in the toolbar and select 'ISO-import', where now the uploaded ISO-file is available for effectively importing (converting) into your ABCD-database. The software will now ask you if it is o.k. to indeed start importing the ISO-records from the selected file. The list of imported records will be shown on your screen to monitor its progress and success.

10. If your newly imported records don't immediately show up in the database, re-open the database from the main menu, this will refresh the database parameters.

11. Now the records should be visible and editable as normal records, only they have not yet been indexed into the Inverted File, so use the option 'Inverted File' update in the 'Other utils' section of the 'Utils' screen.

Warning

If your imported series of records is quite large (e.g. above a few hundreds), it is possible, depending on the system you are working on, that the process will be too long for the web-server (Apache in most cases) and it won't be allowed to finish. For this reason it will be necessary in such cases to perform the Inverted File generation action not from ABCD (as a web-environment) but directly from the command-line, using the dedicated CISIS-tools (for which another section of this manual will give the details).

2.2.3. Copying an existing ABCD database

This last option is the easiest to perform, as only a new name and description need to be entered, after which ABCD will simply re-produce all necessary files into a new folder structure for the new database. The source databases from which you can choose are the ones listed in the database-menu, in other words : the database descriptions listed in the file 'bases.dat' in the ABCD bases-folder.

The system will simply - as above - list all files copied and created in their proper folder-structure and that is it ! An empty database, as a copy of the existing ABCD-database but with new name and description, will be available for normal use.

2.3. Update database definitions

From this option it is possible to edit all existing 'structures' or definition tables related to a database in ABCD. In comparison with 'normal' ISIS-databases, and in order to support some more advanced features in ABCD, there are some more of such database definition elements, as can be seen from the following 'database definitions' menu :

In fact only the first four tables are used in other ISIS-environments : the Field Definition Table (FDT), the Field Select Table (FST), the FMT or edit-worksheet and the Print Format Table or PFT. Since we needed these also in order to create a 'new' database in ABCD, they were discussed in the according section above, the only difference being that instead of an empty table a pre-filled table with the already existing definitions will be presented by ABCD.

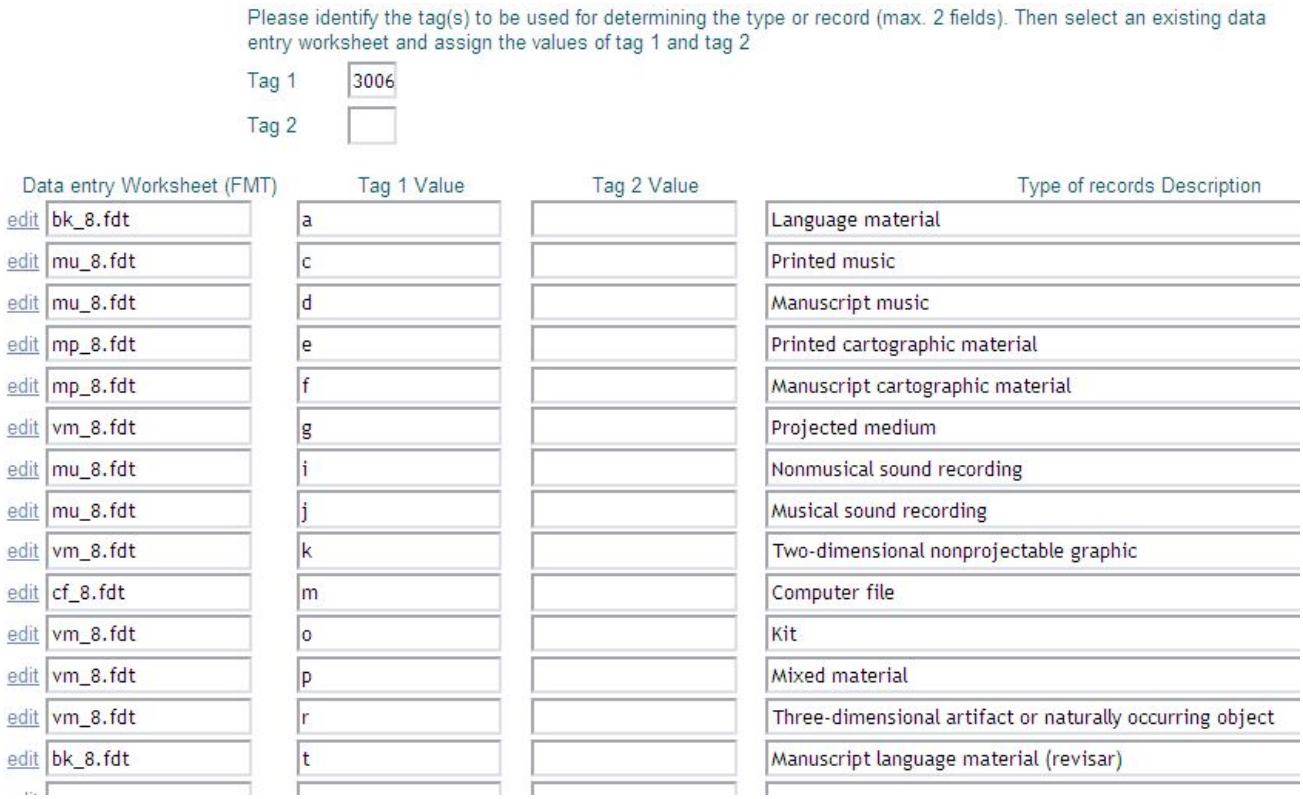
Let us briefly deal with the additional definitions for ABCD-databases now.

2.3.1. Type of records

Some database-structures, such as MARC, require the 'type of the record' to be specifically coded into a dedicated field of the record. The software can then use this code to adjust many features to the specific needs for the type indicated, e.g. worksheets and print-formats can be different according to this type, or simply - as is the case with MARC - the format wants to be very detailed.

The 'type of record' information needs to be gathered at the beginning of the creation of a new record, so a list of types defined (and therefore 'available') will be presented as links, each one leading to the appropriate subsequent data-entry form, as can be seen from the MARC demo in ABCD.

[!!] In order to define such types, ABCD uses a table 'Typeofrecord.tab' - located in the 'def' folder for the actual language within the www/bases/[DB]/ folder. This file is - as is often the case in ABCD - an ASCII-file which contains, for each type defined, 4 values separated by a '|' (pipe-character), so this can be edited directly with an ASCII-editor (e.g. Notepad), but within ABCD an easier-to-use table format is presented :



Please identify the tag(s) to be used for determining the type or record (max. 2 fields). Then select an existing data entry worksheet and assign the values of tag 1 and tag 2

Tag 1 3006

Tag 2

	Data entry Worksheet (FMT)	Tag 1 Value	Tag 2 Value	Type of records Description
edit	bk_8.fdt	a		Language material
edit	mu_8.fdt	c		Printed music
edit	mu_8.fdt	d		Manuscript music
edit	mp_8.fdt	e		Printed cartographic material
edit	mp_8.fdt	f		Manuscript cartographic material
edit	vm_8.fdt	g		Projected medium
edit	mu_8.fdt	i		Nonmusical sound recording
edit	mu_8.fdt	j		Musical sound recording
edit	vm_8.fdt	k		Two-dimensional nonprojectable graphic
edit	cf_8.fdt	m		Computer file
edit	vm_8.fdt	o		Kit
edit	vm_8.fdt	p		Mixed material
edit	vm_8.fdt	r		Three-dimensional artifact or naturally occurring object
edit	bk_8.fdt	t		Manuscript language material (revisar)

The example above is the MARC record-type definition, which is kept in (internal) tag 3006 and has the following 4 columns, each containing values for each type :

- the name of the worksheet to be used for the given type of record - with the 'edit'-link next to this first column one can also immediately edit the worksheet
- the 'Tag1' value, which is in fact a one-character code to internally identify the type of record
- the 'Tag2' value, not used in this case
- the description of the type as it will appear in the list of available types.

Clicking on 'update' below the form will save the table with any changes made.

2.3.2. Record validation

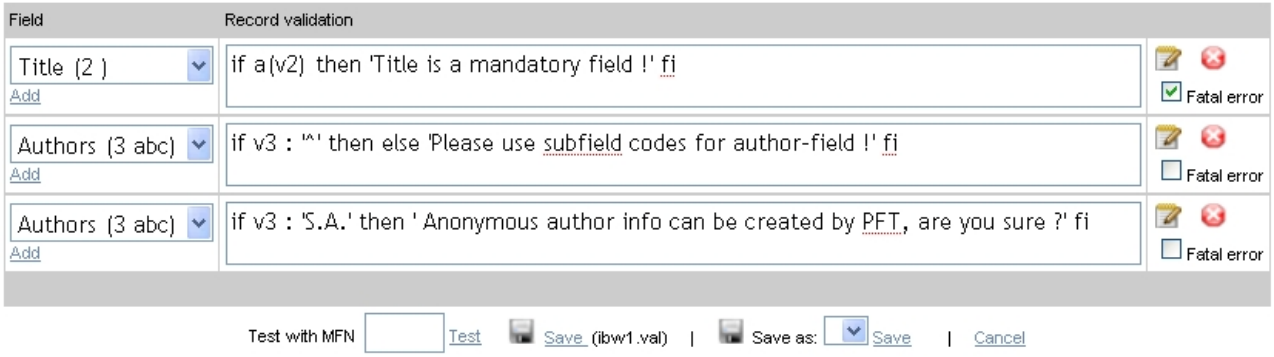
[!!} Record validation allows the database manager to define criteria against which input for a field can and will be checked before entering it actually in the fields, or after registration of the record. Record validation in ABCD can relate to one of the following elements :

- record validation : conditions can be given for each field
- begin format : code can be given to be executed when a (new) record is created, e.g. the date of creation can be added with the date() instruction
- end format : code can be given to be executed when the record is saved, e.g. the date of last update can be added with the date() instruction.

  After selecting on of the three options above, clicking on one of the listed (as pre-defined) record-types will show the editor where the validation conditions can be defined.

These criteria need to be formulated into - no surprise ! - the ISIS Formatting Language. With the Formatting Language one can check a condition and if (not) met, an error message, which will be shown on the next screen, can be produced by the format. [!!] If the validation criterion was defined as 'fatal' however the record will not be saved and the error has to be corrected first.

In this menu-option from the 'Update database definitions' menu each defined field will be listed with a box to enter the validation statement. E.g. :

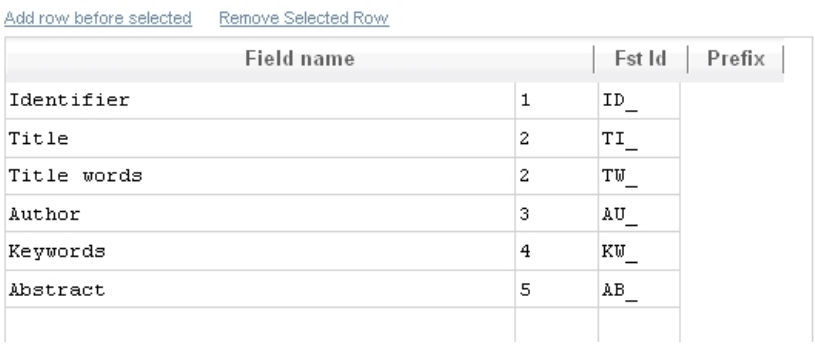


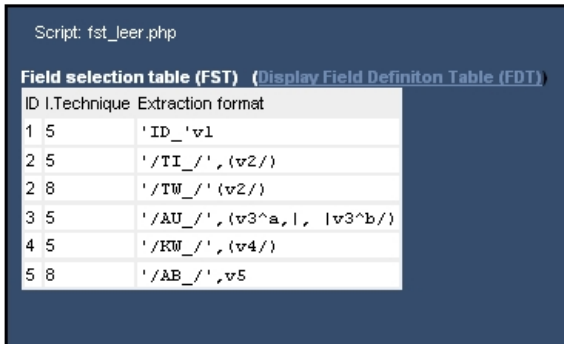
The format used here checks on the 'absence' of a value in field with tag 2 and if indeed absent will produce an error message that this mandatory field is missing. [!!] As stated above, errors can be marked as 'fatal' or not. With the 'ADD' link in the left-most part for each field. As shown in the illustration above, with the 'ADD'-link in the left-most column one can add more fields, even fields already used with another validation criterion to be applied.

When clicking on the 'edit' icon to the right of the edit-window for this field, the box re-appears in a separate small window for editing and the statement can also be tested on a record to see if it is doing the right thing. After editing one has to click on 'send' to put the possibly edited validation format back to the main table.

2.3.3. Advanced search form

The advanced search form is the one used in ABCD at two locations : within the cataloging module to allow the cataloger to quickly and/or efficiently identify a specific record for editing (or duplication checking or copying) and in the OPAC as the advanced search form. Here ABCD simply offers an editor for the table which defines that search form with 3 columns :





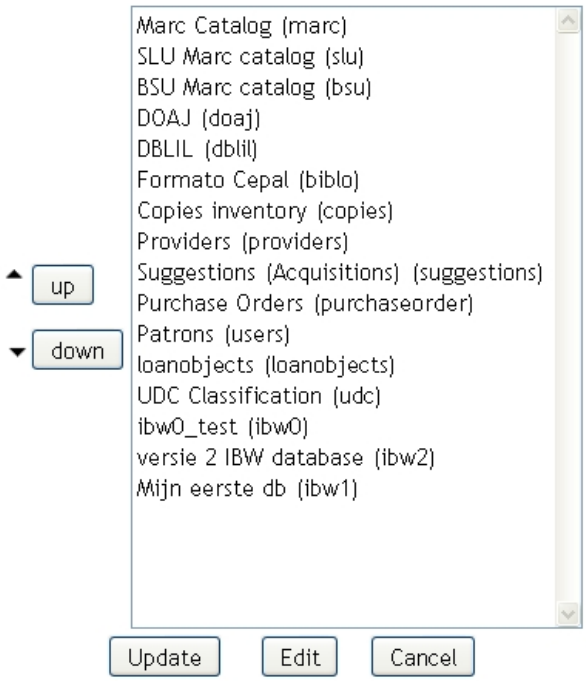
- The first column is the field-name or 'index' as it will appear in the search-form
- The second column gives the identifier used for the given field (or in fact combination of fields) in the FST
- The third and last column keeps the prefix or fixed start-string which is used (if any) in the ISIS Inverted File for this index.

  ABCD will also present, next (to the right) to this table, the existing FST to facilitate the identification of indexes (fields for searching) and the identifiers used.

  Clicking on 'update' saves the table, which in fact is a file 'busqueda.tab', stored in the language subfolder of the 'pfts' folder within the database folder.

2.3.4. Available databases list or table

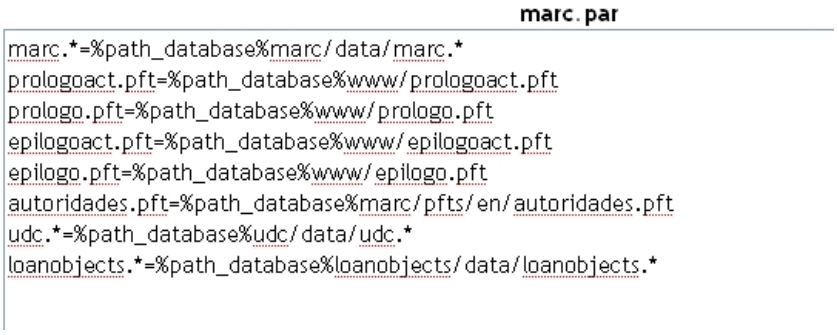
Here simply a list is given of the databases being defined as 'available' in the ABCD-system. Actions allowed here are only changing the sequence (by moving up or down) of the databases in the list and saving the changed list.



For adding or deleting databases one has to actually create the new database or delete the one to be taken out of the list. ABCD will take care of the changes in this list automatically. [!!} Databases can be moved up or down by selecting them and using the UP/DOWN buttons as in many of such controls in the ABCD-interface.

2.3.5. [dbn].par

For each ISIS-database in a multi-database application such as ABCD, there is a file needed to tell ISIS where to find the constituting parts of the database-files - which then consequently can reside anywhere in the system. Such files are named after the database-name (therefore indicated here as [dbn] with the .par extension. Again this is a simple ASCII-file which can be edited directly or, as is the case here, from this ABCD-menu.



In principle ABCD will take care of this file and make sure that the necessary paths are available. The only special feature here - as compared to the same concept of dbn.par in other ISIS-environments, is the use of 'variables', taken from the operating system's environment variables, which can be dynamically substituted for their actual values. E.g.

%path_database%

is a variable which actually will contain the database-path defined in the config.php main configuration file of ABCD.

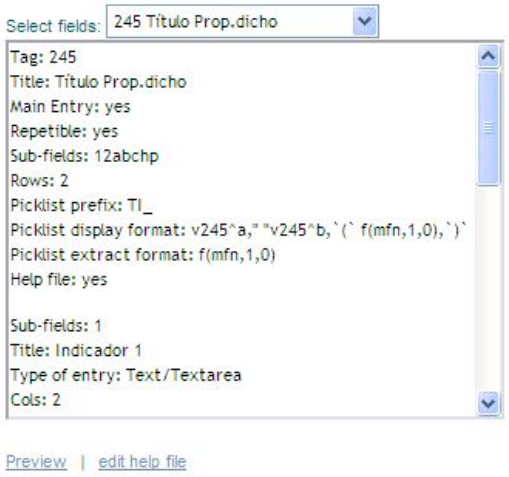
[!!] In the example of the illustration above, the last line is an interesting one as it gives the path to the 'loanobjects'-database for displaying copy-information (if included in one of the MARC-pft's) with the REF-function. In order to find the loanobjects-database for the use of REF->loanobjects, ISIS needs to know the path to this other database and therefore it should be included into the dbn.par.

Note

[!!] For this last feature also to work in the OPAC (iAH), one has to add the path to the first section of the dbn.def file. E.g. the loanobjects-database should be pointed to by a new line there containing : FILE loanobjects.*=%path_database%loanobjects/data/loanobjects.*

2.3.6. Help files on the database-fields

For each field in the database ABCD can provide a help-page, which can be edited from this menu-option. To support the creation of such a page, ABCD will automatically put all known information from the FDT on the given field already available. With the built-in (JavaScript-based) HTML-editor a real nice help-page can then be produced and saved. The 'preview' link of course allows checking the result of your editing effort.

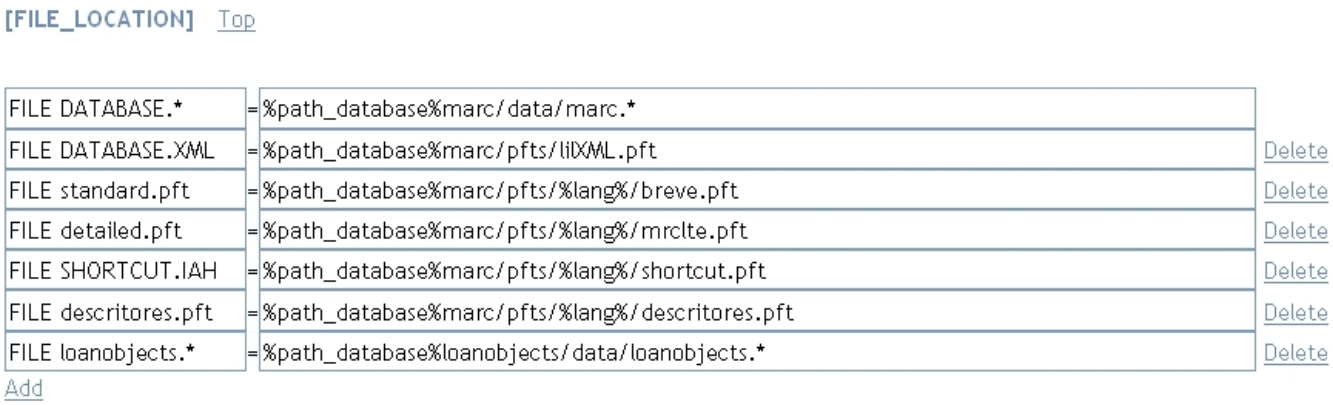


2.3.7. Configure database in iAH (or OPAC)

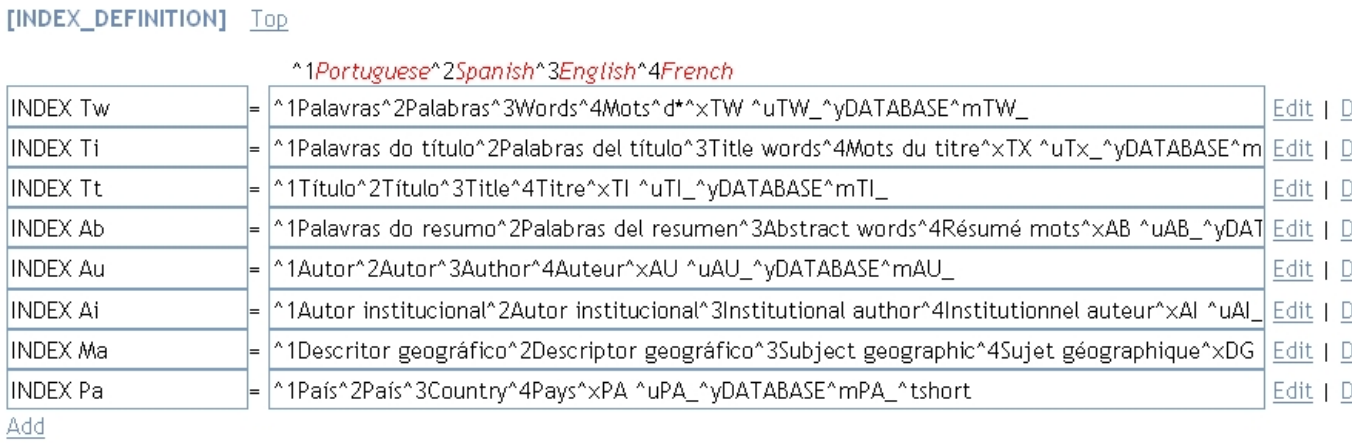
[!!] In order to be able to use a newly defined ABCD-database with the advanced iAH OPAC-interface (or in the ABCD Site), a special configuration file is to be edited : *dbn*.def. (where *dbn* has to be substituted for the actual database-name).

This file has the following sections to be edited :

1. The FILES section : here the paths to the files to be accessed need to be given. In this path both the %path_database% and %lang% variables can be used to refer to resp. the actual database and actual language in use.

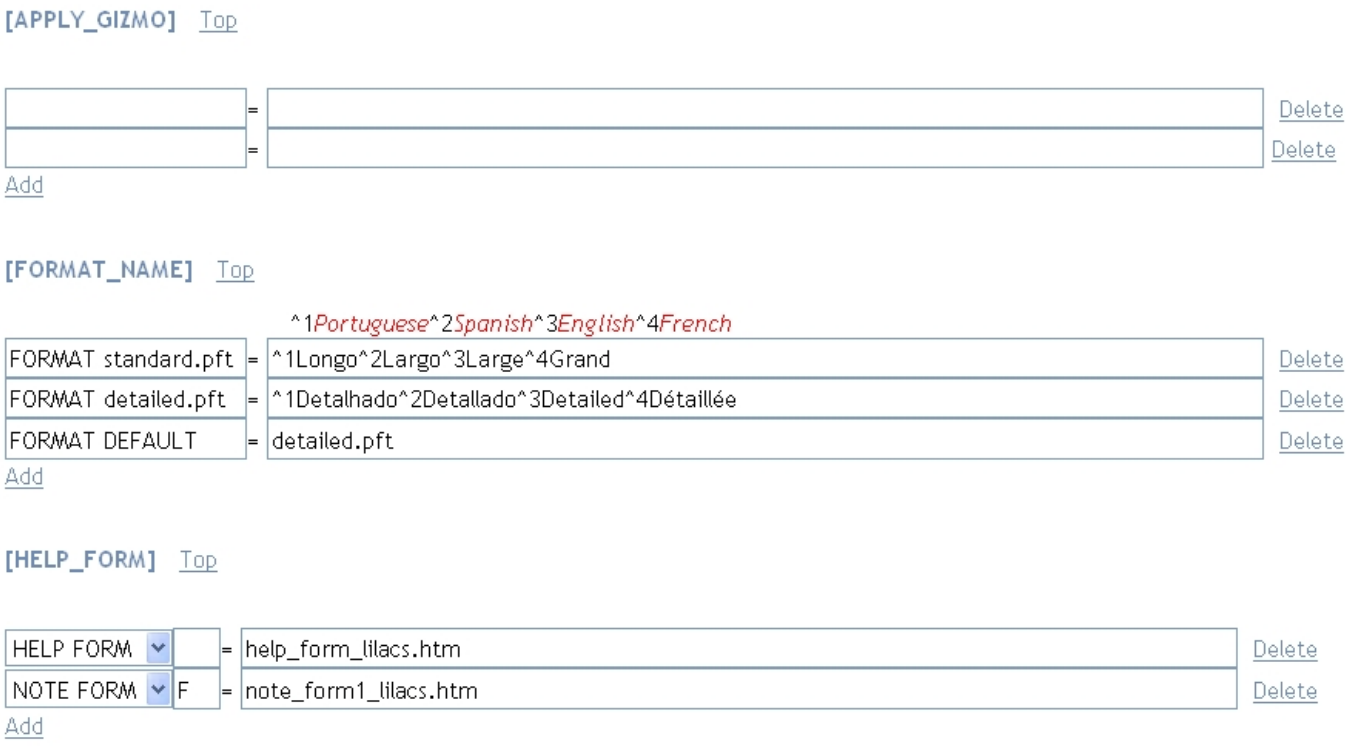


2. The INDEX-DEFINITION section : here all the information for the active languages (numbered as 1, 2, 3 etc.) has to be entered (in subfields) to allow the interface to recognize the prefixes used for each searchable field and to name the field accordingly. The first line with prefix 'TW_' is the one used for the 'simple' search interface (Google-like) for searching words from the fields defined (by the prefix in the FST) to be included in this simple search. This is indicated by the presence of the subfield '^d*'.



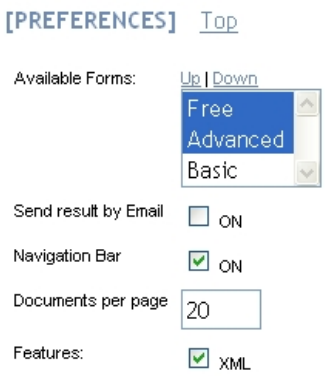
3. The GIZMO section : here - if necessary in specific cases - gizmo databases for automatic substitution of strings by other strings (which can be used to change character sets, but also change codes and abbreviations into full values etc...) can be pointed to.

4. The FORMAT section : here the display formats used in the OPAC should be defined, with for each language (in the numbered subfields) the label to be used on the screen. Remember that only formats (files with .PFT extension) can be used which are referred to somehow in the FILES section ! Also the default format can be identified here by simply indicating which format earlier referred to should be used as default display format.



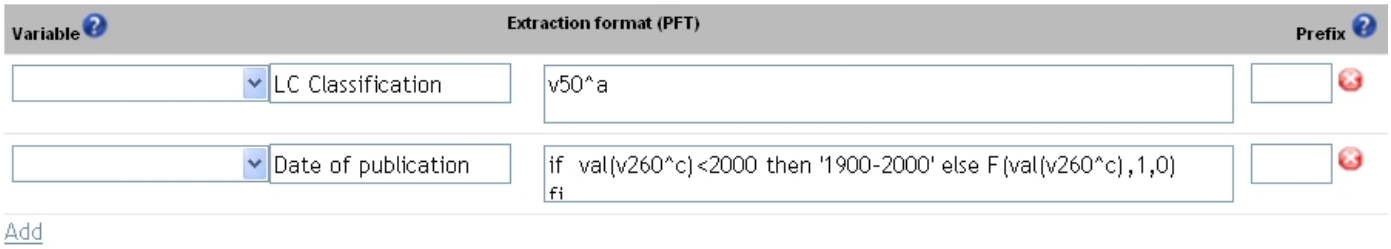
5. The HELP form section : here simply the (HTML-)files containing resp. the help-page and notes-page for the user of this database in the OPAC should be referred to.

6. The PREFERENCES section : here the system manager can indicated which of the three search interfaces (simple, advanced and free) will be available for this database, and some other additional features of the interface : whether sending a search result to an e-mail will be possible, whether the results should be listed with navigation buttons, the number of records to be shown in one page and whether XML-export will be offered.



2.3.8. Statistics : list of variables

This option simply allows quick definition of the variables from the given database with which tables for statistical analysis will be computed. For each criterion or variable (either as row or column for the table) a name and an extraction format has to be given. The extraction format - using the Formatting Language of course - exactly defines how the values in the field should be taken to compute the value in the table. By doing so it is possible e.g. to define ranges of field-values to be combined into one table-criterion. *The file at stake is actually 'stats.cfg' in the (language-specific) def-section of the database-folder.*

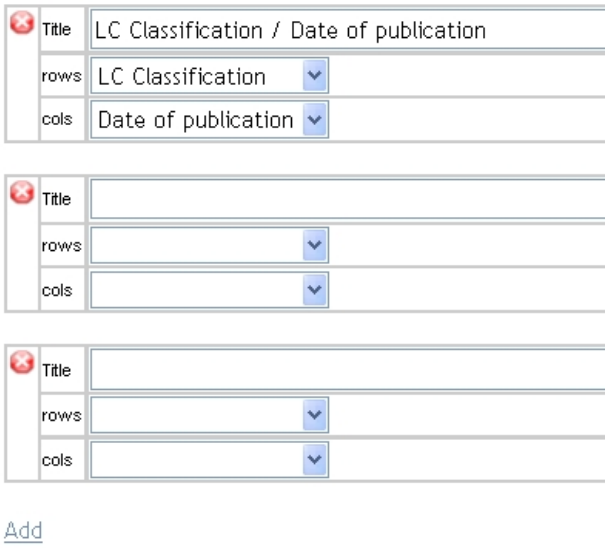


The option to define a prefix is not yet implemented in this version of ABCD. The idea is that the values would be taken from the Inverted File, prefixed with the string defined here. By doing so the values would be computed while 'inverting' the record, not at the stage of producing the statistics table, and therefore allow faster production of the table.

2.3.9. Statistics : list of tables

As with the list of variables above, ABCD also keeps a simple list of available tables, which have been defined previously, for the statistics module. *This file 'tabs.cfg' equally resides in the def-subfolder of the database.* Each line in this file contains three values (separated by the pipe-character) : the name of the table followed by the two criteria used in this table, e.g. :

Classification code / Publication date|Classification LC|Publication date



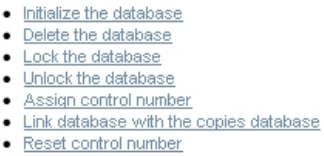
As can be seen from the example the editor in ABCD simplifies editing by providing each of the three values individually but also by providing lists of available row- and column-criteria.

2.4. Reports

In fact creating reports in ABCD means creating ISIS Formatting Language formats (PFT's) with which the reports will create output, because with the F.L. any type of report can be produced and saved for later re-use. We therefore refer to the section on 'Display Format (PFT)' of the 'Update Database Definitions' Central menu option. Exactly the same interface is used here.

2.5. Utilities

In this option ABCD offers some very basic operations on databases :



- Initialize the database means to delete all records in the database but without changing the structures of the database.
- Delete the database of course means fully deleting the whole database with all corresponding files and folders in the ABCD bases/ folder.
- Lock the database means to prevent any other users to make any changes to the records (data-entry), e.g. when a full Inverted File generation would be envisaged.
- Unlock the dabase of course then means to avail the database again to other users.

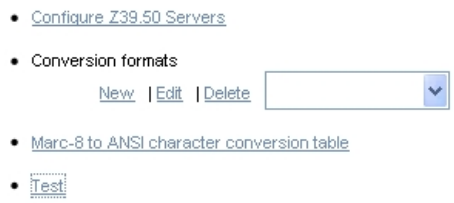
  Use these options with all necessary care and caution !
- Assign Control Number is the option to automatically put sequential control-numbers in a series of selected records. Only a continuous series of MFN's can be selected here.
- Link database with the copies database : as explained on the screen itself, here the option to activate the use of the copies database for the actual (bibliographic) database has to be activated.
- Reset Control Number is the function to manually re-set to a specific value the numerical value in the file 'control_number.cn' of the database, in case for some reason the numbering has to be managed manually. E.g.

setting this number to 1000 will make ABCD to assign from the next record on control-numbers starting from 1000. This could be useful in conditions where e.g. different cataloging centres are using different ABCD-servers but the resulting databases have to be merged into one catalog with control_numbers not interfering, so covering different ranges. By default however reset will revert to 1 as the basic control_number to start from.

2.6. Z39.50 Configuration

Here the ABCD interface allows setting some parameters to define which servers for Z39.50 shared cataloging services will be offered in the Z39.50 list and some more parameters to ensure proper use of the protocol for the server. Such technical information can be mostly obtained from the service provider, e.g. in the case of the Library of Congress consult the following website : http://www.loc.gov/z3950/lcserver.html.

Configuring Z39.50 has the following parts :



- Configure Z39.50 servers : in an interface similar to the one of users-administration, the defined servers are to be configured, with the parameters name, URL, Port, Database and UTF-8 (or not), see the illustration under here for Library of Congress. New servers of course can also be added with the 'Create'-icon.

- Conversion formats : when the incoming records (mostly in MARC21 format) need to be converted to the format used in the destination database, a form can be edited here to specify the conversion from source to destination. The name , subfields and tag of the incoming field will be listed and in the 'Conversion formats' column the ISIS F.L. defines how to extract values for this field in the destination database. ABCD comes with one demo conversion table to convert from MARC to CEPAL formats, with the following example for converting the ISSN MARC field to CEPAL.

- MARC-8 to ANSI character conversion table : this is a table which converts characters from the MARC-8 table (e.g. âa) to ANSI format (e.g. á). This table can be edited here if necessary.
- Finally the Z39.50 can be tested from here, with the same interface as used in the ABCD Data-Entry module (see infra).

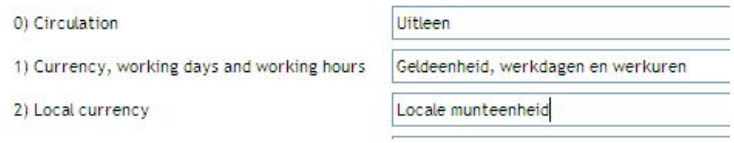
2.7. Translate messages and help pages

There are two types of language-sensitive content which ABCD uses and which needs to be edited when creating or adapting new or other language versions : short messages and labels on the one hand, full help-pages on the other hand. [!!] These can be edited into other languages for each module of ABCD Central.

2.7.1. Translation of short messages and labels

Editing these is facilitated by presenting the default (English) language terms and phrases in a table in wich in the second column the new values should be put by the translator. For each of the main functions such a table is presented :

Here is a sample of some messages translated from English to Dutch for the loans module :

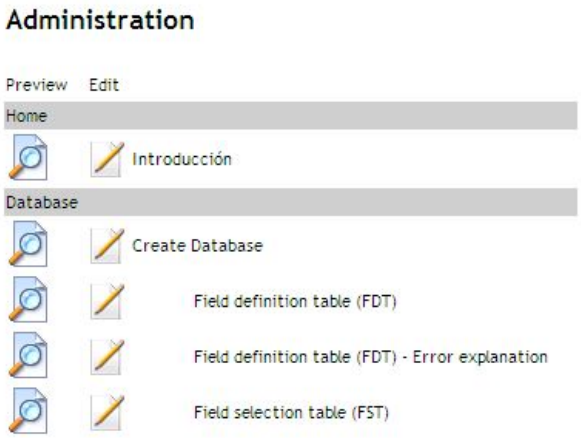


This screen provides a 'save' icon 

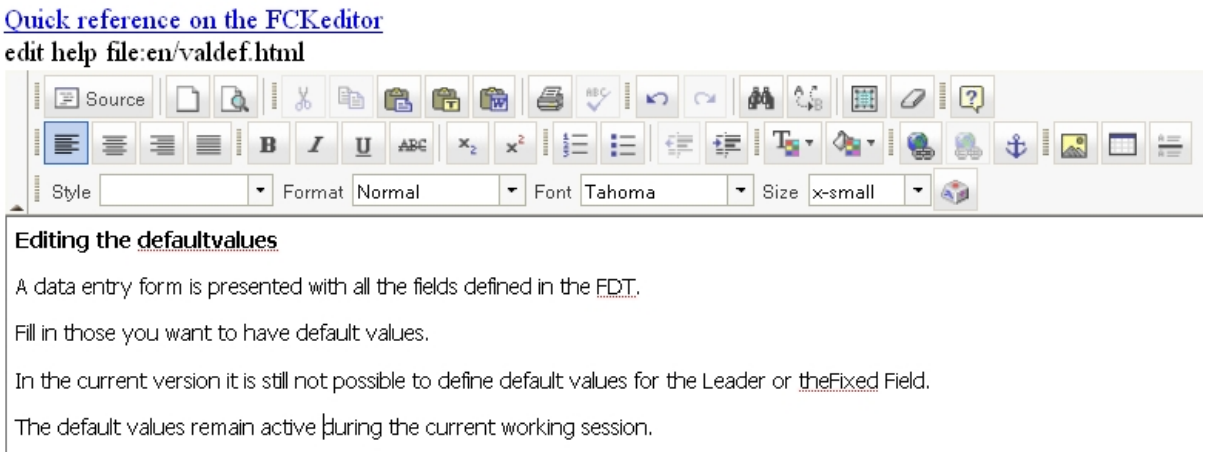
for storing the table with new translations.

2.7.2. Translation of help pages

Here the approach is different : a list of available help-pages is given

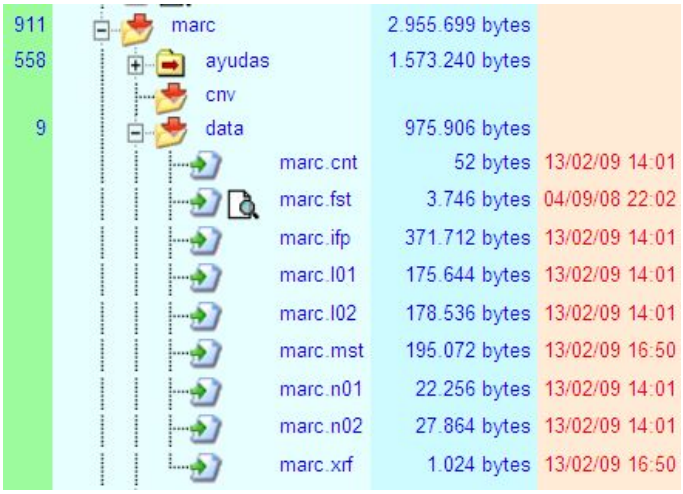


and for each help-file one can 'preview' or 'edit' the page. In the case of editing the built-in JavaScript-based HTML-editor will be provided :



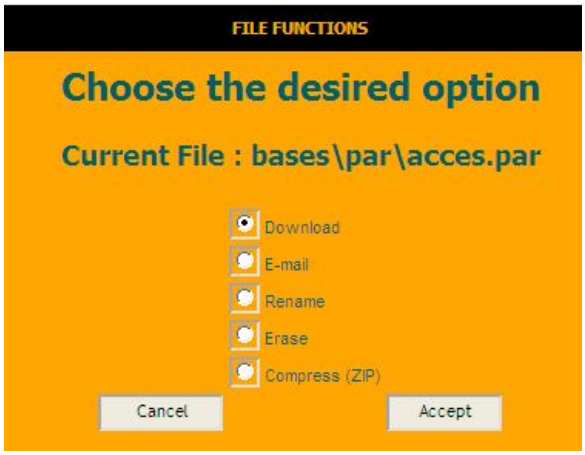
2.8. Explore databases directory

[!!] Exploration of the databases directory can be done (when the dedicated option variable '$dirtree' is put to '1' in CONFIG.PHP !) using a special display in the ABCD-web-page of the folders of the database-folder of the system,



with some possibilities to enter within subfolders and even editing, renaming, zipping etc. some of the text-files in thereon by clicking on the 'details' icon 

, given e.g. the following options to apply on the selected file :



2.9. Statistics

The Statistics module of ABCD also has a dedicated chapter, so here we just refer to this chapter, as this function can also be accessed from this menu but also from the cataloging toolbar and several menu's in the acquisitions and loans modules.

3. Central module : data-entry (cataloging)

In this section we discuss briefly the main techniques of one of the most 'central' functions of ABCD : creating records in a database (in bibliographic applications mostly referred to as 'cataloging').

These frequently used operations for catalogers are grouped into one 'toolbar' on top of the screen - in this way ABCD tries to imitate a 'local desktop software' behavior.

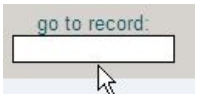
The above shown toolbar is related to the database. Whenever a record is shown, a second smaller toolbar will be shown which contains the buttons related to the current record :

We will discuss all functions of the toolbars here briefly, going from left to right.

3.1. Browsing records

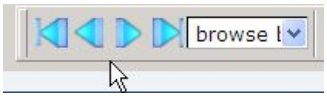
There are 2 ways for browsing records :

- either by entering the record number (MFN) in the dedicated box



  Here you have to simply enter a number from 1 up to the highest available MFN in the database.

- or by clicking on one of the 'navigation buttons' to go the the first, previous, next, or last records



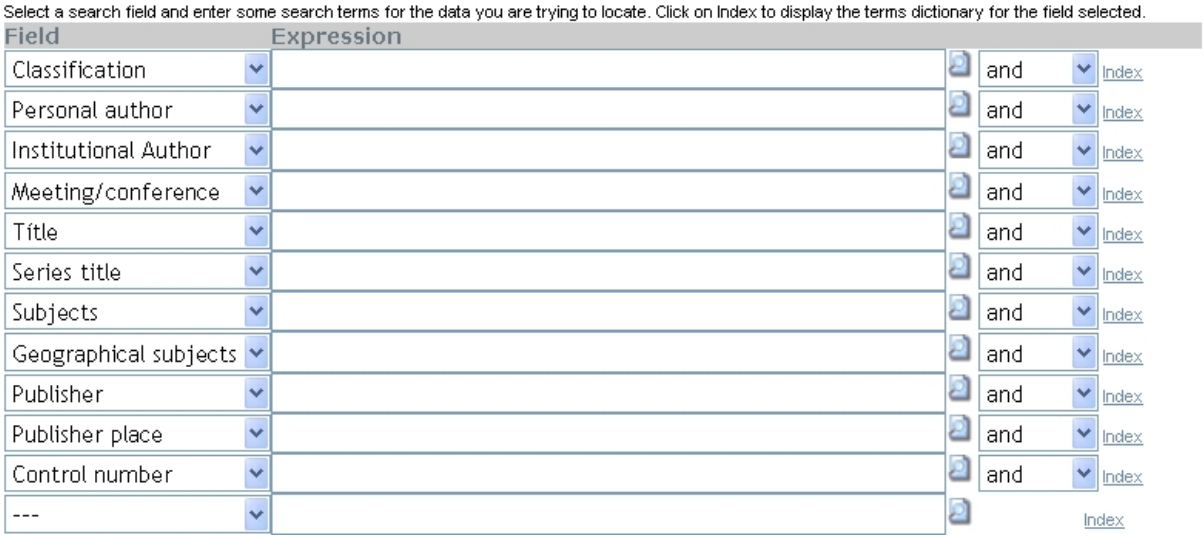
  This navigation will be done either in the full database or in the search result set, according to the selected option in the adjacent menu.

3.2. Searching records

As with browsing, ABCD offers 2 main approaches to identify a specific records for catalogers : by either searching or by selecting records through an alphabetic listing (AtoZ)

-

  by performing a search with a powerful search-function built-in into the database administration module, resulting in a real search-form presented (as defined in the 'define search form' function in the main administration menu) :



  Searching this form is done as explained in the 'advanced search interface' section of the OPAC, which indeed uses the same form.

- the AtoZ selection tool

Clicking on this button will display another, small window in which all records of the database are listed according to the field identified as the 'Identifier' field in the database definition table (3rd column 'I'), mostly the title field e.g. in bibliographic databases. In this list each alphabetic section can be clicked on, and after clicking on the relevant line, the record referenced to by this line will be automatically presented in the main window, ready for e.g. editing. This nice tool extends, as we could put it, ABCD all the way up to Z !

3.3. Using the editing forms

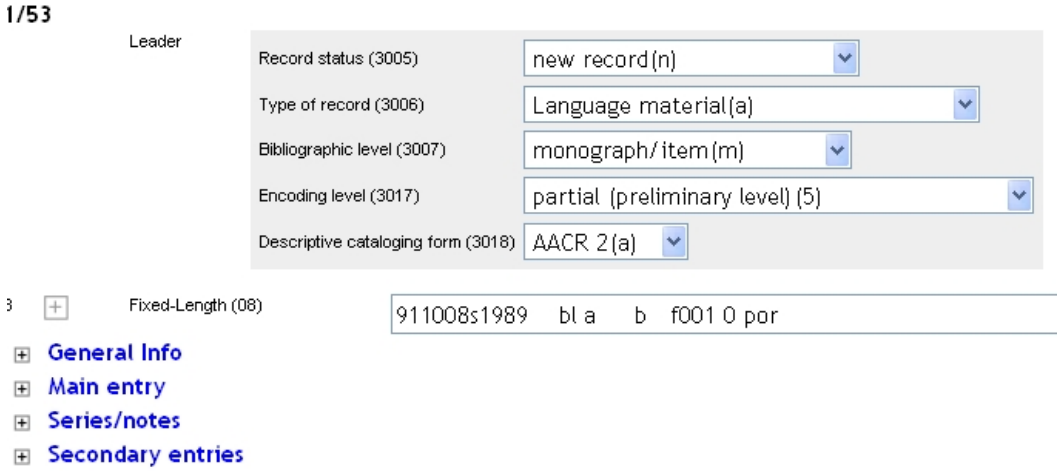
[!!] When clicking on the first icon of the record-toolbar, the record will be presented in an editing form. The same form - but empty - will be used for creation or copy of a new record (from the database toolbar).

Note

In the case of MARC21 and CEPAL databases, whenever editing a new record, first a selection has to be made from a list of possible record types :

Selecting one of these (i.e. clicking on their link) will subsequently invoke the right worksheet with some pre-defined values (e.g. for the 'fixed format' MARC fields).

[!!] The MARC-21 edit form, with its long and complicated structure, will be presented in a special way : form sections can be 'collapsed' or 'unfolded' by clicking on the +-link of the according section. The details of the section will be shown or hidden. ABCD implements only MARC21 at the 'minimal' and 'national' level, by doing so it provides suitable levels for both larger and smaller libraries.

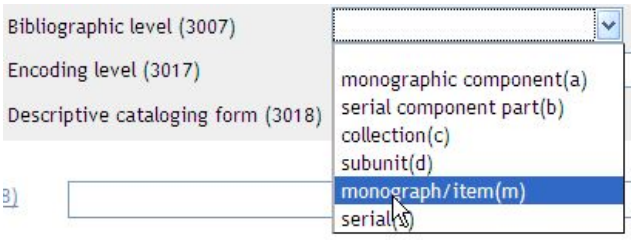


General remark : depending on how the Field Definition Table of ABCD, which in contrast to the one of WinISIS also defines worksheet features, was designed, the worksheet can appear as divided into sections with direct-access links (buttons) and buttons to return to the beginning of the workshe et for faster navigation *within* the worksheet.

Fields for editing can have one of the following formats, each with their own way of dealing with them, briefly explained below as follows :

- a simple editable field, [!!] as in the ISBN- field below, where you can only type in a value (e.g. ' ^a8570251270') yourself

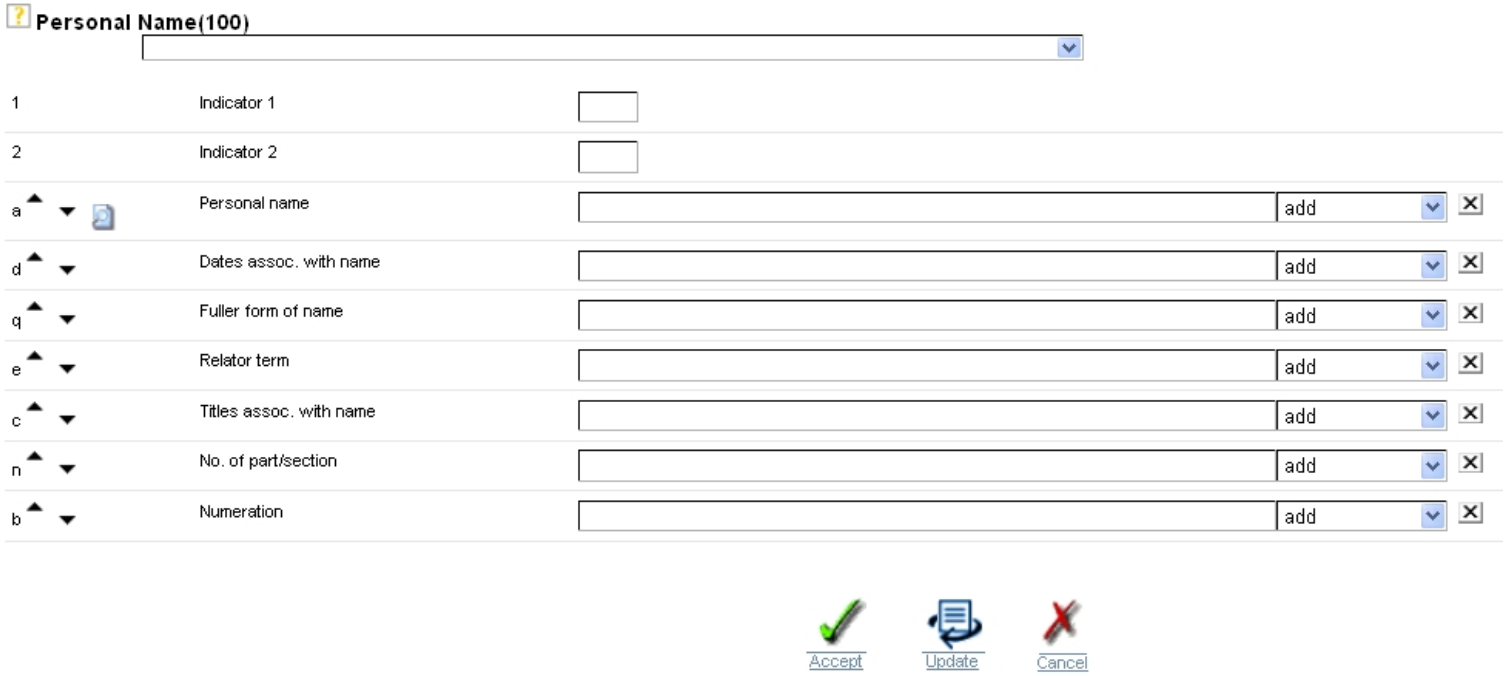
- a field with a menu from which a single option has to be selected, as e.g. 'monograph' as the bibliographic level :



- a substructured field, with a button to give access to a separate window in which all substructures (e.g. subfields or parts of the MARC-fixed header field) can individually entered.

Clicking on the icon

will invoke a new window in which all substructures are presented individually, e.g. in the for MARC field 'Edition' (250) there are subfields for the edition 'info proper and one for the 'additional information' :



In this window each substructure value can be entered into its own edit box, repeats can be added with the 'add' option and the whole 'group' of substructures can be repeated by pressing the button, which will add one series of substructures as a new occurrence of the same field.

[!!] The 'ADD' option at the right end of each subfield allows to create another occurrence of the same subfield. Beware : if you need to create a new occurrence of the field rather than the subfield, use the '+'-sign at the left top as mentioned above.

By clicking on the button Accept the values will be placed into the field box with the proper substructure delimiters added.

By clicking on the button Update the edited values will be entered in the field box of the main editing window. Needless to say that the 'Cancel' button will just close the 'substructure' window *without* adding any values into the field.

It is possible to change the sequence of the repeats of substructures by using the 'up' and 'down' buttons to move up resp. down the actually selected occurrence

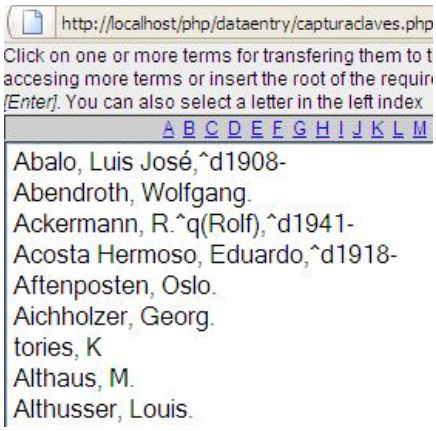
Finally one can also delete one occurrence of a substructured field by clicking on the button

Experienced data entry experts will often by-pass this additional support from the ABCD-interface and type in the substructures of the field with their appropriate delimitors directly into the main worksheet editor, which is perfectly possible : the extra window is just an extra support indeed !

- a field with a picklist, indicated by the icon :

  In a field with 'authority control' by predefined terms listed in a separate box, clicking on this icon will open the list in a smaller window :

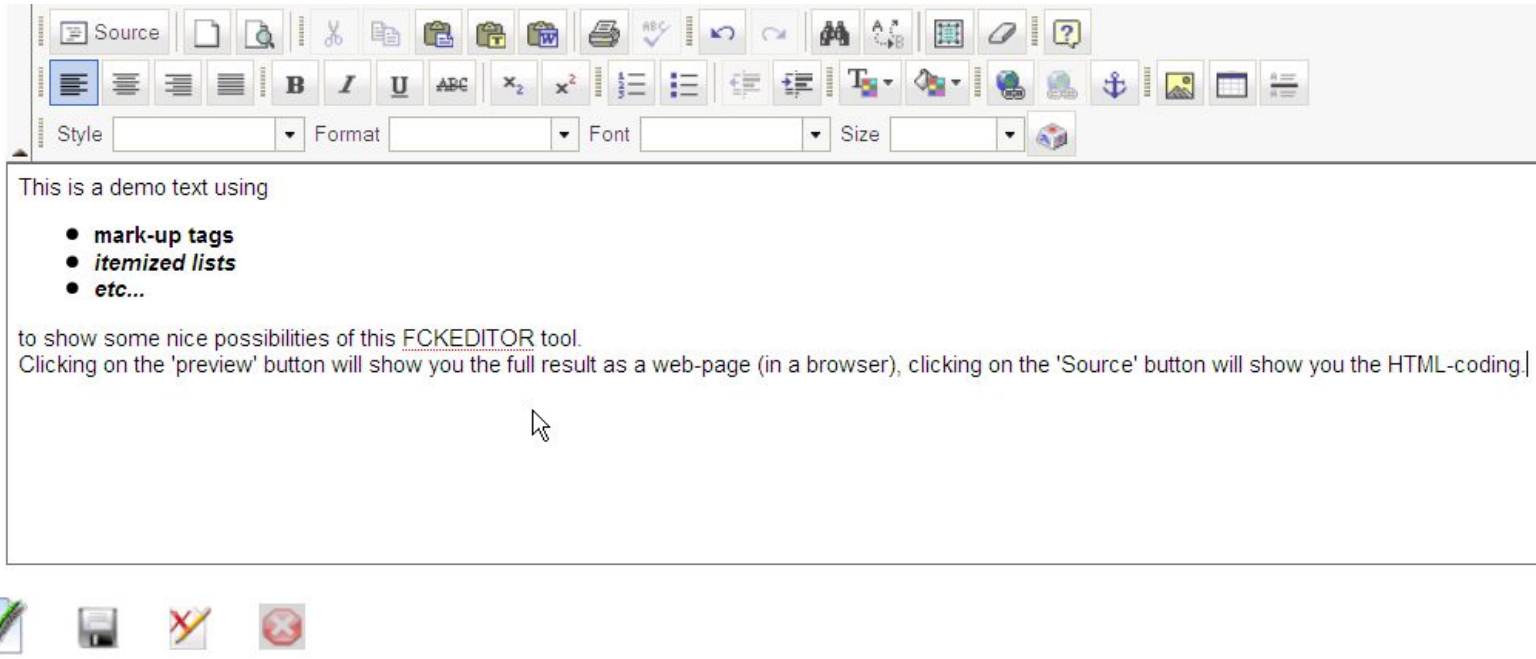
  Depending on the definition of this picklist (see the dedicated paragraph in the section on picklist definition of the database definition functions), only one or more items can be selected. Needless to say that the items listed will be precisely defined as well in that Field Definition Table set of colums on picklists, e.g. by defining a common prefix with which the terms have been indexed.



  In this list you can navigate by using the alphabetic control or select one (or more) items from the list by clicking, Shift-clicking or Ctrl-clicking (or dragging the mouse) for multiple selection.

  After you are ready with selecting one or more terms, click on 'Continue' (below within the picklist window) to transfer your selection to the main edit form.

- a very special, interesting field is a 'Rich Text Editor' field, which shows up as follows :



As can be seen, the field is a larger editing box with a series of icons above as in a real text editor with many text-editing features, such as itemized lists, boldface or italics etc. Such a field in ABCD is edited with a special add-on JavaScript tool, i.e. 'fckeditor', which allows to use the icons in order to create the according HTML-tags in the text of the field value. Whenever this value is shown in a WWW-environment (as ABCD itself, or e.g. in J-ISIS), the HTML-tags will be interpreted as such and result in graphical effects, as this is the meaning of HTML-tags.

It is also possible to 'paste' into such a field text obtained from another document (e.g. a Word-document), using a conversion mechanism (to filter out all non-text elements) provided by ABCD.

3.4. Record and field validation

[!!] ABCD allows the system manager to design and implement validation statements for quality control at data entry stages. Such statements are written in the ISIS Formatting Language - we would say 'of course'..! See the section on this topic above - as it is an option of the main Central 'Update Database Definitions' menu. In the record-toolbar this function can be executed to actually check the current record according to the validation defined. The resulting messages will be shown in a small separate window.

3.5. Shared cataloging through Z39.50

Z39.50 is a protocol which allows 'sharing' records (especially in the MARC-format) by first identifying them through a search with one of a series of Z39.50 servers - like Library of Congress in Washington or British Library in London - and then downloading the records into the local system. Here is how it works in ABCD :

3.5.1. Configuring the Z39.50 of ABCD

Before you can use this function, some configuration has to be set in order to define the Z39.50 hosts you can or want to use and how to reach them. ABCD provides a menu option (main Central menu) for this configuration.

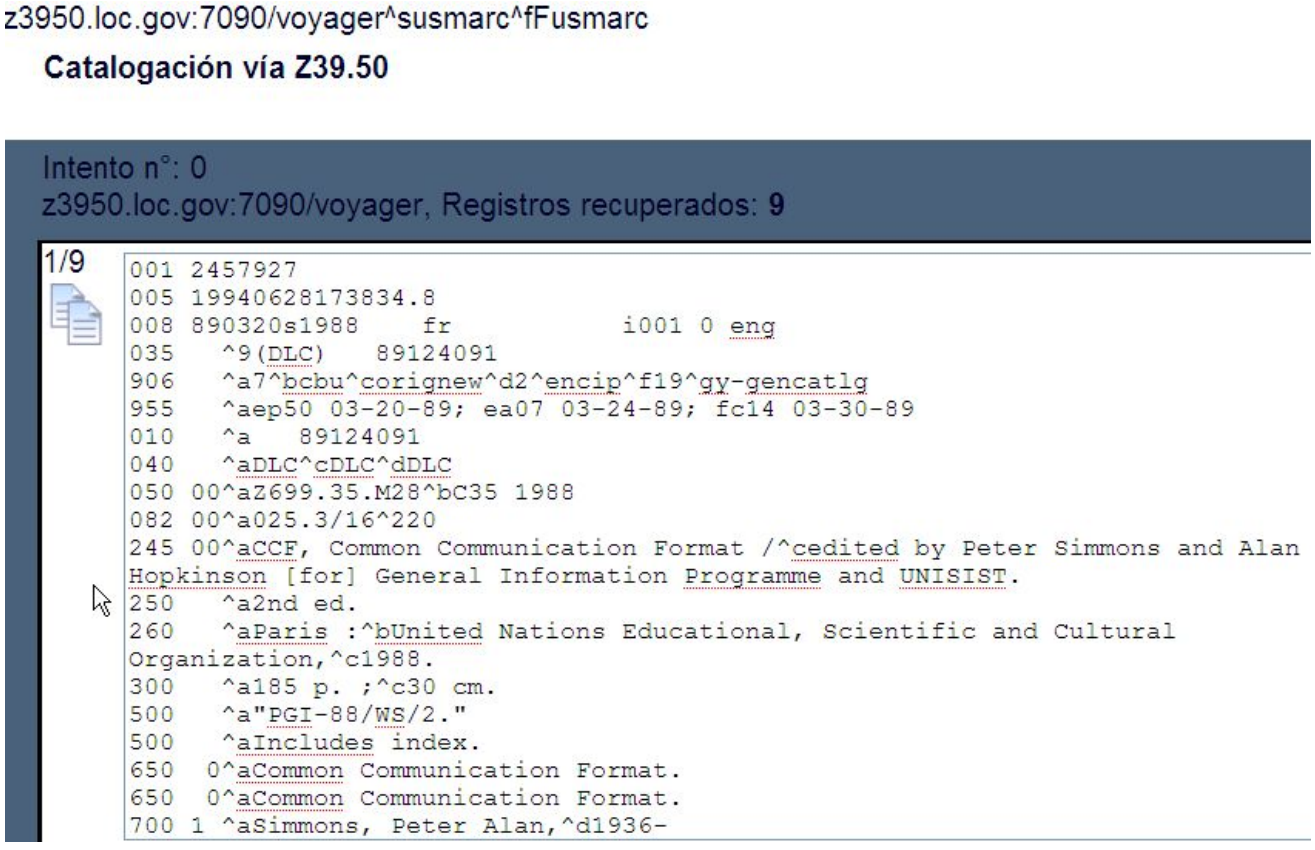
3.5.2. Using the Z39.50 tool of ABCD

The following steps have to be taken for actually using the tool :

1. Click on the Z39.50 icon  in the cataloging toolbar, invoking a screen allowing the selection of a Z39.50 host and either a search statement or one or more ISBN's to be sent as to identify records for downloading :

2. After having clicked on the button 'Buscar' (or search), if the submitted request is successful (meaning : the server accepted your request and identified one or more records indeed, complying with your search criteria),

   a new browser window will present the list of results with a 'copy to database' icon  for each individual record :



3. When clicking on the 'copy-to-database' icon, the selected record will be transferred into your ABCD cataloging screen as the actual record (a new MFN will have been granted), where it can be edited with the edit button if necessary.

   Needless to say that in most cases such records are high quality MARC-records, so this shared cataloging has many advantages, e.g. saving time (if sufficient bandwidth for the communication with the server is available)

and improving quality of your catalogue. If your catalogue does not use the MARC format however you might need to first prepare a conversion mechanism before loading records into your own database. This technique is discussed elsewhere in this document.

4. [!!] When the Z39.50 icon is clicked not from the database-toolbar but from the record-toolbar, the behaviour will be slightly different : select the actual record (preferably by its ISBN) and when available from the selected Z39.50 server, only the missing fields in the actual record will be added, in order to complete the bibliographic record.

3.6. Default values

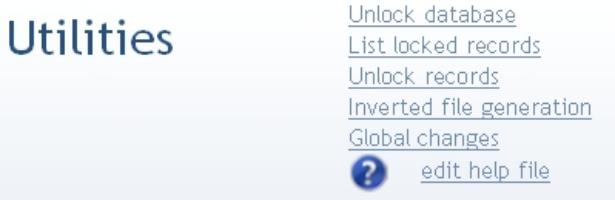
This is a simple function of the database-toolbar : the edit-form will be presented and if so desired default values can be entered (or deleted) into this form.

3.7. Reports (printing)

As explained earlier, ABCD reports are generated by ISIS Print Formats and therefore the interface for the creation of PFT's is re-used here. In the future ABCD will probably offer some pre-defined frequently-used or typically PFT's, e.g. for listing of overdue loans etc.

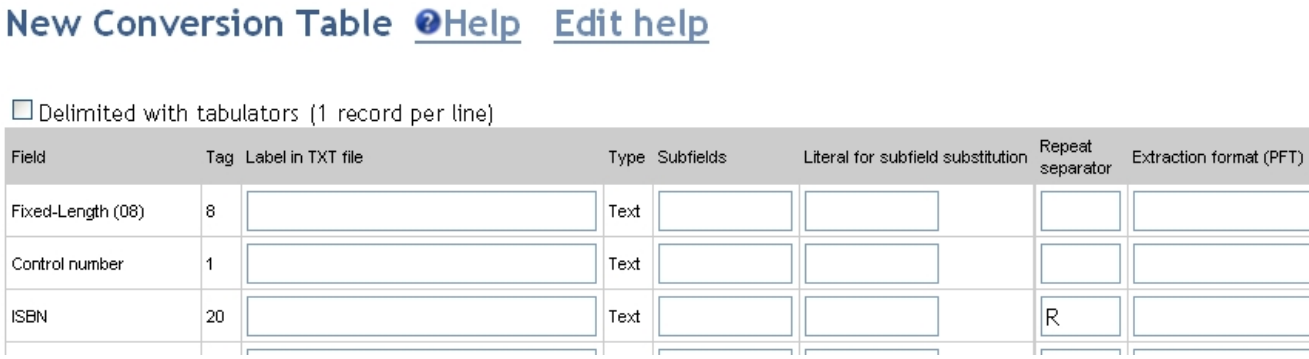
3.8. Utilities

The utilities offered in this Data-Entry module from the database-toolbar are the following :



1. Import

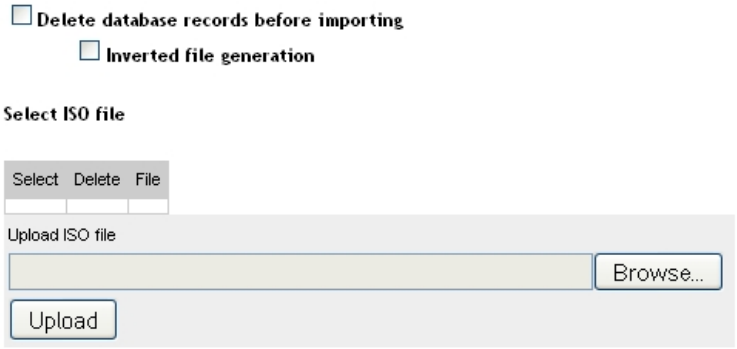
   a. Text Import : text-based structured files can be imported by defining a conversion table, which will take the label for each (sub-)field (or the literals used to separate subfields) in the input-file, the 'repeat-separator' used within one field for different occurrences and the extraction format in ISIS-Formatting Language. :



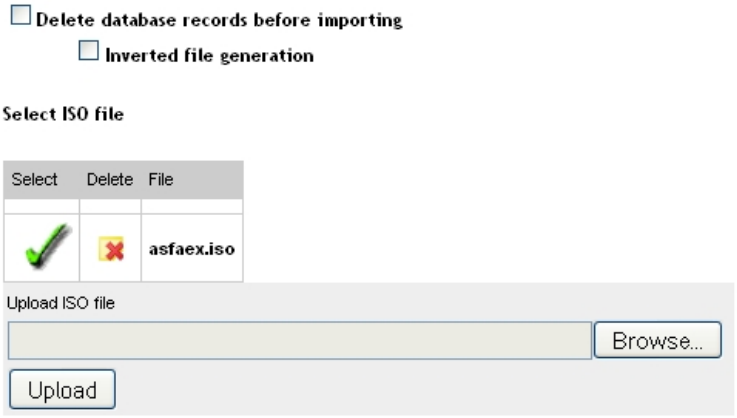
   If the text-input file is actually a 'TAB-delimited' file (a 'comma-separated values' file with TAB's in stead of comma's), activate the button and ABCD will simply - without a need to identify each field or column - import each value into a separated sequential field (meaning : the first value or column will go into field 001 etc.).

   b. ISO Import

   ISO-files can be imported into an ABCD-database in 2 steps : first the ISO-file has to be uploaded by browsing to it :



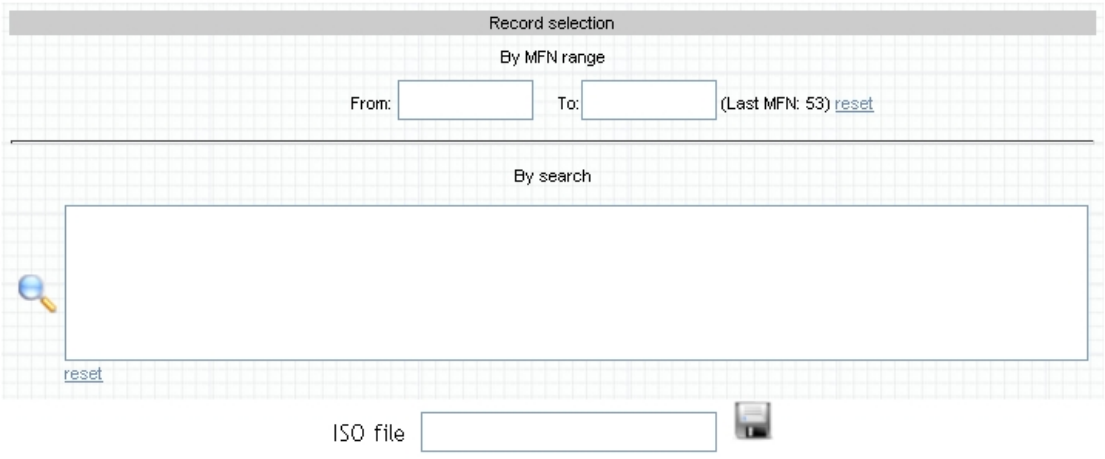
Then the uploaded ISO file will be listed and can be actually imported with the green 'activate' button, after which ABCD will actually import the ISO-records into the actual database (using the CISIS tool for this purpose), meaning that in case of high volumes the Apache webserver might issue a time-out error - then use the command-line tool for this ! :



2. Export

   a. Export to ISO

      This function simply asks for a selection of records (by MFN-range or search-expression) to export and a name of the export ISO-file :

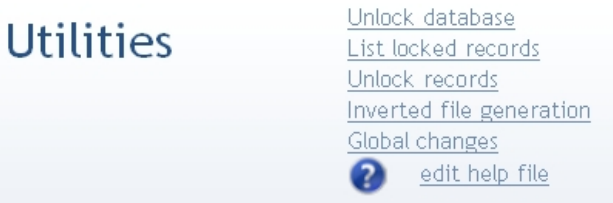


   b. Export to TXT

This is the reverse function from the TXT-import, using the same conversion table format, but logically doing the reverse operation on fields. By activating the 'delimited by tabs' option the export will be actually a CSV file which can be imported into spreadsheets etc.

3. Other database utilities

In addition to importing and exporting databases from and to different formats, ABCD also needs to provide some tools for the database-administrator, which we will briefly discuss here :



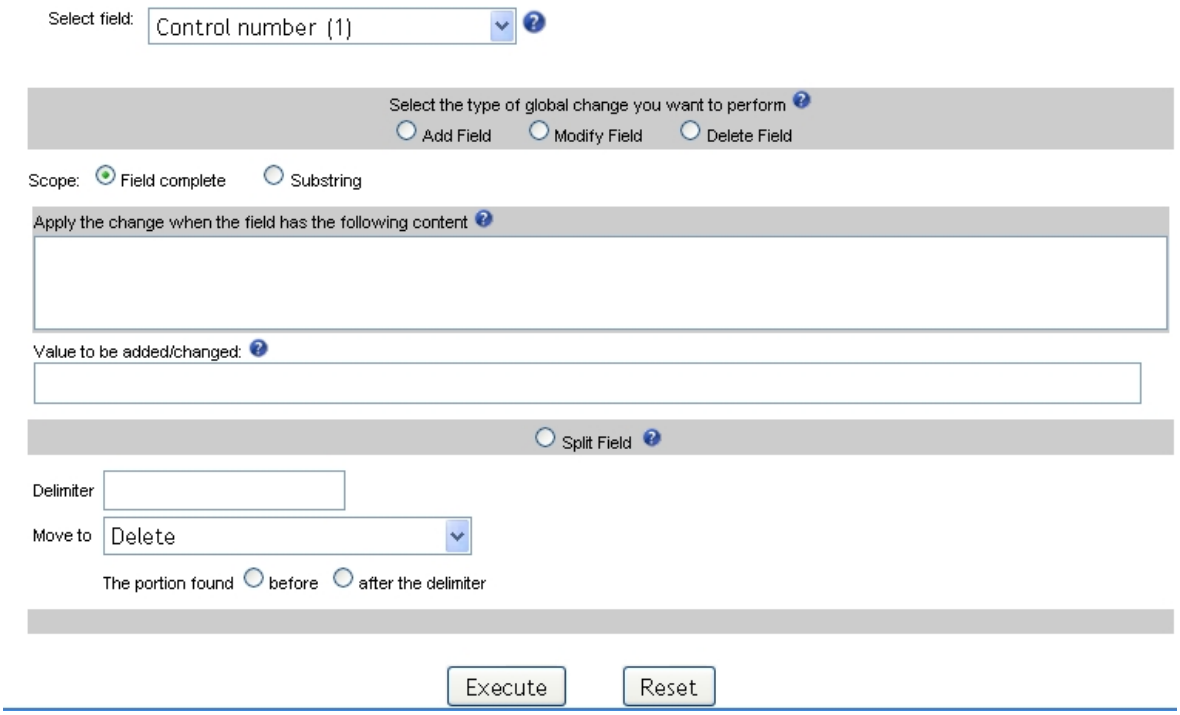
- unlock the database : in case a database got locked (as part of the multi-user protection) without having the chance of being unlocked, e.g. in the case of a sudden power-cut, the database can be manually unlocked. ABCD will simply report on the status after unlocking.

- List locked records : ABCD will list the records which are locked (if any) with their status :

Mfn	Locked by
13	LOCKED^iabcd^t200911111700^x1257955216
39	LOCKED^iabcd^t200911131738^x1258130331

- Inverted File generation : in case structural changes have been made to the FST, the system administrator should re-index the database to apply the new FST-definition onto all the records of the database (not only on the newly edited ones, which will be done automatically). If the set of records is not too large - to be seen in function also of the complexity of the FST and the average size of the records - the full Inverted File generation can be done from within ABCD here. In case of larger databases this should be done by a command-line operation with one of the CISIS tools, e.g. 'mx MARC fst=@marc.fst fullinv=marc now -all tell=1000'.

- Global changes : as in WinISIS (and even provided in the old days of CDS/ISIS for DOS with additional ISIS/Pascal programs) changes can be edited semi-automatically over the full database or a range of records (identified by range or search-expression). The following interface illustrates the possibilities here :



- select field : which field to operate on
- type of global change : either the 'new' value will be added as a new occurrence of the field, the existing field will be modified within or deleted.
- scope of the field : the complete field has to be matched or any substring within the field
- content to be located : the string of or within the field to be changed
- value to be added/changed : the new string which will overwrite the located string
- split field : in case a delimiter - as given here - is met in the field, the section before or after the delimiter (as checked) can either be deleted or moved into another field (to be selected from the list).
- finally the operation can be executed or the form can be reset.

  ABCD will show the result of the operation. Extreme caution should be applied with these operations as they can be irreversible and affect many records. Taking a backup (export to ISO as explained above) is a good idea for sure.

- The dedicated HELP-file for these utilities can be edited from this option. The existing text will be given in the HTML-editor and can be edited and/or translated.

3.9. Statistics

This remaining database-toolbar option (since we don't have to discuss the help- and home-options as they are quite self-explanatory) is the one on Statistics. This option is explained in its own section but can be accessed also from the cataloger's toolbar.

3.10. Barcodes

1. Introduction

   ABCD (as from v1.5) has a function to assist in the production of barcodes. Earlier on ABCD-administrators had to rely on external - but existing and freely available - tools to produce their barcodes, but now ABCD has its own build-in option.

Only databases which have been configured to use barcodes (as this is not a mandatory option in ABCD) will show the barcode icon  in the toolbar. To activate this option for a database add the following line into the file 'dr_path.def' of the database concerned :

barcode=Y

To access this option the operator must either

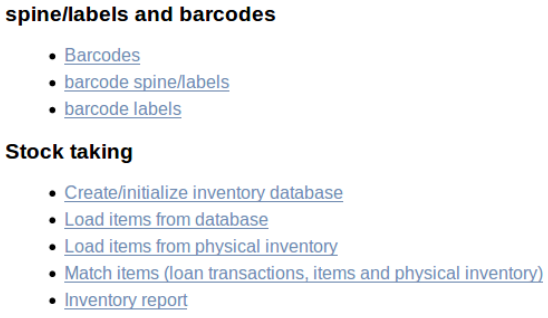
- be System Administrator OR
- have all the cataloging permissions on the database OR
- have permission to print labels and bar codes (in the permissions settings of the related database).

This function will help to print labels that identify the physical objects in the circulation system or other databases with barcodes. A configuration file needs to be edited, where - among other variables - the dimensions of the labels or barcodes are defined. The results can be sent to the screen, to a txt file or to a word processor. At this moment barcodes can only be applied when the inventory information is stored inside the bibliographic records (the catalog-database, e.g. MARC).

Labels can be produced by

- Classification numbers : a range of record classification numbers is provided and a report with barcodes is produced for all inventory numbers stored in the record
- a range of inventory numbers : instead of classification numbers a range of inventory numbers needs to be indicated; insert the inventory numbers for which you want to print barcodes in the corresponding box and separate them by comma's (,)
- Mfn range: a range of MFNs is provided and the report with barcodes is produced for all inventory numbers contained within each record of the requested range

The option 'barcodes' with the above mentioned barcode-icon when clicked will show up as follows, together with the option 'stock-taking' (which will be discussed in the next session), with the three main formats, resp. 'barcodes, 'spine-labels' and '(normal) labels' :



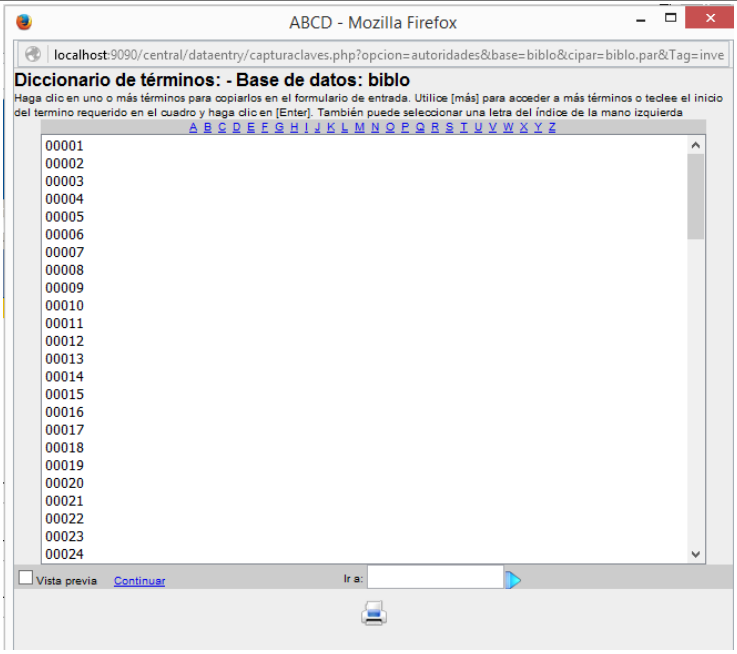
2. Configuration

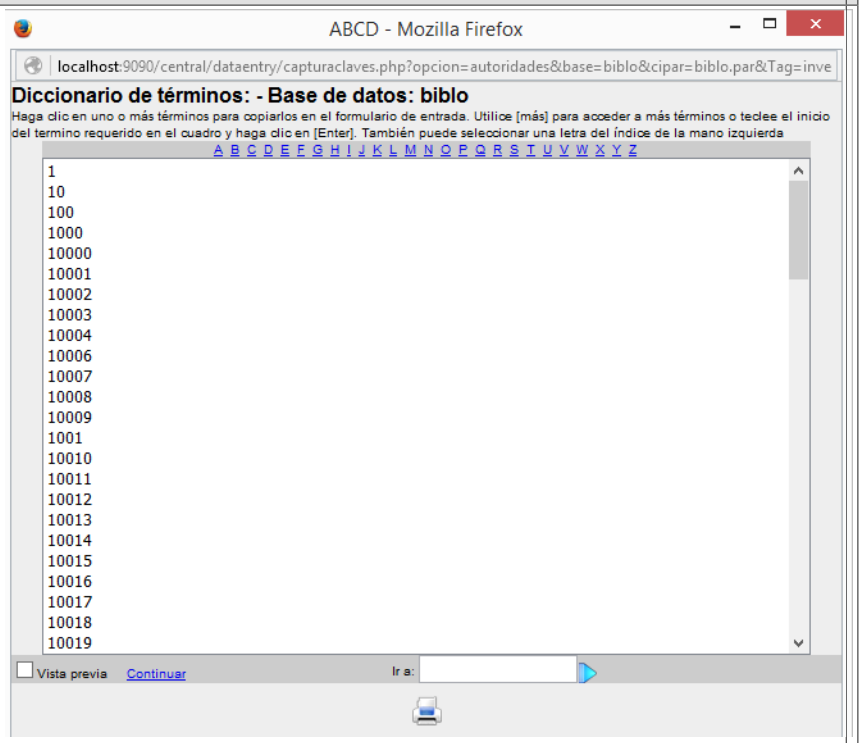
This section defines the parameters that help to extract information from the database, prepare the presentation of the label and inform its dimensions. The following information is requested:

Table 2.1. Barcode parameters

Parameter	Explanation
FST classification number prefix	Preliteral used to identify the classification number in the FST. Example: ST_

Parameter	Explanation
Format to locate the classification number	Format that will be applied to the database to extract the classification number. It is preceded by the previous prefix to present the list of classification numbers in the process of selecting the records. Example: if p (v82 ^ a) then v82 ^ a, "" v82 ^ b "." V82 ^ c, "" v82 ^ d, "" v82 ^ e, else v82 ^ b, "." V82
Prefix of the inventory number in the FST (edited)	Preliteral used to identify the inventory numbers in the FST and present them properly organized for the rank search, for example, to add zeros to the left even if it has not been entered that way. Example: NICLA_ **Note** It is recommended to add this field to the record indexing table (FST). Example: (if p (v900 ^ n) then 'NICLA_', If f (val (v900 ^ n), 1,0) = v900 ^ n then replace (f (val (v900 ^ n), 5.0), ``, `0`), else v900 ^ n, Eur-lex.europa.eu eur-lex.europa.eu Fi) The script in this format: • Determines if the field is numeric (if f (val (v900 ^ n), 1,0) = v900 ^ n), which verifies whether the value of the field expressed in numerical form is equal to the value of the field; • If so, the field is converted to a number in a fixed format of 5 characters by filling in spaces on the left which are then replaced by zeros: replace (f (val (v900 ^ n), 5.0), ``, 0 '); • If it is not numerical, no change is made to the field. **Note** The value 5 shown in the format must be adapted to the number of positions currently held by the major Barcode entered in the database.
Format to locate the inventory number (edited)	It will be applied to the database to locate the inventory number and present it in the record selection window in order to issue the labels by a range or by a list of inventory numbers. In this example, zeros have been added to the left of the inventory number to ensure that the sequence presented and subsequently retrieved is correct.Example: if f (val (v900 ^ n), 1,0) = v900 ^ N then replace (f (val (v900 ^ n), 5.0), ``, `0`), else v900 ^n fi This script above : • Determines if the field is numeric (if f (val (v900 ^ n), 1,0) = v900 ^ n), which verifies whether the value of the field expressed in numerical form is equal to the value of the field; • If so, the field is converted to a number in a fixed format of 5 characters by filling in spaces on the left which are then replaced by zeros: replace (f (val (v900 ^ n), 5.0), ``, 0 '); • If it is not numeric, no changes are made to the field. Here is the list presented for inventory number ranges when zeros are added to the left:

Parameter	Explanation
	 Remember that you must create in the FST the NICLA_ recovery key with the inventory numbers with leading zeros: 900 0 (if p (v900 ^ n) then 'NICLA _', if f (val (v900 ^ n), 1, 0) = v900 ^ n then replace (f (val (v900 ^ n), 5.0), ``, `0`), else v900 ^ n fi, '%' / fi) See the difference if this method of left-filling with zeros is not used : a. The normal indexing in the FST would be 900 0 (| NI_ | v900 ^ n |% | /) b. The prefix "NI_" would be used to retrieve the inventory numbers c. The extraction format of the inventory number would be v900 ^ n d. The list of inventory numbers would be displayed as follows, but making it impossible to establish a continuous range of inventory numbers.

Parameter	Explanation
	
Prefix of the inventory number in the FST (unedited)	Preliteral used to identify the inventory numbers in the FST respecting the way they were entered. Used when printing is requested through a list of inventory numbers since it is necessary to compare the list supplied against the value entered in each of the records. `Example: NI_`
Inventory number format (unedited)	This format will be applied to the database to locate the inventory number as it was entered into the database. It is used to determine if the inventory number of the repeatable field of stock-items corresponds to the range or the requested list. Example: v900 ^ n
Print Format (PFT)	Format to use to print the selected label or label. Example of barcode label: `Proc ('a1000 ~' if p (v84 [1]) then v84 [1] else v82 [1] fi '~'), (If p (v900 ^ n) then, 'INV *', </ Span> </ span> </ span> <span style = "font-size: V1000 ^ a [1], v1000 ^ b [1], v1000 ^ c [1], v1000 ^ d [1] '%%%' / Fi)` **Note** to understand the example PFT above : a. the use of ` to correctly encode the PFT b. the inclusion of the selected font to generate the bar code: c. the inclusion of the literal * INV * before and after the inventory number (v900 ^ n). This is essential because if you request the printing of a range or a list of inventory numbers ABCD you must have access to this value to compare the information extracted from the registry against the requested values to determine if an occurrence of the stock-items will be or will not be included in the requested output. d. line breaks were added to provide better visibility to the format. The configuration file should not have line breaks because one parameter is interpreted per line.

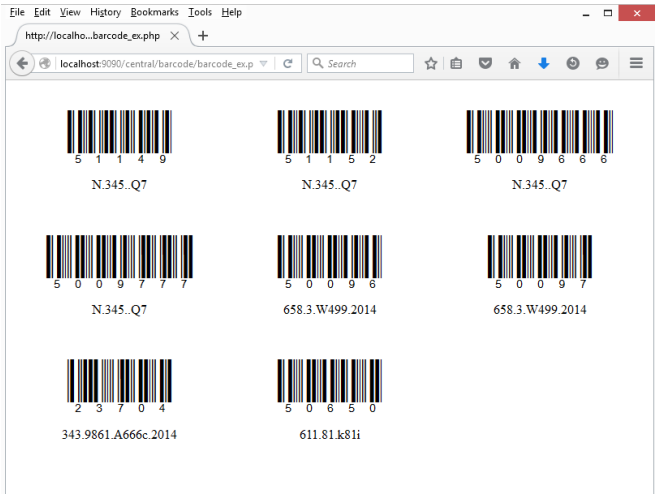
Parameter	Explanation
	If desired the format can be supplied as an external pft using @ format_name .pft

Print Format (PFT) to send to TXT	Format to apply when the result wants to be sent to a TXT file in order to be processed by a special printer. You can enter directly in the text box or provide the name of an existing format or to be created in the same process. Use the edit-icon to edit or create the corresponding format.
	
	Example of the format to be used to generate a TXT file for printing barcodes on a printer of type Zebra:
	
	`Proc ('a1000 ~'`
	`if p (v84 [1]) then v84 [1] else v82 [1] fi '~'),`
	`(If p (v900 ^ n) then,`
	`'INV *' , (2-methoxyphenyl) -2-methyl-2-oxo-1,2,3,4-tetrahydro- '^ XA ^ LL0203',`
	`'PW831', /`
	`1, 1 H-NMR (DMSO-d 6):? 1, 1 H-NMR (DMSO-d 6):? 1, 1 H-NMR (DMSO-d 6):?`
	`1, 1 H-NMR (DMSO-d 6):? 1, FT97,165 A0N, 25.43% FH + FD + FT97.134% A0N, 25.43% F`
	`If a (v900 ^ m) and a (v900 ^ e) then | | V900 ^ l, else`
	`If a (v900 ^ m) and p (v900 ^ e) then | T. | V900 ^ e, | Ex. | V900 ^ l else`
	`If p (v900 ^ m) then | Vol. | V900 ^ m, | T. & lt; / RTI & gt; Ex. Fi, Fi, Fi,`
	`'^ FS', /`
	`1, FT97,166 A0N, 25.43% FH + FD + FT97.165% A0N, 25.43% FH + FD +`
	`If p (v1000 ^ f) then v1000 ^ f fi '^ FS', /`
	`1, 1 H-NMR (DMSO-d 6):? 1 H-NMR (300 MHz, CDCl3):?`
	`(1) 1, FT498.36 (M + H), 22.28 (M + H) + = 1, 1 H-NMR (DMSO-d 6):? 'PQ1,0,1, Y ^`
	`'%%%' Fi /)`
	
	If desired the format can be supplied as an external pft using @ format_name .pft
	
	Note
	
	• the text 'INV *' must appear at the beginning of each occurrence because this way the process can identify the inventory number Corresponding to determine whether or not it is included in the listings by rank or by inventory number
	
	• the classification number is added to field 1000 and then that label is mentioned in the repeatable group using always the first occurrence of the same field
	
	• / ' %%%' : this is the marker, on a final new line, to separate the different occurrences generated by the format

Tag height	Height in centimeters of the label. This value will be converted to em , multiplying it by 2.37106301584 for the purpose of fitting in the frame in the printer.
Width of the label	Width in centimeters of the label. This value will be converted to em , multiplying it by 2.37106301584 for the purpose of fitting in the frame in the printer.
Number of labels per line (columns)	The number of barcodes/tags, for pages with more than one barcode per line

3. Output examples with the following configuration file:

   • Example barcodes



Example configuration :

Classification_number_pref = ST_ V82 ^ b, v82 ^ b, v82 ^ b, v82 ^ b, "v82 ^ b," v82 ^ b, The components of formula (I)

Inventory_number_pref = NICLA_ (V (v900 ^ n), 5.0), ``, `0`), else v900 ^ n

Inventory_number_pref_list = NI_

Inventory_number_display = v900 ^ n

Label_format=@barcode.pft

Label_format_txt=@barcodetxt.pft

Height = 4 Width = 7 Cols = 3

- Example spine labels

  Configuration :

  Classification_number_pref = ST_ V82 ^ b, v82 ^ b, v82 ^ b, v82 ^ b, "v82 ^ b," v82 ^ b, The compoments of formula (I) Inventory_number_pref = NICLA_ (V (v900 ^ n), 5.0), ``, `0`), else v900 ^ n

  Inventory_number_pref_list = NI_

  Inventory_number_display = v900 ^ n

  Label_format=@lomos.pft

  Label_format_txt=@lomos_txt.php

  Height = 3 Width = 5 Cols = 4

  Sample PFT :

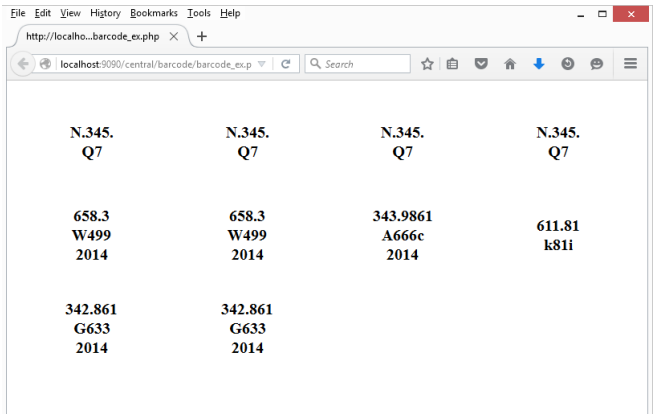
  (If p (v900 ^ n) then 'INV *' V900 ^ n '* INV *' '<Center> '

  If p (v82 ^ a [1]) then V82 ^ a [1], v82 ^ b [1], v82 ^ c [1], v82 ^ d [1], " 1], Else V82 ^ b [1], v82 ^ c [1], v82 ^ d [1], v82 ^ e [1], Fi " ',

If a (v900 ^ m) and a (v900 ^ e) then Ex. | V900 ^ 1, Else If a (v900 ^ m) and p (v900 ^ e) then | T. | V900 ^ e, | Ex. Else If p (v900 ^ m) then Vol. | V900 ^ m, | T. & lt; / RTI & gt; Ex. Fi, Fi, Fi,

'%%%', Fi /)

Example output :



- Example labels

Example configuration :

Classification_number_pref = ST_ V82 ^ b, v82 ^ b, v82 ^ b, v82 ^ b, "v82 ^ b," v82 ^ b, The components of formula (I) Inventory_number_pref = NICLA_ (V (v900 ^ n), 5.0), ``, `0`), else v900 ^ n

Inventory_number_pref_list = NI_

Inventory_number_display = v900 ^ n

Label_format=@eags.pft

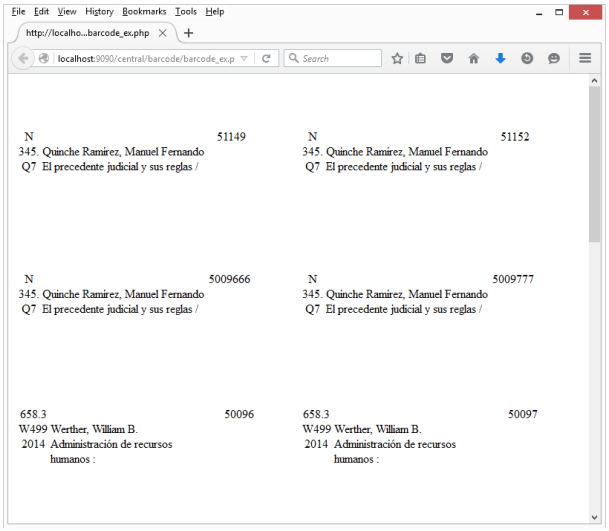
Label_format_txt=@etiques_txt.php

Height = 5 Width = 10 Cols = 2

Example PFT for labels :

('<Table> <tr> <td valign = top align = center width = 50>' 'INV *' V900 ^ n '* INV *' If a (v82 [1]) and a (v84 [1]) then " fi, If p (v82 ^ a [1]) then V82 ^ a [1], v82 ^ b [1], v82 ^ c [1], v82 ^ d [1], " 1], Else V82 ^ b [1], v82 ^ c [1], v82 ^ d [1], v82 ^ e [1], Fi, " ' If a (v900 ^ m) and a (v900 ^ e) then Ex. | V900 ^ 1, Else If a (v900 ^ m) and p (v900 ^ e) then | T. | V900 ^ e, | Ex. Else If p (v900 ^ m) then Vol. | V900 ^ m, | T. & lt; / RTI & gt; Ex. Fi, Fi, Fi, (V100 ^ a [1]) then " ", v245 ^ a, '</ Td> <td width = 50 valign = top align = right>' V900 ^ n '</ TD> </ tr> </ table> %%%' /)

Example output :



Output destination : Send to

The medium to which the issued labels are sent according to the following possibilities :

screen : Sends the generated output to the computer screen. You can use the Print option in the browser menu to obtain a printed copy; In this case you must configure the printer to set the margins and remove all page headers

MsWord Document : Send the output to a word processor. It has been tested successfully in OpenOffice. Once the document is displayed in the word processor, you must set the word's margins

Txt Use this option to generate a temporary file with other software that you want to use to print generated tags. In this case ABCD will use the above Display Format (PFT). Send to TXT in the corresponding configuration file.

Selection of records

The following methods are provided to define the set of records on which barcodes will be produced :

- By classification number : uses the parameters 'Prefix of the classification number in the FST' and 'Format to locate the classification number' to display the list of classification numbers. Select from that format also the range 'from' and 'to' which you want to extract. A tag will be generated for each inventory number contained in each record in the selected range.
- By range of inventory numbers : this method uses the parameters 'Inventory Number Prefix in the FST' and 'Format to locate the inventory number to display the list of classification numbers'. From the same PFT, the range 'from' and 'to' to extract will be taken. A tag will be generated for each inventory number contained in each record in the selected range. Remember the considerations already explained above to present a list of properly ordered inventory numbers, left-padded by zero's.
- By list of inventory numbers : this method uses the parameters 'Prefix inventory number (unedited)' and 'Inventory number format (unedited)' to construct the search that locates the list of inventory numbers.
- By range of MFNs : this methods will read each of the records of the requested Mfns range. A tag will be generated for each inventory number contained in each record in the selected range.

3.11. Stock-taking tool

[for future use, to be added]

As an alternative to the ABCD-interface for stock-taking, a simple 'manual' method can be used, which will allow to produce reports on which barcodes found on the shelves are missing in the loanobjects or copies-databases, but also vice-versa : which existing loanobjects or copies cannot be found on the shelves.

To this end the following elements are needed - which can all be created using the standard ABCD Central techniques discussed earlier on in this chapter :

- a dedicated database 'stock' (already included in the databases directory as from ABCD 2.0), which will only serve to store the barcodes found on the shelves. This database has the simplest possible structure (FDT) :

```
F|1|Barcode|1|0|||X||10|||||||0||0|0|||
F|2|Date of checking|0|0|||OD|||||||||0||0|0|||
```

  where even the 2nd line is optional to store also time and date. So in essence only one simple field is necessary to store the barcodes. These then can be easily stored by scanning the barcode of the books when browsing the shelves with one single click into the data-entry form for this database and another click to save the record. As an alternative one can also store all barcodes as lines in a text-file and convert that file into an ISIS-database with the command :

```
mx seq=mybarcodes.txt create=stock now -all
```

  and in the FST one single instruction to index the barcode in v1 with prefix 'BC_' :

```
1 5 '/BC_/', v1
```

- a print format (PFT) to produce a report from the 'stock'-database checking the presence of the barcode in the copies-database (or loanobjects-database if that database-name is referenced to), which essentially should contain the following code :

```
if (ref->copies(L->copies('IN_'v1),v30)='')
then   'book with barcode <B>'v1'</B> <font color="red">
is missing in COPIES database </font><BR>'
 else 'ok  book with barcode 'v1' found in copies inventory'
fi,
```

- a print format (PFT) to produce an 'inverse' report, meaning : from the copies database checking into the stock database to see if the actual barcode is found in that database, so exists on the shelves :

```
if (ref->stock(L->stock('BC_'v30),v1)='')
then   'book with barcode <B>'v30'</B> <font color="red">
is missing in STOCK database </font><BR>'
 else 'ok  book with barcode 'v30' found on shelves'
fi,
```

Note

These are basic PFT's which can be elaborated for more decoration or inclusion of more data, e.g. the title of the book, for which another REF(L()) construct, referring to the catalog database this time, will need to be added.

The REF(L()) function actually checks whether the barcode field, found when locating (L()) the barcode in the alternate database, remains empty, formulated as '' (two single quotes without anything in between, or an 'empty string') indicating absence.

The actual production of the stock-taking reports will be done - as can be expected - from the 'reports' menu-option or toolbar-icon. When producing the report, select the PFT's mentioned above, with the following reasoning : if the existence of a barcode, found on the shelves (and thus included in the STOCK-database) is to be checked in the ABCD-system, use the PFT with ref->copies(L->copies('IN_',v1) ='') statement, if on the other hand you want to check whether a barcode (or item) existing in the ABCD-system was indeed found on the shelves, use the PFT with the statement if (ref->stock(L->stock('BC_'v30),v1)='').

Such reports can be conveniently sent to the screen, to a text- or word-processor file or, mostly probably the most handy option here, a spreadsheet, which then can be added to the stock-taking report of the library.

4. Central module : statistics in ABCD

The Statistics module can be invoked from either the main Administration menu or from the cataloging toolbar's option :

The main Statistics screen offers three functions :

1. Use an existing table
2. Create a new table
3. Generate output

a. Use an existing table

Existing tables will be listed in the options list by their row/column criteria. In the ABCD demo version one table has been pre-defined : a 'classification code by publication date' table.

Note

The list of available tables is taken from a text-file 'tabs.cfg' in the /bases/{dbname]/def/[language]/ folder. This file contains, for each pre-defined table, three values separated by the '|' character :

- the name as displayed in the menu of tables (which for clarity could be a mentioning of both criteria)
- the field for the rows
- the field for the columns

e.g. Classification number / Date of publication|Classification LC|Publication Date

After selection of the table one simply has to continue by using the 'Generate output' option, see below.

b. Create a new table

A table has to be defined by identifying which values (as contained in an ISIS database field) will be used in the horizontal (rows) direction and which ones in the vertical (columns) direction. ABCD will display a list of available criteria (or fields) to select from for both rows and columns.

Note

The list with available fields or criteria is taken from the text-file 'stat.cfg' in the folder /bases/[dbname]/def/[language, which contains on each line one criterion, specified as a field name and a PFT to exactly define how the values should be extracted from the field. The MARC database demo example is :

Classification LC|v50^a

Date of publication|if val(v260^c)<2000 then '1900-2000' else F(val(v260^c),1,0) fi

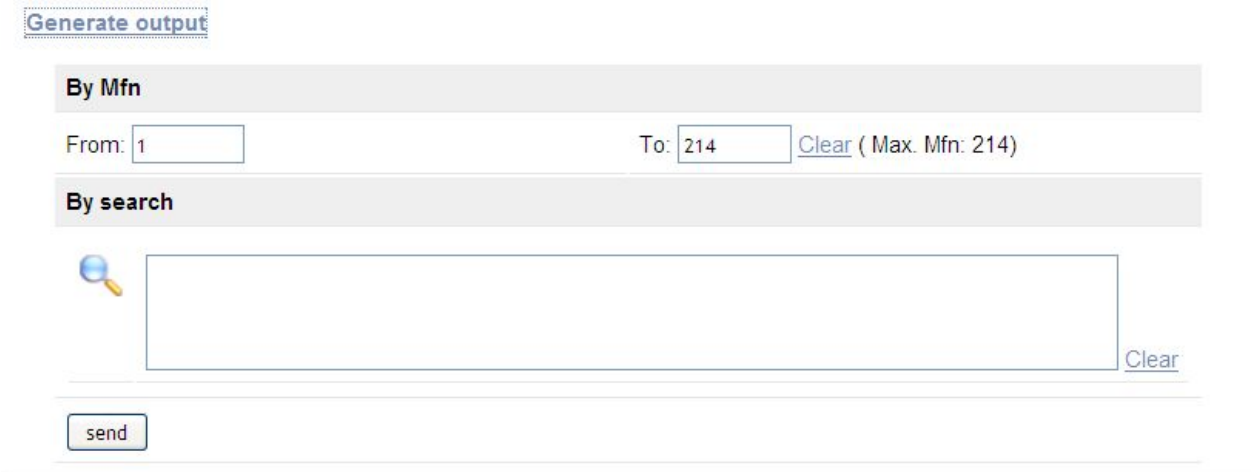
in which the second line illustrates how it can be more than just a simple field value statement by using conditions etc.

Once both the 'Rows' and 'Column' criteria have been defined, the table can be used to generate output as explained below.

c. Generate output

Generation of output in ABCD involves 2 stages : creating the table with the values and - if so desired - creating graphical output charts or exporting the table to other external document formats (e.g. worksheet for spreadsheet software which offers mostly more advanced/detailed graphical output possibilities).

Before creating the output one has to define the range of MFN's (ISIS records) to be used or a 'query' statement to apply the statistics only on a certain sub-set of the database. In the 'search' (or query) box one can use the ISIS Query Language with any accepted statement.



In the first stage it is a matter to actually create the table with all the values in the cells combining both row- and column criteria (e.g. the documents published at a certain publication date belonging to a given

LC classification code). Be careful to not calculate tables with individual extended range values (e.g. all years or all LC-codes), but rather use the Formatting Language to combine values into classes, as these make probably much more sense in your statistical report. The example above gives some hints on how to obtain such classes by using a condition on the value of a date, so the user can derive more classes from this basic example.

In this sense the following excerpt of a table gives a BAD example, as cell values are mostly if not always '1' due to too-detailed row- and column- criteria :

Clasificación LC/Fecha publicación

	Fecha publicación					
Clasificación LC	1900-2000	2000	2001	2002	2003	Total
No data	5					5
AS4	1					1
AS4.U8				1		1
AS4.U82				1		1
B1802	1					1
B4815.L84	1					1
B841.4	1					1
BF125	1					1
BF728	1					1
CR115.V4			1			1
D16.8	1					1
DL411	1					1
F2269.1.C38				1		1
F2313	1					1

The output options given by ABCD are the following :

Send to: Worksheet | Document | Printer | Animated graphic (requires flash) | Nonanimated graphic |

- Output to a worksheet will transfer the table to a Spreadsheet for further processing
- Output to a document will transfer the table to a Word Processor document for further processing - this could be practical to include results into your annual report document etc..
- Output to printer allows direct printing of the table using the printer(s) available to your Operating System
- There are 2 graphic outputs provided by ABCD itself :

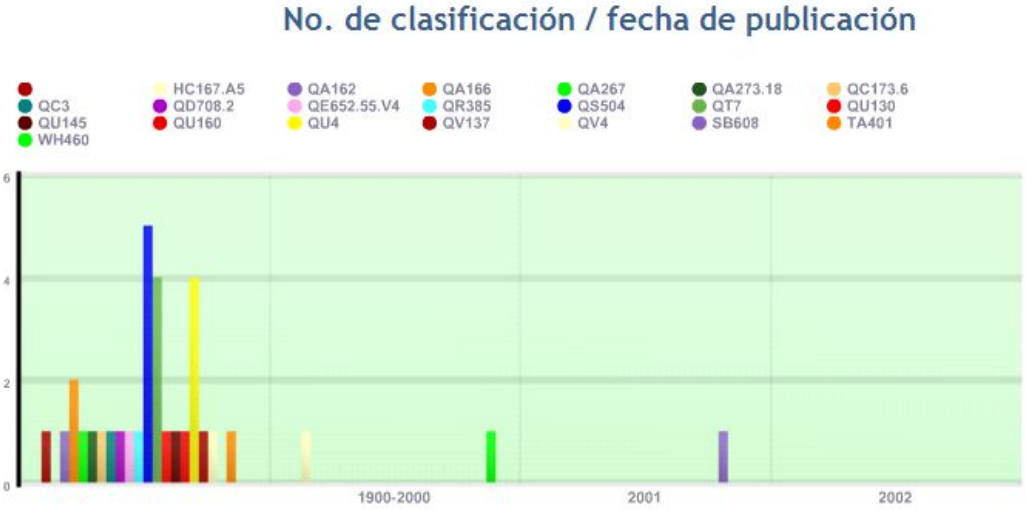
  i. Animated graphic : this is a fancy way of displaying the graphs, mostly suited for attention-tracking presentations (but requiring Flash software to be installed on the computer)

  ii. Non-animated graphic : displaying the graphs but without the fancy animation.

  In both cases several typical graphical output styles (horizontal or vertical bars, lines, three-dimensional bars etc...) are available and should be selected.

If you are not acquainted with such styles, why not just try them and decide for yourself which one presents the message of the statistics in the most clear way ? Remember that conveying a message is the crucial thing here with statistics, not impressing an audience or reader with a jungle of too-much data - as is often the case with statistical tools. Often the simpler the more convincing !

An example output of the graphical output looks like this :



5. Central module : acquisitions management

This module deals with the administration of newly acquired objects and a pre-cataloging function. The pre-cataloged objects can, once acquired, be stored as objects for the Loans module. This is what ABCD 'integration' is about.

The acquisition module has the following main logical functions :

1. Suggestions : the starting process of obtaining objects
2. Purchase orders : the actual acquiring of objects
3. Databases : management of the 4 acquisitions-related databases (suggestions, providers, orders and copies)
4. Administration : configuration, statistics and reports, weeding (discarding objects).

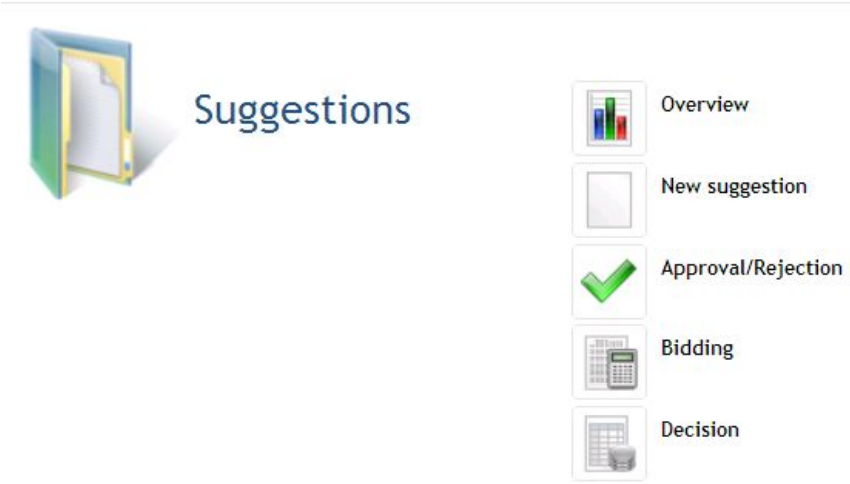
Let's discuss each of these in more detail.

5.1. Suggestions

The logical 'workflow' of acquisitions starts with suggestions (by library users, colleagues...) to purchase a book, followed logically by a purchase decision (approval or rejection), a bidding process and a decision on where to buy the book. Where the suggestions come from, at this time, has not yet been included into ABCD, but it is conceivable that a form to submit suggestions exists either on a webpage (e.g. the ABCD Site) or an e-mail can be sent to the librarian, either to be converted automatically into a suggestion record or to be manually edited into such a record by a librarian.

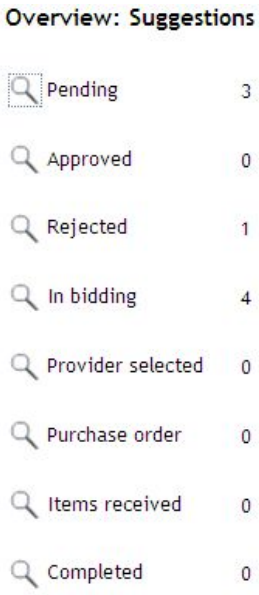
A record on suggested books, whether or not actually purchased, needs to be kept for future use (e.g. when the same book is suggested again).

This 'logical' flow is more or less reflected in the ABCD-acquisitions module.



5.1.1. Overview

Here the librarian can consult the status of the actual acquisition system's activities, with an option to get a listing (popping up in a separate window) of the activity.



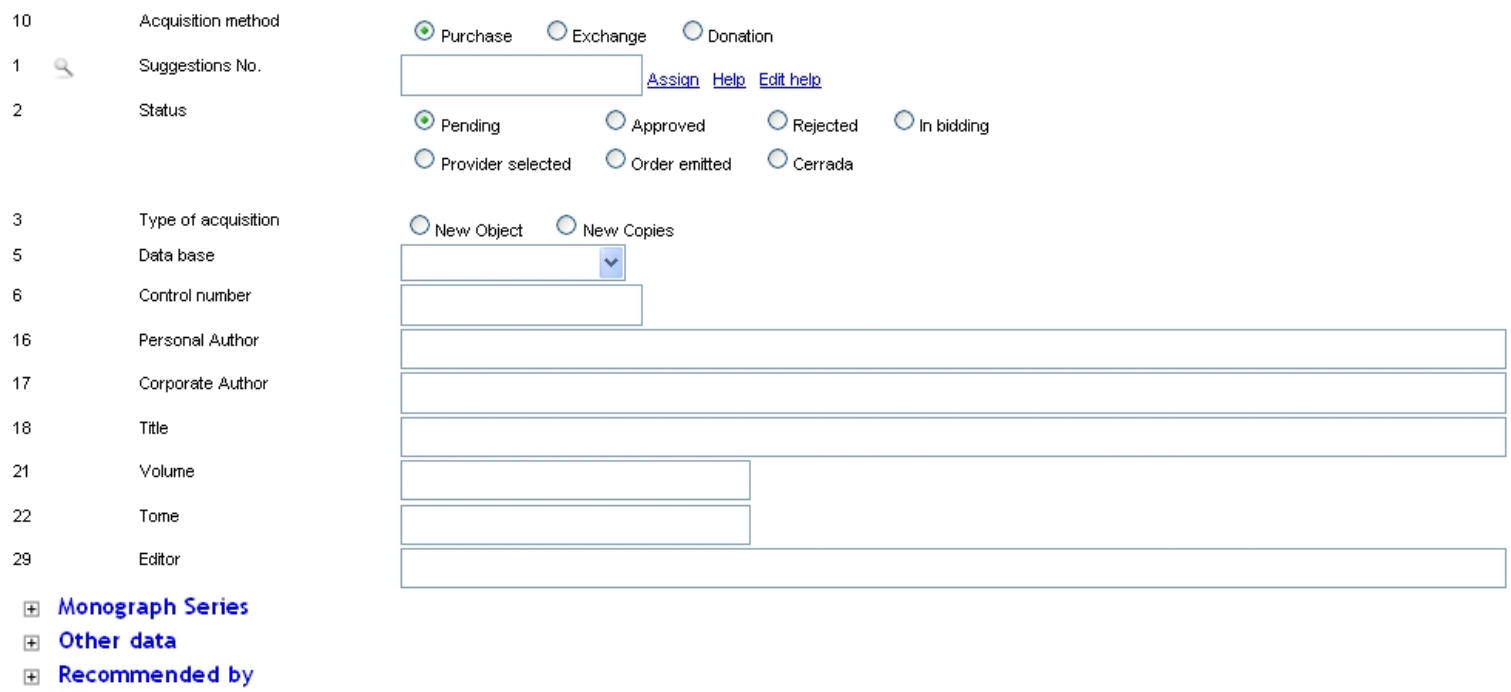
No changes or editing functions are provided here, so this is only for consultation of the acquisitions system.

5.1.2. New suggestions

Here a worksheett is provided to actually enter a new suggestion for acquiring an object, with some bibliographical fields but also the status and a 'recommended by' field. Under here is shown part of the data-entry worksheet.

Note

For the 'date of suggestion' field a calendar pop-in function is used.

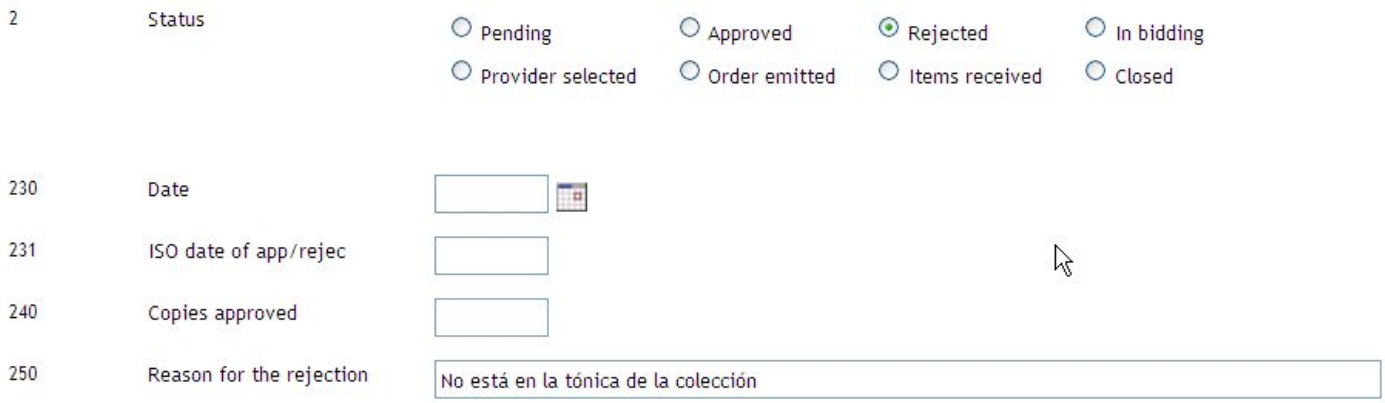


5.1.3. Approval/Rejection

ABCD will give a listing of the requested activities with their status, sortable by either Title, Recommended by or Date of recommendation.

No.control	Recomendado por	Fecha	Título	Autor	No.copias	
38	Ascencio, Guilda	10/02/2009	Music for liquid days	Glass, Philip	3	Rech
39	Ascencio, Guilda	10/02/2009		Marai, Sandor	2	Penc
37	Zubillaga, Erasmo	07/02/2009	El último encuentro	Marai, Sándor	5	Penc

Each of the items can be opened for editing, where the actual approval or rejection can be given. Under is an example of a rejected item :

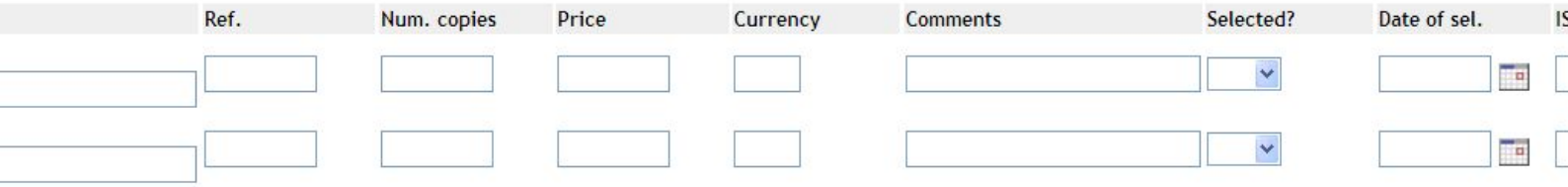


At this point, after having either cancelled or saved (updated) the transaction, direct access to the main Acquisition menu's options is also offered :

Also options are provided to send the listing to either a document (.DOC format) or a Spreadsheet (.XLS).

5.1.4. Bidding

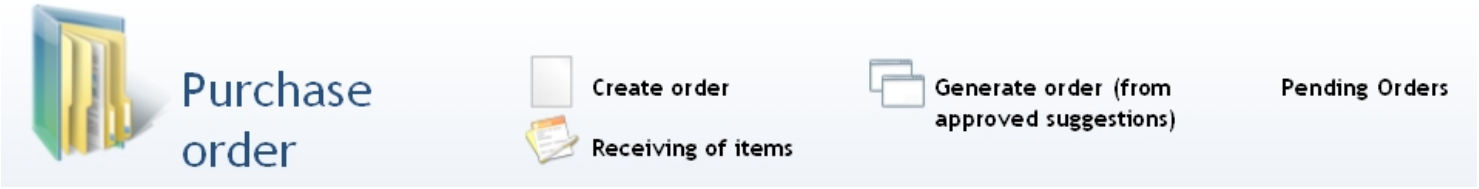
At this stage ABCD offers, starting from a sortable listing of the actual transactions, a worksheet where the bidding process can be kept track off : for each participant in the bidding some essential data (name of bidder, price offered etc.) can be stored with a box to mark acceptation or not :



5.1.5. Decisions

Finally the recommendations administration is completed by a listing of the approved offers in a format (work-sheet) similar to the above. Again this listing can be exported to either a document or a spreadsheet software.

5.2. Purchase orders

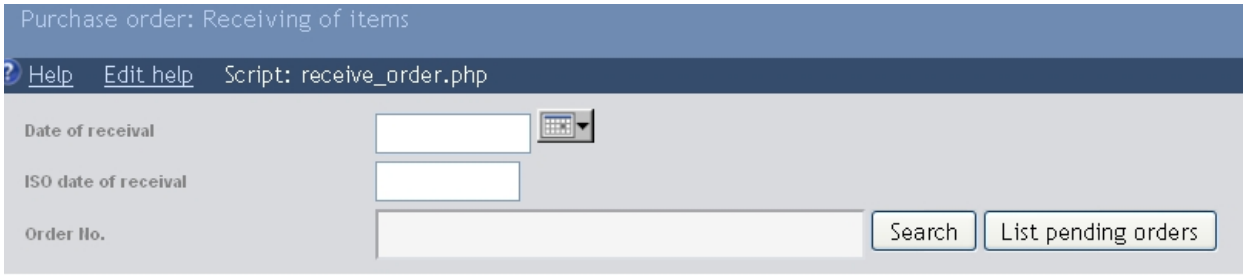


[!!] The new acquisitions first need to be categorized as either purchase, donation or exchange. According to the selected option the subsequent editing form will contain slightly different elements, but these don't need a lot of explanation here.

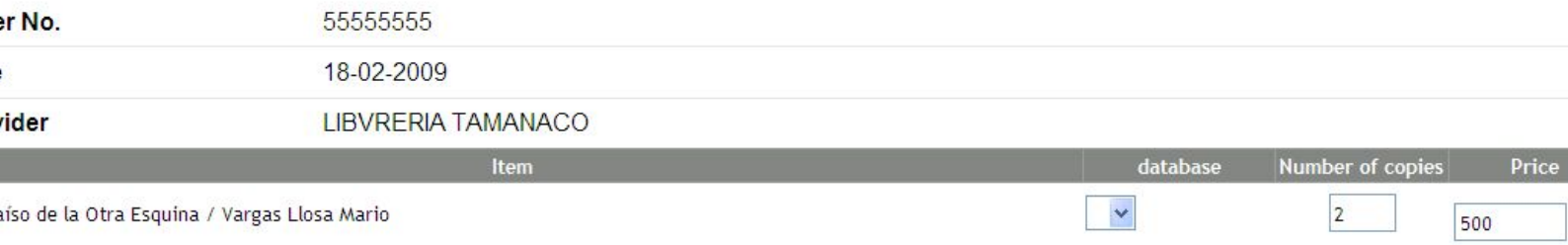
> **Note**
>
> In many cases libraries obtain objects without having gone through the 'suggestions' stage. Here it is possible to create a new order without having to refer to the suggestions phase. However there is another option here to deal with approved suggestions and to continue the full procedure into the actual acquisition process.

ABCD facilitates the ordering of objects by providing a form to create a new order, by listing the pending orders with editing possibility and by providing a listing or search possibility in the list. The received items, with their price and number of items acquired, can be filled in. Whenever a selection has to be made from a list of candidates, these can be sorted by different criteria which are listed above the list itself.

Acquisitions which have been received can be entered into the inventory with their date and order-number in a small form :



Listing the received orders is the last option in here :



5.3. Databases

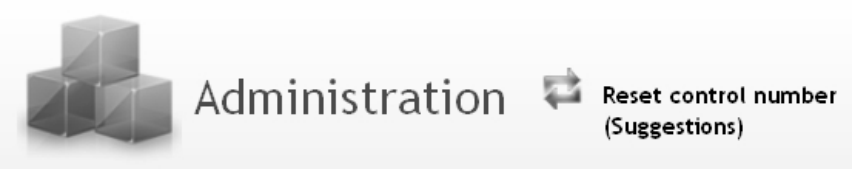
The ABCD Acquisitions module maintains 4 databases, which can be accessed directly here, each time with a search function for fast retrieval of a specific record. Each of the retrieved records can then be edited with its dedicated worksheet and saved if indeed changes have been made. As an example we show under here the providers database.

The 'Create' icon allows to create a new provider record.

5.4. Administration of acquisitions module

In this last section of the Acquisitions module some functions are offered to manage the acquisitions system with reports, statistics, configuration of some parameters and also weeding the acquisitions system by deleting unwanted

items. In the initial version only one function is provided : reset the control number (which is the number of the last entered item or copy control number, to which 1 will be added when creating a new one ('auto-increment' field). Actually this number is kept in a file 'control_number.cn' in the folder \ABCD\www\bases\copies\data\, where simply the last assigned number is stored. Whenever copies are created, this file is write-protected to avoid duplication of the same number by concurrent users - so other users have to wait until this file is given free for writing.



6. Central module : loans/circulation

This section is about the ABCD basic loans module as it comes pre-configured with the system. Still, always following the ABCD philosophy of flexibility, the loans system can work with any bibliographic and objects database.

In addition, ABCD offers an 'advanced loans' module (EmpWeb) which can cope with much more complicated situations in e.g. multi-branch organisations with different loans policies etc. [!!] The transactions in this Advanced Loans module will be stored in SQL-tables and it can also access user-data (through WebServices) stored in SQL-tables elsewhere in addition to accessing the user-data in the ABCD-user database (in ISIS-format). This of course allows very advanced high-performance applications of the loans module. In addition EmpWeb or ABCD Advanced Loans will offer a reservation function and a function 'MySite' where end-users can check and keep track of availability of loan-objects in their personal (password-protected) account of ABCD-Loans.

[!!] This advanced module however requires installation of additional software (Java/Jetty and an SQL database) and is only offered as an optional extra module.

We suggest to check the following criteria in order to decide on whether you will need the advanced module or not :

- multiple servers in your system ?
- multiple users/copies databases in your system ?
- high volume of transactions ?
- any data source needs ODBC drivers ? (ODBC drivers are software to couple mostly relational databases or SQL-tables to software)

If you have a 'yes' with serious importance for your system, you will be better off with the advanced loans module. [!!] Most smaller libraries however, certainly where there is a lack of expertise on using Java and SQL-databases, will be better of with this ABCD Central Loans module as it is fully based on the ISIS-database technology and doesn't need any additional software to be installed.

6.1. The ABCD inventory copies and loanobjects databases

ABCD uses two different databases to deal with information based on the physical objects in the library : the COPIES and LOANOBJECTS. Both databases have different purposes and scopes and therefore the data are kept separate. A good understanding of their different purposes and structures will help in understanding ABCD. Let's discuss each of them here.

1. The ABCD copies database.

   This database has the function of keeping track of all administrative data on objects in the library collections. These data in principle need to be kept for inventory and book-keeping reasons, whereas data related to loan-objects can be deleted when the loan-object is removed from the collection. Actually the record created at the

end of an Acquisitions procedure (from suggestion to received item or anywhere in between) will be stored in this copies database. The copies field structure is as follows (field tags are followed by the field name) :

1|Control number	68|Acquisition type
10|Database	70|Provider/Donor/Institution
20|Call number	80|Date of arrival
30|Inventory number	85|ISO Date of arrival
200|Status	90|Price
35|Main Library	100|Purchase order
40|Branch library	110|Suggestion number
60|Volume/Part	300|Conditions
63|Copy Number	400|In exchange of

The field 30 (Inventory Number) is created by the auto-increment mechanism of ABCD : the next number to be assigned is stored in a small file 'control_number.cn' in the data-folder of the database (in this case COPIES), as each record has to be identified by a unique value which can be used for linking from the other databases (as 'primary key').

Differently from the LOANOBJECTS-database the COPIES-database has one record per object. These records are normally created by the 'Add to copies' function of the cataloging module.or the Acquisitions module.

This database has to be indexed with 2 mandatory keys :

1 0 "CN_"V10,"_"V1,

200 0 "STATUS_"V200^a

The first key here assures that the identifier of each copy can be quickly referred to by the prefix 'CN_' followed by the bibliographic database name V10 (because several catalogs can be combined into one copies-database) and the identifier in that catalog itself (V1). For an entry in the copies-database to be allowed to enter into the loanobjects database it needs such an entry in the Inverted File (and assumed the status-field is at least at level 2, which means 'verified and stamped', that is why the second FST-entry indexes on the status of each copy).

In the cataloging module copy records will be shown in a separate smaller window when clicking on the 'Add copy' icon of the record-toolbar and then on 'show copies'.

2. The ABCD loanobjects database

This database has different goals from the copies-database : it's purpose is only to provide the necessary data about all objects which can be used in the loans-system. Its contents are quite limited :

1|Control number

10|Database

959|Copies

subfields :

i|Inventory number

l|Library

b|Branch library

o|Type of Object

b|Volume

t|Part

As can be seen from this very short list of fields (in fact only 3), each loan-object is identified by its control_number V1, the bibliographic database to be linked to V10 and the actual details on each copy as a repeated subfielded field V959 (this gives, in ISIS-technology, the fastest performance in case of high number of copies). The subfields allow to specify the unique identifier (e.g. barcode) of each copy, the main library and location within that library (e.g. shelving information), the loan object type (books e.g. will have different loan-parameters from video's) and in case of a multi-volume work the volume and part - identifiers.

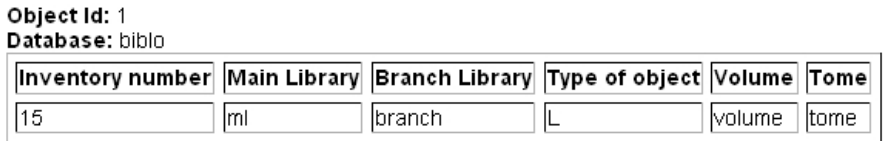
The loanobjects database has to be indexed with at least the following instruction :

1 0 "CN_"v10,"_"V1

because (as in the copies-database) the string 'CN_' followed by the catalog-name and catalog-identifier allows each copy to be identified. If this entry cannot be found in the loan-objects database for a given book, it will not be possible to use the object in the loans system.

Loanobjects normally are created from the cataloging module after having created the according inventory copy with the icons in the record-toolbar of the cataloging module :

After having clicked on 'Add to loan objects' the record will have been created in the loanobjects database. If this database is to be edited directly as an ABCD-database, remember to include it in the list of databases (bases.dat, which can be accessed directly from the Operating System but also through the corresponding 'Update database definitions' menu in ABCD Central) and to the profile of users which need access to this database directly. There the loanobject records will be presented for browsing and editing in a table format (as explained in the section on the FDT-definition for 'group'-fields).



6.2. The basic ABCD loans module

6.2.1. Introduction

This loans module is called 'basic' because it is fully integrated with the other ABCD Central modules, using the same underlying technology: ISIS-databases, ISIS Script and PHP. Looking at its functionality however one could hardly call this 'basic' : this module takes as its departing point the 'objects' created by the acquisitions module into the database 'copies', to apply rules on all kinds of 'transactions' on them : issuing to a user, returning, reservation, loans renewal. Rules for all types of transactions can be defined and will be applied according to the object category in combination with the user category. Categories for objects and users can be defined 'ad libitum' with specified number of objects, hours/days (taking into account a calendar specific for the library), fines and renewal conditions for each combination object/user.

The main menu of this loans module has three sections :

1. Transactions : here the real every-day loans transactions (loaning a book to a user, returning it, reservations etc.) are dealt with.

2. Databases : here the databases on which the loans system is based can be accessed and managed : the borrowers or users, the transactions, the reservations and the fines.

3. Configuration of the loans system : here the 'rules' can be defined for combinations of object types with user categories, and calenders, currency etc. can be defined.

6.2.2. General loans parameters and configuration in abcd.def

The file 'abcd.def' (in the bases-folder of the ABCD-system) contains the parameters applied to the whole system. Some of them relate however specifically to the ABCD Central Circulation system. A brief listing with explanation is given here :

Table 2.2. ABCD Central Loans configuration parameters in abcd.def

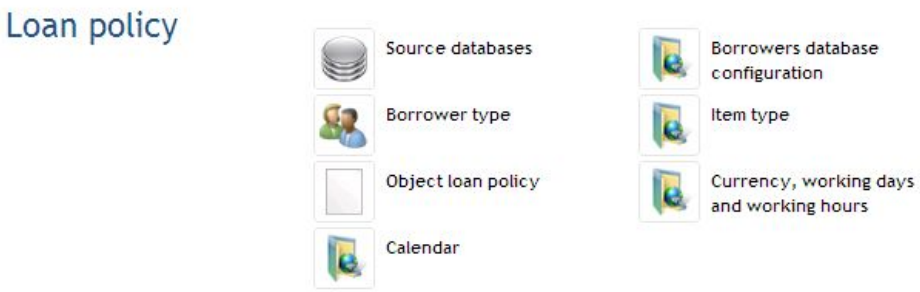
LOGO_OPAC	Url of the logo that is displayed in the key request, statement and online reservations
BG_WEB	Background color of the key request window
WEBRENO-VATION	Y / N to enable / disable online renewal option
RESERVATION	Y / N to enable / disable access to the reservation process from the loan menus
LOAN_POLICY	BY_USER to indicate that the object type of the item to be loaned is requested at the time of processing the loan. This option should be used when it is not provided from the database of copies and the inventory of the objects does not have a subfield with the type of object
ASK_LPN	Whether or not the option to request the return date is enabled, ie not calculate it from the policy
E-MAIL	Y / N to enable / disable sending loan reminder emails
AC_SUSP	Sets the date from which a suspension begins. The Y value indicates that it must start from the last active suspension of the user. The N value indicates that it starts from the day it is recorded.
CALENDAR	To establish the way in which the period of fines and suspensions is calculated

As an example section of the abcd.def loans configuration parameters :

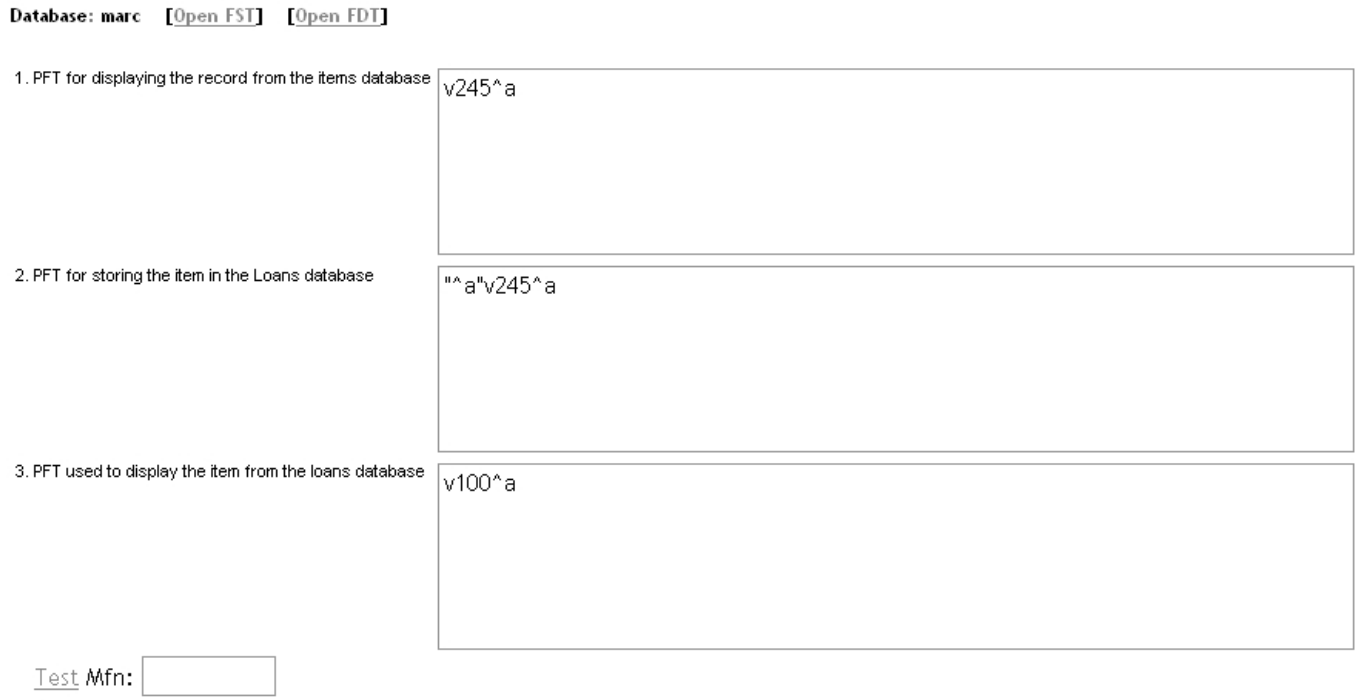
```
LOGO_OPAC = .. / css / logo_opac.png
   BG_WEB = # ffffff
   LOAN_POLICY = BY_USER
   WEBRENOVATION = N
   EMAIL = Y
   RESERVATION = N
```

6.2.3. ABCD Loans detailed configuration

The loans configuration in ABCD allows to define which source bibliographic databases (catalogs) to link the loans system to - it can be any database indeed ! - and to define the parameters which will constitute the 'policy' on each object/user combination to be applied.



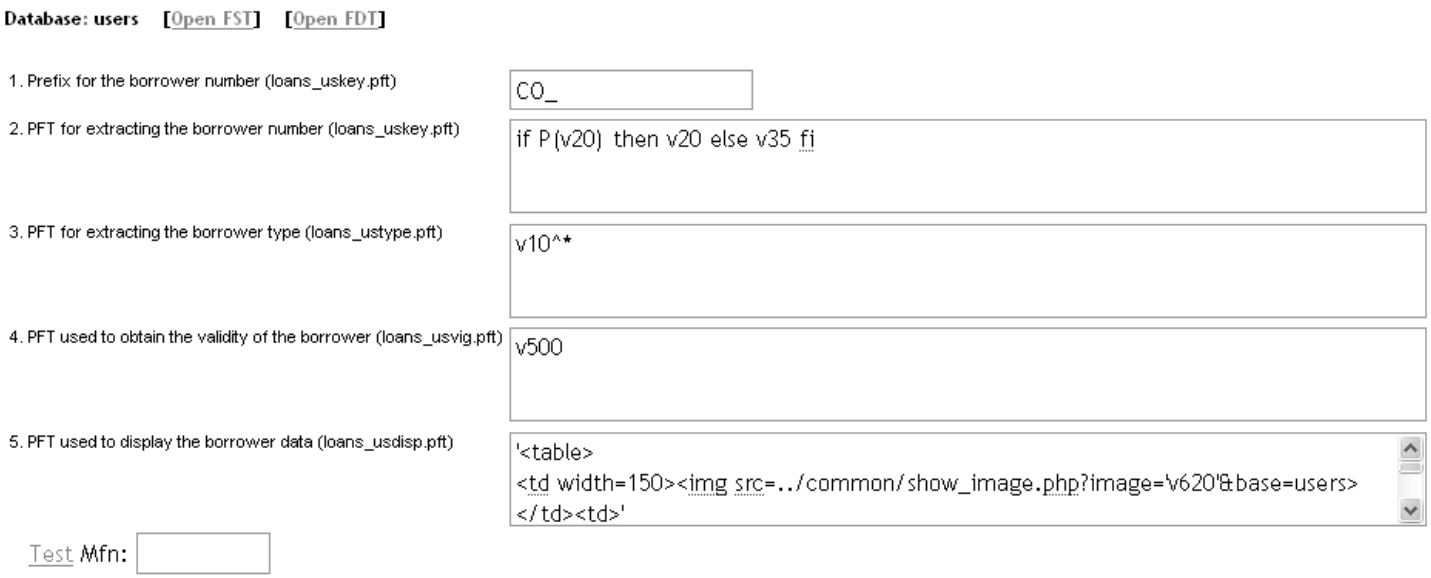
Since any database can be linked into the loans system as 'source', there is a need to explain to the loans system how values from these databases will be used in the loans system. We can best illustrate this by giving the example of the CEPAL-database of the ABCD-model application :



This form shows the information needed for the loans system, e.g. which prefix is used in the index for the accession number or which print formats (PFT's) to use to produce the data in the loans screens.

The power of the Formatting Language can be applied here, of course. For example, instead of the rather dumb example above here as the 'PFT to be used to extract the type of record', one could define a different type (with consequently different loan parameters for the type) according to some conditions, e.g. the date, the month etc. So an object which is a normal loans object could be changed into 'special material' during the exams period etc. The ISIS Formatting Language provides most necessary functions (e.g. Date() with substring extraction) for this purpose.

The same applies to the definition of the borrowers data :



Instead of simply taking the 'value of Field 10' (v10) to define the borrower type, one could put a more sophisticated Formatting Language statement here in order to make the status of the borrower even dynamically dependent on other conditions (again : date, but also other conditions can be defined).

Using a table ABCD will then present the defined borrowers types and allows to add any number of more such types :

Add a row before the selected one Remove the selected row

Borrower type	Description
di	Directors
co	Coordinators
te	Technical staff
ad	Administrators
ex	Experts
po	Postgraduate
pr	Professors
cl	Consultants
al	Alumni

Here we only show part of the table, but in fact the interface will always offer a few more empty lines to add more types, and lines with a type can also be added in between existing rows.

The same approach is used for the definition of the objects types :

Add a row before the selected one Remove the selected row

Item	Description
L	Books
V	Videos
CD	CD's
R	Reserved
K	Item type k

Needless to say that each time a 'cancel' or an 'update' button is provided to either cancel the editing of the table or to actually store it again.

From these two types (users and objects), ABCD then creates the 'loans policy table', which lists in a matrix all possible combinations of user types and object types, and parameters can be entered for many aspects of the loans policy :

Bor. type	Lim loans	Length (norm)	Length (res)	Unit	Renew	Fine	Fine (reserv)	Days susp	Days susp (reserv)	Days wait	Permit loan overdue	Permit renew overdue	Multiple copies	Deadline user	Dead obj
Directors	7	2		Dias	1										
Coordinators	5	3		Dias	1	2						Si			
Professors	5	3		Dias	1	2						Si			
Administrators	4	2		Dias	2	2				1	Si	Si			
Experts	2	2		Dias	1	1						Si			
Technical staff	2	2		Dias											
Directors	10	3		Dias											
Coordinators	5	2		Dias											
Directors	2	2		Dias											

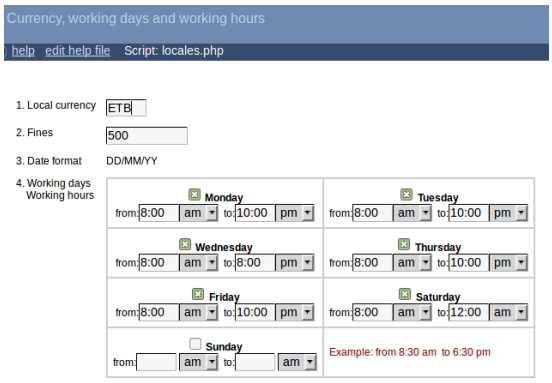
ncel

As can be seen, lots of parameters are stored and used in the decision-making process of each transaction (e.g. is this user allowed to loan this type of material, how many of them, for how long, what is the fine for late return etc.). Units can be either days or hours and the calculation of 'number of days elapsed' will be based on a calendar function (see below).

A special (new as from v1.5) feature is to allow the librarian to divert from the stored 'duration' parameter : at the time of lending out the number of days or return date can be changed manually, making it different from the official calculated duration according to the defined policy. For this a new parameter is to be added into the file abcd.def : ASK_LPN =Y/N.

Configuration of the loans system continues with two more options :

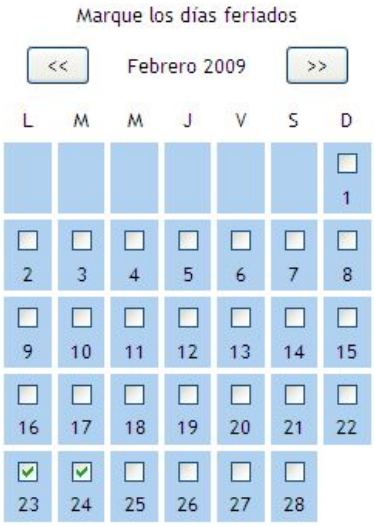
- definition of the currency , fine unit, date format and working days/hours :



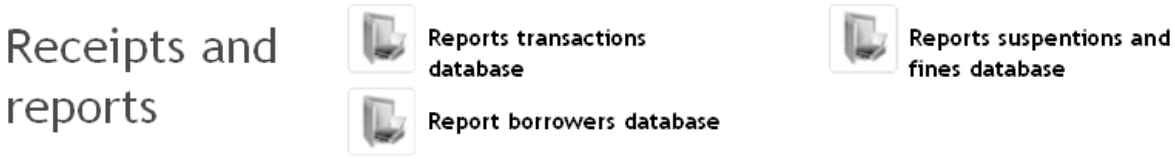
Caution

In order for the loans system to work well, don't leave this 'working days' calendar empty ! If no working hours or days at all are defined the creation of a loans records will fail as no return date can be calculated.

- definition of the holidays (non-working days) in the calendar , where simply the holidays need to be indicated on each month's map :

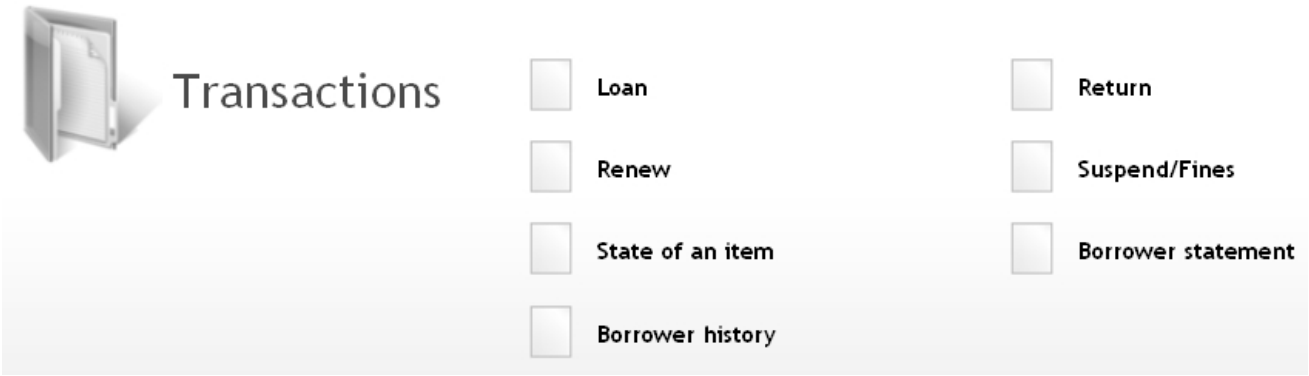


- generation of Loan reports : for each database in Loans (transactions, borrowers and suspensions) a set of PFT's (as these are in fact the reports in ABCD) can be generated using the same interface as used for other PFT's (e.g. in the Database Administration module). As the procedure is fully identical we won't repeat the steps here, please refer to the dedicated section in the Database Administration module chapter.



6.2.4. Transactions : loan, returns, reservation, renewals, fines/suspensions

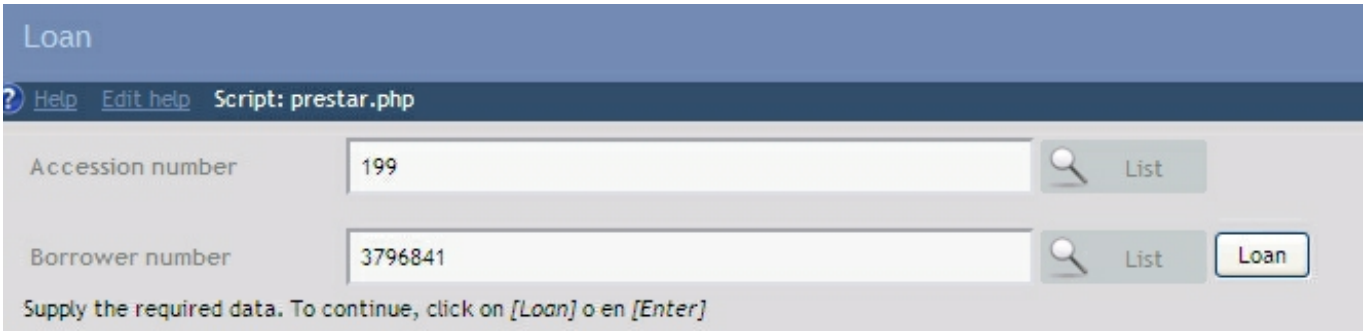
Most of the transactions themselves are rather easy to understand. Efficiency is the key here : mostly the system simply needs one or two bar-codes to be scanned in, and pressing a button 'go' to store the transaction. A list of possible transactions is shown in the transactions menu of ABCD :



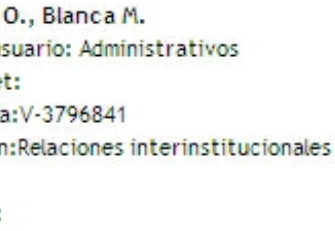
Let's discuss each of them now.

6.2.4.1. Issuing a loan on an object

This page offers identification boxes (in which either the barcode or identifier can be input directly or selected from a list) for both the object and the user, the two crucial elements of any loans transaction.



After having identified both elements, a simple click on the 'loan'-button will actually create the transaction (into the 'loans' database). The user information is shown and all loans transactions related to the users will be listed in a table, where one or more transactions can be selected for immediate editing (e.g. returning or renewing the loan).



Control number	Classification number	Reference	Item type	Loan date	Devolution date
42			TOTAL:0	28-03-2009 11:24:51 AM	29-03-2009 11:24:51

6.2.4.2. Returning an object

Here only the object returned needs to be identified and with a simple click the transaction record in the loans database will note the fact that the object has been returned. The loaned object can also be returned from the table in the borrowers' statement, then the transaction will be removed from the table. [!!]

6.2.4.3. Renewing a loan

This is a simple continuation of a running loan, but dependent again on specified rules as on whether the object has not been reserved by someone else and the user requesting the renewal has no pending fines etc. When consulting the list, only the objects on loan will be listed. In case all conditions for renewal are fulfilled, the transaction will be granted and listed, if not a warning or error message will be shown, e.g.

6.2.4.4. Fines and suspension of users

Fines and suspension of users are offered in the same ABCD-page :

Tanio Reyes, Josefa
Tipo de usuario: Directores
No.carnet:02
No.Cédula:445436789
E-mail:
Teléfono:

Sanction type	Suspension
Date	28-11-2009
Number of days of suspension or number of fine units to apply	11
reason	see decision Board Meeting dd. 22/11/2009
Comments	this is just a test suspension

Update

For the selected user the following fields can be filled in : the type of sanction (fine or suspension), the date, the number of days or the amount of the fine, the reason (motivation) and any comments.

6.2.4.5. The borrower statement

From this screen all information on a borrower or user is displayed, giving also direct access to other functions where only an access from the object was given, e.g. to allow renewal from the borrower's identification instead of the object.

[!!] Interesting to note that borrowers not only can be selected from the list of borrowers but also from the inventory-number of the objects loaned by the borrower.

Tanio Reyes, Josefa
Tipo de usuario: Directores
No.carnet:02
No.Cédula:445436789
E-mail:
Teléfono:

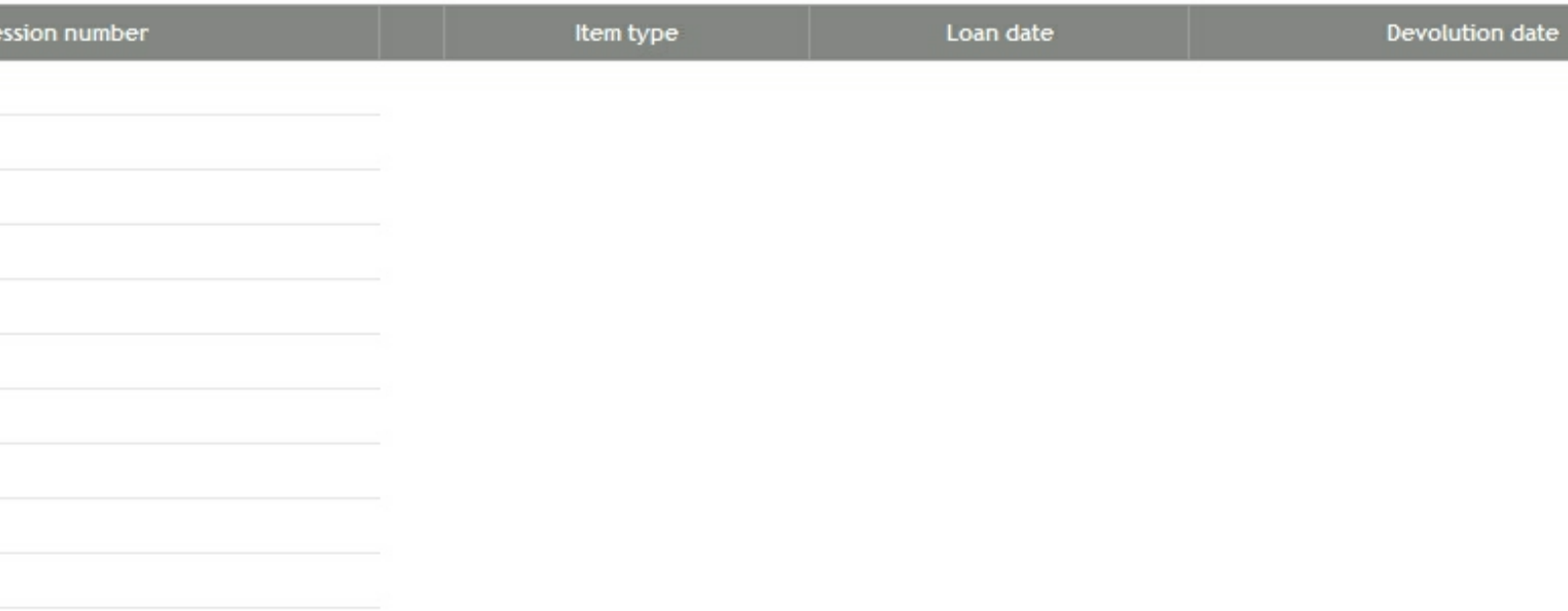
d	Date	Concept	Amount	Comments
	27-06-2009	Fines (20)	12	
	29-06-2009	Fines (20)	20	
	29-06-2009	Fines (20)	20	
	29-06-2009	Fines (20)	20	

n	Volume	Tome	Control number	Classification number	Reference	Item type	Loan date	Devolution date	Over
			3(biblo)	BX4700.L7T42	Tellechea Idígoras, José Ignacio. Ignacio de Loyola: la aventura de un Cristiano, Compañía de Jesús, Provincia de VenezuelaUniversidad Católica Andrés BelloFundafesi. 1997. Monografías.	L	22-06-2009 06:21:28 AM	24-06-2009 06:21:28 AM	15

w

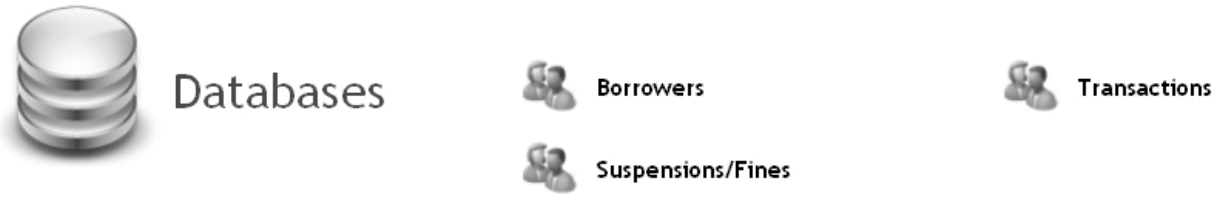
6.2.4.6. State of an item

As with the borrower's statement, an overview (history) of all loans of this given item or object can be retrieved here.



6.2.5. Databases in the loans module

The following databases are used in the Central Loans module :



For each of these databases, whose names explain their purpose, an interface is presented which lists the records with a search function and edit or delete buttons.

The transactions database records the actual events in the loans system. ABCD opts to keep this database as 'mean and lean' (i.e. compact) as possible, without duplicating data e.g. from the bibliographic databases. This database will be rather 'dynamic' with many movements, and since ISIS is not at its best in such environment (for which simple tables with fixed structures would be more suited), we need to keep its structure as compact as possible, using the REF-function of ISIS to 'loan' data from other databases, e.g. the bibliographic data.

So this will allow the librarian to directly interact with the records of the borrowers (e.g. to create new library users), the transacions (e.g. to check a loans record), reservations and suspensions or fines. In this last database the librarian could interfere with existing due fines and suspensions of users in case of a necessity to do so by-passing the rules - take care !

6.2.6. Loans administration

This third and last section of the loans module not only offers the loans configuration option (discussed above here) but also gives access to the Statistics module (which is also discussed elsewhere in this manual) and the 'Reports' option, which will be added in a later version of the ABCD-software. This module will create all types of output documents, e.g. alerts, confirmations of loans etc.

6.3. The advanced loans module

The advanced loans module of ABCD is an add-on module which can be installed as a separate member of the 'ABCD Suite'. It requires additional software technology (e.g. JAVA, MySQL) to be installed and can also be run as an independent software. Since this module was originally programmed as a DOS-software named 'Emp' (Portugese for 'loans'= emprestímos', the module is called 'EmpWeb' as for ABCD it has been converted into a Web-software, using new web-technology like web-services. So we consider this 'E'mpweb as an 'E'xtra or 'E'nhanced module', maybe changing 'ABCD' into 'ABCDE' ?

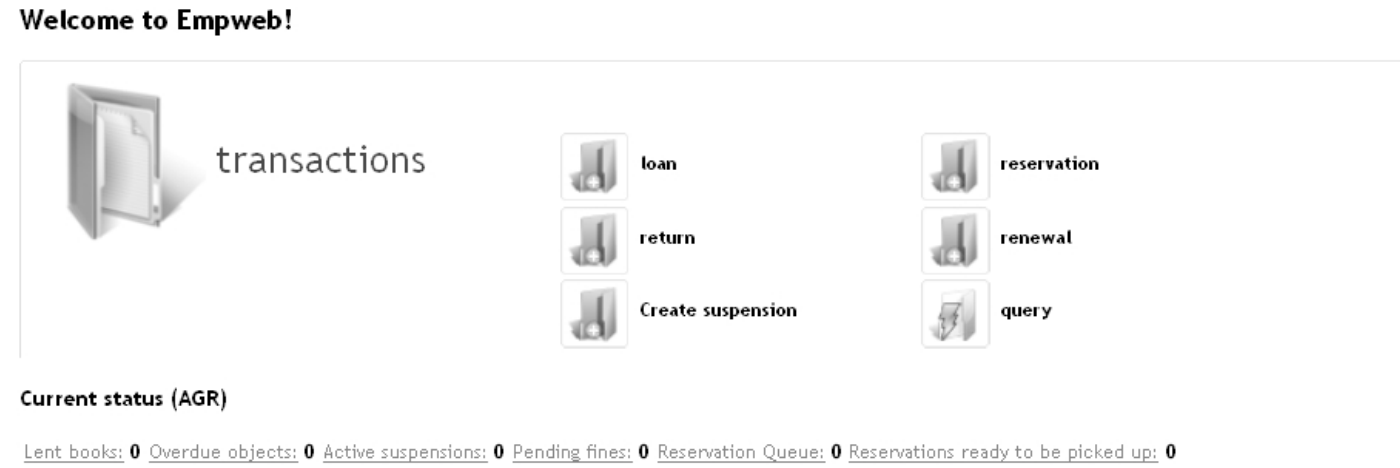
Extra functionalities as compared to the integrated loans module are :

- better capability to deal with complex organisational structures (multi-branch libraries with different loans policies, different servers e.g.)
- more robust handling of high-volume transactions situations
- more interaction with users form the OPAC module, e.g. the 'MySite' function to allow logged-in users to keep track of their own status etc from the OPAC.
- a 'MySite' function allows registered end-users (after logging in) to enter their own space in the loans system to check on their status as a library user and other interactive users. This function at this time is not yet available in the Central Loans module.

Some important concepts of EmpWeb are briefly presented here :

- web-services : instead of needing full access to external resources (databases), which can in some case create problems with the data-providers, web-based 'requests' are sent to the server to just deliver - as a response to the request - some specific data.
- pipe-lines : any transaction (like a loan, a reservation, a return...) goes through a pipe-line of conditions which can be defined. Only if all conditions are met throughout the pipe-line, the transaction will be 'granted', if not it just stops and returns the defined (by the software) error message or instruction (e.g. 'User has been suspended'). This allows any number of conditions and rules to be applied on any decision taken by the software.

EmpWeb therefore can run on any set of external data for which drivers are available or can be accessed by webservices and apply any set of rules onto these data and perform processes (like changing a record) in case of having succeeded to pass all rules and conditions. [!!] EmpWeb in this way is more of a generic transactional engine but used as a loans system in ABCD.



[!!] This advanced loans module is discussed in detail in a separate Manual.

7. Central utilities

The Central module, being the 'administration' module of ABCD, has a menu 'utilities' which is probably the most dynamic part of the system. More and more utilities have been and keep being added as needs or possible uses arise.

Utilities are special scripts which serve a purpose of more advanced, non-daily use functions mostly related to manipulating and managing databases. One example has been discussed already above : the creation of a collection of digital library records in a database from a batch of documents.

In this section we will discuss the most important utilities. Some of them however are very simple and obvious in their use (e.g. initialize database) so they don't need a lot of explanation.

7.1. The main utilities menu

From the main Central page, after logging in and having selected a database (because most utilities require a database to be active), one can select the main utilities menu :

Figure 2.1. Main Central Utilities menu

- Inverted file generation(MX)
- Copy the database to another folder
- Read database/ISO file with MX
- Restore database
- Initialize database
- Delete database
- Protect database from initialization or deletion
- Unlock database
- Assign control number
- Explore databases directory
    - par
    - www
    - wrk
- EXTRA UTILITIES

Note

This menu can be different depending on the database selected or the Central configuration. E.g. the 'explore databases directoy' can be dis-allowed. Also utilities can have been added or re-arranged.

We give some brief explanation now for each of the utilites.

7.1.1. Inverted File generation

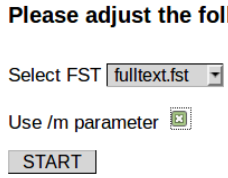
Normally ABCD indexes immediately and automatically any edited or newly created record , but often it is desirable or necessary to have a database fully re-indexed. In ISIS-technology this is called 'Inverted File generation' since a special index is created which has more features and power than normal database-indexes (which are more like sorted pointers). An Inverted File provides a dictionary of searchable terms and - in the background when a term is selected - all information necessary to provide the related record, field, field occurrence and in the case of full-fledged (or classis ISIS-)indexing also the position of the term within that field occurrence. CISIS is extremely fast in creating such indexes, so running the process over a full database typically takes seconds or some minutes at maximum. Typical usage cases are when the FST has been changed (then full IF generation is mandatory to reflect the changes) or when a batch of records has been added into the database e.g. by ISO-import.

In the Central data-entry toolbar also Inverted File update is an available option under the utilities-icon; depending on the option chosen there, either the same utility as the one currently discussed is invoked or an older version running over the HTTP-protocol and using wxis (instead of mx) is used, resulting in the same index but (much) slower. This utility in fact creates a command-line command and sends it to the server OS to run as a batch-process, only returning to the interface (the ABCD screen) when the process is finished with some result information.

The interface of this utility is quite simple :

- a selectable list of available FST's for the given database
- whether or not to use the special /M parameter, which means that if used (default is NOT) no proximity parameters are included into the Inverted File; this speeds up the process a lot, results in smaller index files but disallows the operators of 'distance' in between words, rarely used anyway.
- The 'START' button to launch the actual process.

An example here is shown for the special 'Digital Library' database 'DUBCORE', where the special 'FULLTEXT.FST' has been selected and the /m parameter is indeed used :



The result is given in detail, including the actual parameters used in the command. This might allow you to find out what went wrong if the process was not successful. For example one can copy/paste the command-line to the terminal of your server, run it there and probably get more feedback on the problem.

The process finishes when the following feedback is shown :

```
database: /var/opt/ABCD/bases/dubcore/data/dubcore
fst: @/var/opt/ABCD/bases/dubcore/data/dubcore.fst
mx: /opt/ABCD/www/cgi-bin/utf8/bigisis/mx Test
stw: stw=@/var/opt/ABCD/bases/dubcore/data/dubcore.stw
uctab: /var/opt/ABCD/bases/isisuctab_utf8.tab
actab: /var/opt/ABCD/bases/isisactab_utf8.tab

Command line: /opt/ABCD/www/cgi-bin/utf8/bigisis/mx /var/opt/ABCD/bases/dubcore/data/dubcore fst=@/var/opt/ABCD/bases/dubcore/data/fulltext.fst uctab=/var
/isisuctab_utf8.tab actab=/var/opt/ABCD/bases/isisactab_utf8.tab stw=@/var/opt/ABCD/bases/dubcore/data/dubcore.stw fullinv/m=/var/opt/ABCD/bases/dubcore/d
now tell=100
```

```
Process Result:
Process Finished OK
```

In this case of the Dubcore database, we note the following :

- a special version of mx has been used : /opt/ABCD/www/cgi-bin/utf8/bigisis/mx, which is the 'Unicode BigISIS' version as that is the one used (and this has been indicated in abcd.def and dr_path.def of the Dubcore-database.
- a stopwords-file was found in the Dubcore data-directory and therefore used : stw=@/var/opt/ABCD/bases/dubcore/data/dubcore.stw

- since the database is identified as being Unicode, the special actab and uctab tables for UTF8 were located and found and also used :

  uctab: /var/opt/ABCD/bases/isisuctab_utf8.tab

  actab: /var/opt/ABCD/bases/isisactab_utf8.tab

- The full command used to generate the Inverted File is given, so the preceding elements can be recognized again in this command :

  /opt/ABCD/www/cgi-bin/utf8/bigisis/mx /var/opt/ABCD/bases/dubcore/data/dubcore fst=@/var/opt/ABCD/bases/dubcore/data/fulltext.fst uctab=/var/opt/ABCD/bases/isisuctab_utf8.tab actab=/var/opt/ABCD/bases/isisactab_utf8.tab stw=@/var/opt/ABCD/bases/dubcore/data/dubcore.stw fullinv/m=/var/opt/ABCD/bases/dubcore/data/dubcore -all now tell=100

  ### Note

  The FST used in this command is not the default one 'dubcore.fst' but the special one which loads the HTML-text files for inclusion into the Inverted File, allowing 'full-text' searches.

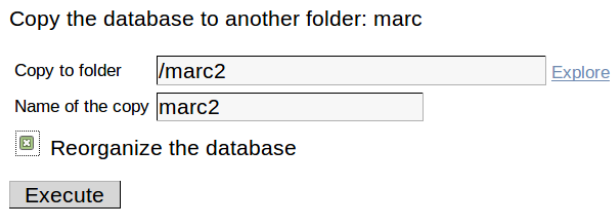
For very large databases it is advised to run this process only when necessary and preferably run the same command on the server itself, securing the database to be locked because the process might take long enough to disturb users.

7.1.2. Copy the database to another folder

This utility will copy all the files of the active database into a named other folder, e.g. as a backup of the database.

One extra option here is to 'reorganize' the database, which means logically deleted MFN's (records) will be left out in the copy to make it a 'clean' or compacted database.

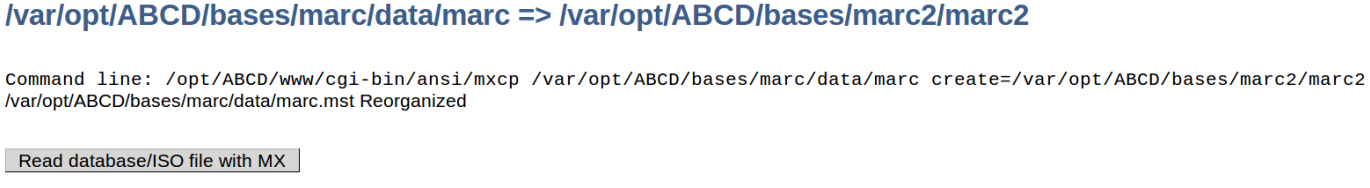
The utility's dialog only asks to identify the folder with the explore-function (allowing to create a new database-folder if it does not exist), the name for the copy and whether or not to 'reorganize' the database while copying :



After clicking on 'Execute' the next screen will ask to confirm for continuation and then the last screen will - if everything worked out well - confirm the copy, show the executed command and offer the option to read the database contents as a final check - which is actually running the next utility described under here :

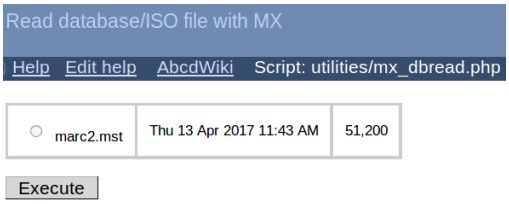
Note

This procedure will NOT add the database in the list of available databases, as opposed to the 'create new database' option which also allows to copy a database.



7.1.3. Read database/ISO file with mx

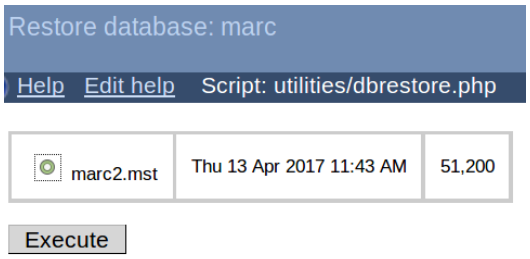
This utility is a simple one : it allows to view the contents of a database quickly, e.g. before using the database for other purposes (import e.g.) in ABCD. The CISIS-utility 'mx' is used to this end. The dialog will invite to identify the database to be viewed with the 'explore'-function to select a folder. All 'MST' files of the selected folder will be shown so as to select the one to be printed on the screen. Actual printing is by batches of 20 records with the option, at the bottom of the records listed, to continue to other parts of the database.



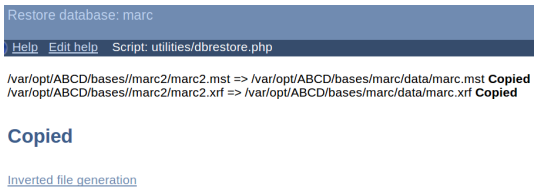
7.1.4. Restore database

Restoring a database means to load a database with the records from a copy of the master-file (.MST), e.g. after such copy has been created with the utility 'copy to another folder' as a backup. In that case 'restore' means to restore a backup. If the backup MST was a 'reorganized' one, the resulting restored database will be a clean copy.

The dialog is again very simple : using the 'explore'-function select the .MST file to be used as 'origin' and after clicking on 'execute' that master file will be loaded over the currently active database. So be careful and make sure you have the correct database as the active one.



After the restoring the dialog will also allow to re-generate the Inverted File (= re-indexing the database).

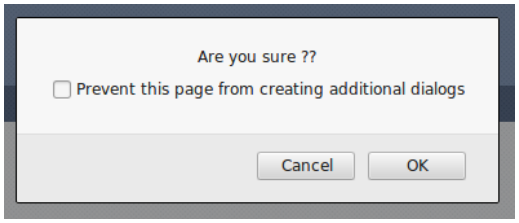


7.1.5. Initialize database

Initializing a database re-sets the existing database to 0 records without changing anything in the structure. So it is equivalent to making the database empty and literally 'restart from zero'. Since this is a non-reversible action a confirmation will need to be given while showing the actual number of records, even twice to avoid doing it by accident.

7.1.6. Delete database

With this utility the active database will be deleted and no longer be available (removed from the list of available databases). Since this is a non-reversible action it will ask for confirmation followed by a 'confirmation of the confirmation' dialog to avoid any accidental action because 'canceling' the action is still possible at this stage :



7.1.7. Protect database from initialization or deletion

In this utility the active database can be either protected or 'unprotected' from deleting or initializing, in order to secure a really important database.

As a consequence of protecting a database, trying to initialize it will result in the following error message :

Script: dbadmin/administrar_ex.php

Database protected. Request not executed

7.1.8. Unlock database

In certain uncontrollable conditions, e.g. an unexpected power-cut, a database can end up being 'locked' at either the record-level (meaning a record was still open for editing at the time of the condition's appearance) or as a whole when e.g. the database was being fully indexed. Whereas the data-entry toolbar already provides the option to list and unlock one or more records (a range), in the case of the whole database being locked, another approach is necessary because no longer the operator can enter into this database to use the toolbar on it. In such case the database needs to be unlocked first.

- **Query: /opt/ABCD/www/cgi-bin/ansi/retag /var/opt /ABCD/bases/marc/data/marc unlock**

- **process Output:**
  process Finished OK

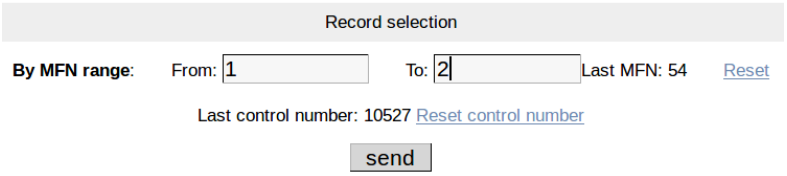
The resulting screen shown above confirms that the database was unlocked with the CISIS-utility 'retag' from the non-unicode (=ansi) version of CISIS.

7.1.9. Assign control number

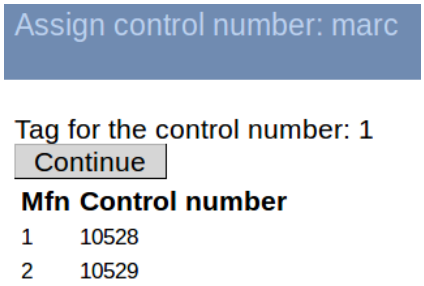
In ABCD, each database to be used in a multi-database environment, such as a library catalog in a loans system, needs a 'Control Number' field as a 'primary key' or real identifier of each MFN. The MFN itself can NOT be used to this end since it is not a fixed element : it can change e.g. when exporting/importing (or 'compacting, reorganizing) a database.

This 'Control Number' (or CN) preferably is assigned automatically (meaning : defined as 'auto-increment' in the FDT) so that the software itself controls the assignment of control-numbers, based on the file 'control_number.cn' in the data-directory of the database which contains the last assigned number. In the edit-form for such a field a link is always given to allow manual assignment of the control number, overriding this default 'auto-increment' mechanism. However at times it is desirable or necessary to assign consecutive control-numbers, starting from the existing 'last assigned number', for a range of records. That is what this utility is for.

The dialog asks for the MFN-range (first and last) to be defined and allows the CN range to be reset to any other number to give full control.



After running the process the result is shown, e.g. in the case only MFN 1 and 2 were to get new CN's :



7.1.10. Create DSpace 'bridge' database and load repository into ABCD

This new function in ABCD 2.0 allows to copy the metadata and documents from an existing DSpace repository into a dedicated ABCD-database ('dcdspace', using the Dublin Core metadata-set). For this ABCD uses the DSpace REST-API in a 'web-services' approach. The DSpace community is changing a lot in this respect, mainly shifting more functionality to this REST-API, so this solution will need to be adjusted whenever DSpace has finished their process of maximizing their REST-API approach.

The automatically (i.e. by script) filled database can be made available, like any other ABCD database, in the Site and included into the ABCD Site Metasearch configuration, to allow users to search the DSpace repository along with the library catalog or any other ABCD-resource. The update of the repository in ABCD will need to be done by putting the update-script into a scheduler, e.g. to perform the update every night.

The script uses the CURL functionality, so this needs to be installed on the server. In Linux (Debian) it can be installed easily with the terminal-command :

apt-get install php7.0-curl *(or change php7.0 for php5 if your server uses PHP5.x)*

The following steps need to be followed in order to use this DSpace-bridge functionality :

- get the DSpace REST URL and test it directly in a browser to verify its accessibility;
- [only the first time] create a database 'DCDSPACE' using the existing DUBCORE database as the model; this is very easy as all DUBCORE structures will be copied into a new directory bases/dcspace and all files renamed accordingly in an automated script;
- create the extra option in the UTILITIES-menu : htdocs/central/dbadmin/menu_mantenimiento.php by inserting the following code in the function 'EnviarForma', as one extra 'switch'-option :

```
switch (Opcion){

case "DCDspace":

document.admin.base.value=base

document.admin.cipar.value=base+".par"

document.admin.action="dcdspace.php"

document.admin.target="" break;
```

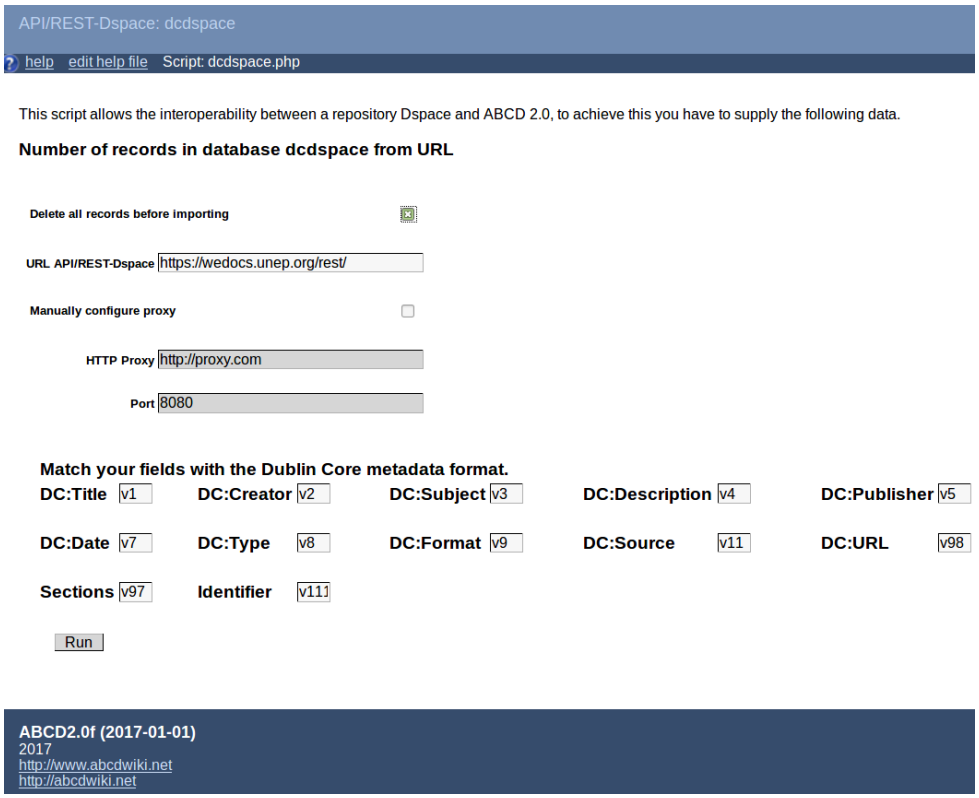
Note

This has already been done for the Utilities-menu of ABCD 2.0, so only needs to be done when building upon an older version.

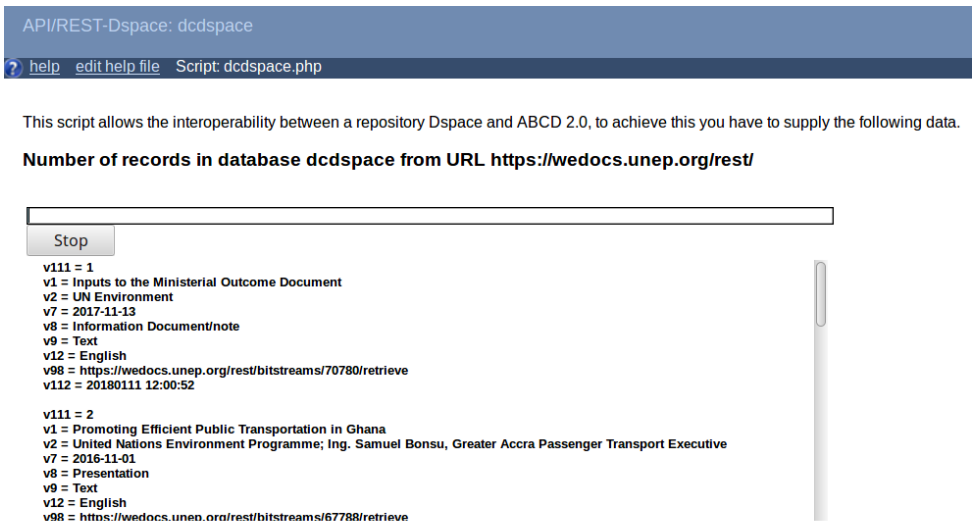
- Only when the 'dcdspace' database exists and has been selected (!!), the Utilities-menu will show the extra option :
  - Interoperability between DSpace and ABCD

The interface for this DSpace Bridge function is rather simple :

- First a screen is presented with some basic explanation, the URL defining the source (and if necessary the proxy-settings) and an option to 'clean' the database before loading new records, followed by the 'fields mapping' area, in which the Dublin Core fields can be 'mapped' to the fields of the local database, which by default is the very simple straightforward list of fields (1-15) of the Dubcore demo-database; but note the non-standard fields v97-v98-v111 to store e.g. the source-URL for the full-text PDF - note that this is not always available in the source records :

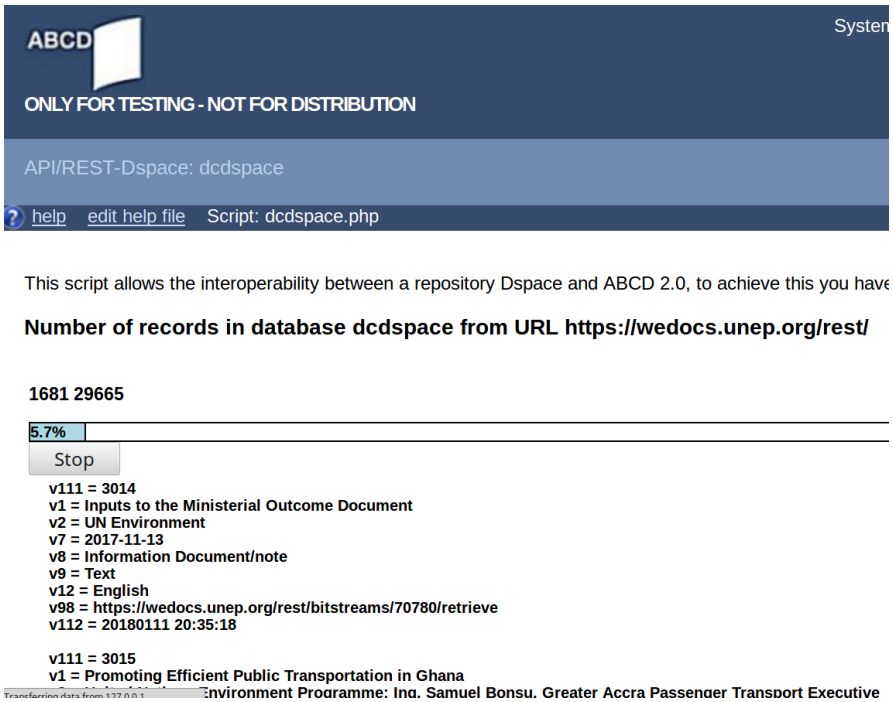


- When in the first interface screen 'Run' is clicked, the second screen will show the progress and a listing of records with their fields,



This process can be stopped by clicking 'Stop' at any time. Depending mostly on the connection and server-response speed (which can vary a lot even during the same process) this process can take some time.

- At the end of the record-loading process the interface returns to the first screen from where also the 'BACK'-button can be used to return to the main ABCD menu. Entering the DCDspace database from there will show - in the usual standard way - the records downloaded, including the hyperlink to the full-text if that was available. An example record is partially shown here :



As mentioned before : this database can further be processed like any other ABCD-database : indexed, creating new/other display formats, inclusion into the metasearch of the ABCD Site etc.

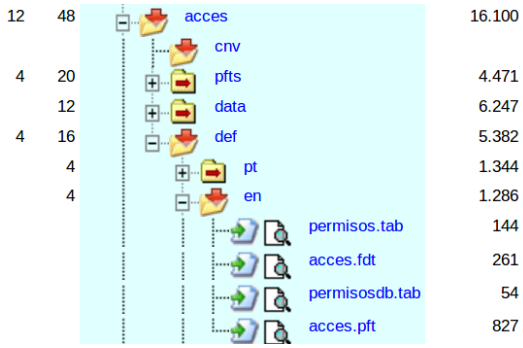
7.1.11. Explore databases directory

If in the Central configuration the option 'dirtree' is 'ON' (or '1') and the operator has administrator rights, this option will appear in the utilities menu. There are some security issues to be taken seriously here because the dirtree-script (which is not an original ABCD-script) allows text-files, such as configuration-files etc., on the server to be deleted and edited ! This is a very advanced function with many possibilities, including uploading files, so to be used very carefully.

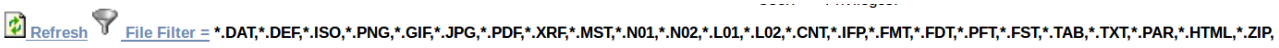
First of all, the utilities-menu entry allows either to open the whole 'bases'-directory (highest risk level...) or either the special system subdirectories par, wrk or www. Only the contents of the selected directory (or folder) will be shown.

- Explore databases directory
  - par
  - www
  - wrk

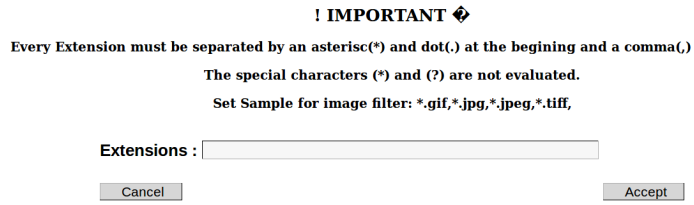
Depending on the link clicked, a file-browser will show all files which are not filtered out by their extension. The list of extensions can be edited as well within this utility. The illustration below shows a segment of the file-browser at the level of the 'acces'-database, inside the English definition directory where e.g. the file 'acces.fdt' has an icon to download it or to view it - from where then also an 'edit'-link will be given.



At the top of this file-browser there is a link to edit the extension-filter, meaning only files with the listed extensions will be shown.



Clicking on the link for 'File Filter' will give the dialog below, where the list of extensions can be re-defined.



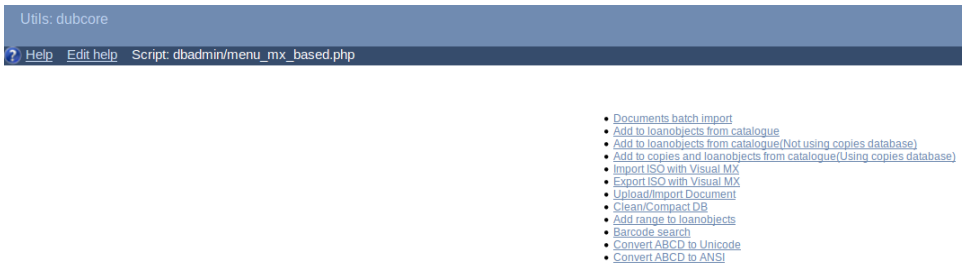
7.1.12. EXTRA UTILITIES

Since ABCD v2.0 another batch of utilities have been presented separately in an 'extra' submenu for utilities which we will discuss subsequently.

7.2. The extra utilities menu

Less frequently used or more specific utilities are listed in this sub-menu :

Figure 2.2. The exta utilities menu



Let's look a bit closer to each of these.

7.2.1. Documents batch import

For this utility we refer to the more detailed discussion on 'Digital Library' in ABCD (section 9 of Chapter 2) since this is the main script used to create databases with a 'digital library' concept to use full-text indexing and original-document linking. Together this script and the utility 'Upload/Import document' (see later in this section under 1.2.7) constitute the ABCD 'Digital Library' feature.

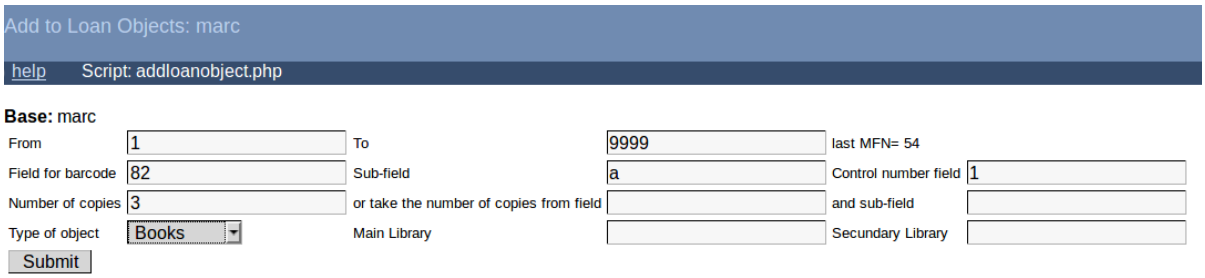
7.2.2. Add to loanobjects from catalogue

There are actually three similar utilities addressing the batch-creation of loanobjects, this one (1.2.2) and the subsequent two ones (1.2.3 and 1.2.4). They partly overlap but have different 'optimal' use cases, depending on the exact situation of the initial records from which the loanobjects need to be created.

With this utility it is possible to create a series of loanobjects directly from a catalog-database in which the necessary fields are available, with the special option to create a fixed number of loanobjects.

All three utilities assume that the information on the loanobjects, which are physical units contrary to the catalog records (which are intellectual units according to the FRBR-model), is available somewhere in the catalog records. E.g. when you create a catalog from a '.mrc' (marc21) export, like from a KOHA-library catalog, the data to identify the loanobjects are in some extra fields of the marc-records. The assumption is that the current active catalog-database still has these fields present in the records, e.g. these fields were used in the ABCD system set up for operation without copies/loanobjects - which is exactly the idea of having the copies-data in the catalog-records.

For this utility, the first screen, shown below, deals with a series of parameters to be checked by the operator of the utility. These parameters will define the range and from where to get the information to put into the ABCD Loanobjects records' fields :



- From : the first MFN from which to create a new loanobject
- To : the last MFN from which to create a new loanobject, with the last available MFN in the database shown
- Field for barcode : the field in the catalog-records which contains the unique identifier of the loanobject
- Subfield : if that identifier is in a subfield, identify the subfield identifier to be used (without the caret ^)
- Control Number field : the field which contains the Control Number of the catalog entry
- Number of copies : the maximum number of copies/loanobjects to be created from one record (which supposes each copy is identified in the catalog as an occurrence of the field)
- OR : if the number of copies are given in a field, the field-tag (and if applicable the subfield identifier) are to be identified, so the number will be taken from the value of that (sub-)field
- Type of object : the 'loanobject'-type (as defined in the circulation system) to be used for this range of loanobjects
- Main Library : the main 'location' information, mostly the name of a library or library branch
- Secondary Library : the location within the main library to more exactly specify the object's location, e.g. a shelf no. or classification code

It is very important to have made - in advance - a very good analysis of your original 'incoming' data, in order to know in which exact fields and subfields the copy-data can be found. In general there will be two different

'models' : one with the copy-data repeated in a repeatable field of one record, and one with each copy in a different record. We give examples of both situations under here.

1. copies repeated in one field : e.g. exported from Mikromarc software : in the one record shown there are 3 occurrences of v99 which has the copy-informaiton in subfields (in which exact subfields comes which exact information is to be checked in the original software !) :

```
mfn=     2
3005  "n"
3009  "a"
3017  " "
  1  "20040000000002"
 21  "00^a1858050324"
 82  "00^a330"
 96  "00^a330POW"
 99  "00^a2^bCopy 1^d1/8/2003^hMzumbe University^i1^k0042765,^qWaiting
 on reservation shelf^r0^uMzumbe University^01^1Mzumbe University^3DBA
^71^8Peter^93/9/2004^gGBP"
 99  "00^a3^bCopy 2^d1/8/2003^hMzumbe University^i2^k0042768^qAvailabl
e for loan^r0^uMzumbe University^w3/8/2004^01^1Mzumbe University^3DBA^
42^71^8Poni^98/23/2008^gGBP"
 99  "00^a5893^bCopy 3^d5/17/2004^hMzumbe University^i3^k0042767^qAvai
lable for loan^r0^uMzumbe University^w5/17/2004^01^1Mzumbe University^
3Peter^41^71^8Mwampiki^96/19/2004^gGBP"
100  "10^aPOWELL,Ray"
245  "10^aEconomics for professional and business studies"
250  "00^a2nd ed."
260  "00^aLondon^bDP Publications^c1993"
300  "00^aix, 476p^billus, index^c29cm"
522  "00^aNEW^b1/8/2003"
522  "00^aNEW^b3/31/2004^c3/11/2005"
650  "00^aeconomics"
999  "  ^c2^d2"
..
```

2. copies repeated as subsequent records, e.g. export from a KOHA-catalog : the example shows two different MFN's in which each a v952 contains the copy-data in subfields and v999 contains the CN in both ^c and ^d :

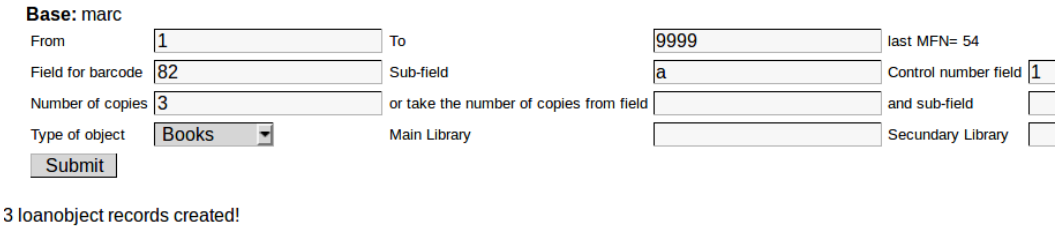
```
mfn=    14
3005  "n"
3009  "a"
3017  " "
  5  "20140903152808.0"
  8  "050514s19uu    xx           00    eng d"
 20  "  ^a0-8039-6092-1"
100  "10^aBLASE, Joseph"
245  "10^aEmpowering teachers :^bwhat successful principals do /^cby Joseph Bla
se."
260  "0 ^aCalfornia :^bCorwin Press,^c1994."
300  "  ^axxiv,166p."
520  "  ^aIncludes index"
650  " 4^aEmporeworing Teachers"
942  "  ^c UKN"
999  "  ^c122853^d122853"
952  "  ^00^10^40^6371_201200000000000_BLA^70^929849^aACL^bACL^cLending Section
^d2005-05-14^g23.28^o371.2012 BLA^p116023^r2014-09-03^w2014-09-03^y UKN"
..
mfn=    15
3005  "n"
3009  "a"
3017  " "
  5  "20140903152808.0"
  8  "050514s19uu    xx           00    eng d"
 20  "  ^a0-8039-6092-1"
100  "10^aBLASE, Joseph"
245  "10^aEmpowering teachers :^bwhat successful principals do /^cby Joseph Bla
se."
260  "0 ^aCalfornia :^bCorwin Press,^c1994."
300  "  ^axxiv,166p."
520  "  ^aIncludes index"
650  " 4^aEmporeworing Teachers"
942  "  ^c UKN"
999  "  ^c122854^d122854"
952  "  ^00^10^40^6371_201200000000000_BLA^70^929850^aACL^bACL^cLending Section
^d2005-05-14^g23.28^o371.2012 BLA^p116024^r2014-09-03^w2014-09-03^y UKN"
..
```

Note

The previous illustrations were created in a 'terminal' window with a CISIS-mx command which converts the typical '.mrc' subfield-characters to the ISIS-subfield character (a caret or ^) with a gizmo-database 'sv' and in addition specifies that the 'leader-fields' should be presented in the tag-ranger of 3000 :

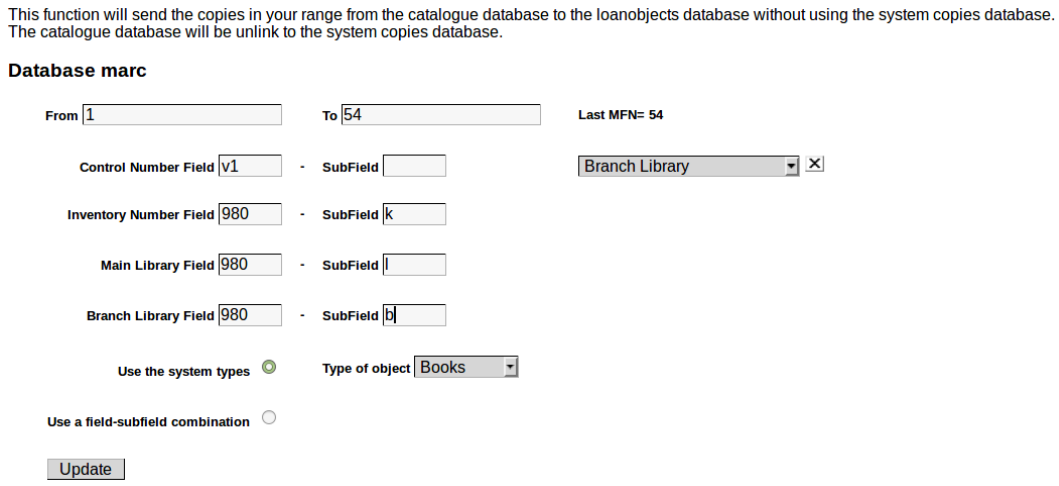
mx iso=marc=koha.mrc isotag1=3000 gizmo=sv

If everything was properly identified, the script will run over the range of MFNs of the catalog database and report the number of loanobjects-records created. In this case 3 MFN's were created since the 'number of copies' was fixed at 3 - of course a more interesting approach is when the exact number of copies is noted somewhere in the catalog-record, so no fixed number needs to be created.



7.2.3. Add to loanobjects from catalog (not using copies database)

This utility is a more general one without fixed number of loanobjects to be created, so the related 'addloanobject-copies' script is meant to create the loanobjects records stored in the catalog database without using the copies database. Let's look at the dialog to be filled in before running the process :



As can be seen, information needs to be specified on the range of MFN, the (sub-)fields for CN and Inventory No. (barcodes) and, if from the 'ADD' menu an extra value needs to be added to the loanobjects (main library, branch, tome and volume/part), for each of these also a value can be given. For the 'loanobject'-type either one of the system-defined values can be selected from the menu, or if 'use a (sub-)field combination' is chosen, the field (and optionally subfield) which defines the type of loanobject can be given by its tag.

Clicking on the 'UPDATE' button will run the process over the selected range of MFNs and report the result.

7.2.4. Add to copies and loanobjects from catalogue(Using copies database)

When a library prefers to first create all 'copies' (or acquisition/stock) records for its catalog, at some moment it needs to also create the corresponding loanobjects in ABCD, which are differently organized (as explained supra) by having all physical units (with their barcode or identifier) repeated in one field (v959) rather than having one individual record per unit as in the COPIES-database.

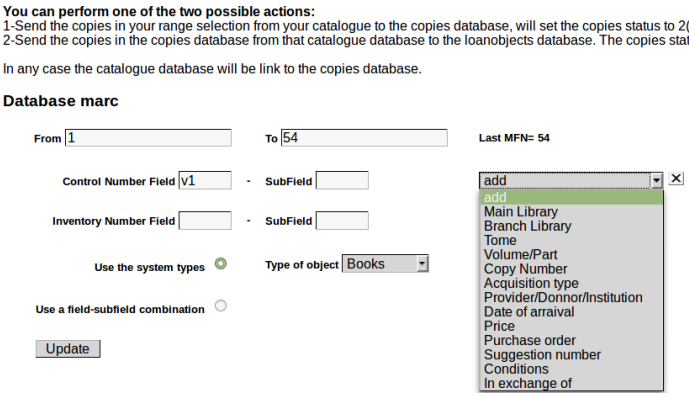
This script allows to create a batch of loanobjects from an existing copies-database, in addition to allowing to create them directly from the catalog (as in utility 1.2.2). The interface is similar to the previous one (1.2.3) but the built-in options 1 and 2 are rather different :

1. Create both copies and loanobjects records from the existing data in the catalog database : fill in the inventory number field

2. Create the loanobjects for a range of existing copies-records : don't fill in the inventory number field

The 'ADD' element of the interface now allows to additionally also insert all standard fields of the COPIES-database and define from where in the catalog-records to take the value. So this is optional and depends on the availability of such information in the catalog records.

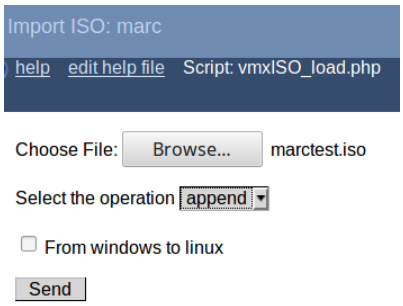
As in the previous tool, also the loanobject-type can be either defined as a fixed value selected from the system-defaults, or taken in a variable way from a field/subfield of the catalog-records. Again it is important to carefully analyze the information available in the catalog records !



This utility, as does the previous one, uses an 'mx'-command sent to the OS of the server, so it will perform its task very quickly but without client-server communication and protection, meaning that if the process takes long or hangs for some reason, no feedback will be given. So use this type of utilities carefully, i.e. after first testing with a very limited range of MFNs and checking the results, before doing the full range. However once carefully prepared and run, it can save an immense number of work-hours for the manual alternative.

7.2.5. Import ISO with mx

As a faster alternative to the ISO-file import in the Central Data-entry toolbar (utilities icon) here is a tool to quickly import large numbers of records. In view of the HTTP-protocol interaction in between server and client, the dataentry toolbar option needs to do this in limited batches (of 200 records e.g.), here there is no such limit but one has to ensure the server and database is ready for the job. Taking into account the fact that mostly this process just takes seconds, even for large databases, it still can be a very handy tool.

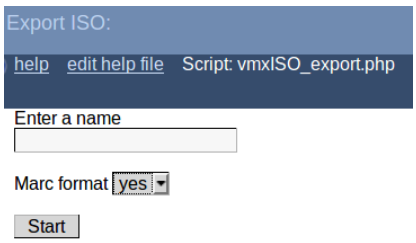


The utility requires to select (with the default ABCD file-browser) the ISO-file to serve as input, to specify if the incoming records should overwrite the existing database (create) or be appended at the end of existing records, and finally whether or not the ISO-records need converted from Windows to Linux. The ISO2709 format is a text-format and in Windows uses different end-of-line markers.

7.2.6. Export ISO with mx

This utility is very simple : a command will be created and run on the server to export your database to ISO2709 format. This utility simply takes the active database as a whole.

The only additional option here is 'use MARC format yes/no'. If yes is selected, the MARC leader will be exported to the 3000-range of tags so as to be included in the ISO-records.



7.2.7. Upload/import document

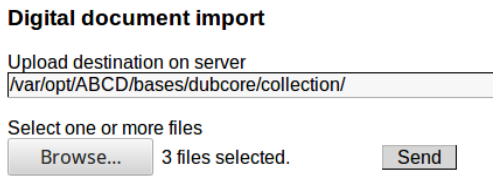
When your database has a 'Digital Library' style, e..g the 'DUBCORE' demo-database, documents can be uploaded and linked to the automatically created metadata-records. This principle and technique is described in detail in the section on ABCD Digital Library (section 9).

In fact there are 2 utilities used to implement this 'Digital Library' feature in ABCD :

1. a batch-tool to convert the typical document-formats (.docx, .pdf, .odt etc.) to HTML-text-format and load all words of the text into the Inverted File; see the discussion in the Digital Library section (no. 9) of this manual. With this tool a digital library database or repository can be created from scratch, but also a batch of new incoming documents can be added to the existing database. New records will be created with both metadata, full text and the link to the original document;

2. this 'interactive' tool which allows to select one or more documents individually to be added into existing records or at the end of the database. If selected from the Central Data-entry Toolbar (the icon appears as part of the toolbar  if the database has the parameter 'IMPORTPDF' set to 'Y'(es)) this utility will add the document to the existing active record, otherwise the new record(s) will be appended to the database.

The interface runs in several steps, in fact only adding one step for selection of documents to the batch-import tool :



1. First the defined 'collection' directory for the active database will bes shown and the documents to be uploaded will need to be identified with the default ABCD-filebrowser (more than one file can be selected). The COLLECTION-parameter in dr_path.def defines the destination directory.

2. After having sent the documents to the server (3 selected in the illustration above), the 2nd step will ask to 'map' the metadata and text-fields, exactly as described in the discussion of the batch-import tool above.

3. Finally - again exactly as with the batch-import tool - the process progress will be shown with the final information about the result.

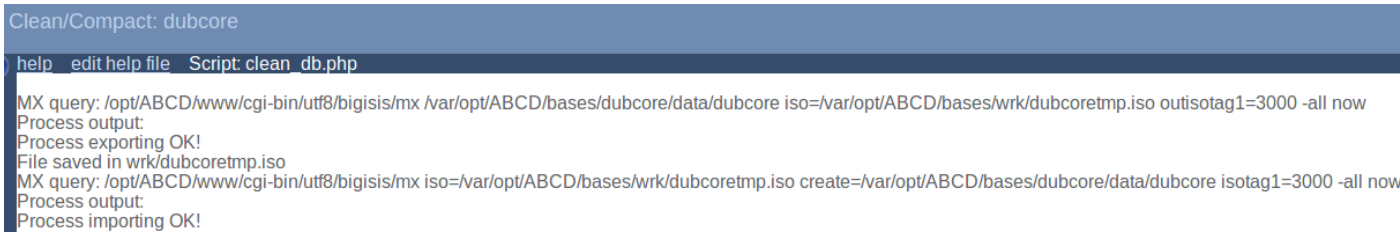
7.2.8. Clean/compact database

Cleaning or compacting a database is the process which gets rid of logically deleted records (or MFNs in ISIS). Whenever a record is edited, the reference in the XRF-file to the old version is deactivated and a new one added to the new version of the record, so after some time lots of records remain in the MST without actually being used. To recover this space, the tool will quickly export and re-import the records.

This is the simplest possible tool with no interface at all, except for the result screen which shows the two commands executed on the active database on the server :

1. the export command

2. the command to re-import the export created in step 1.

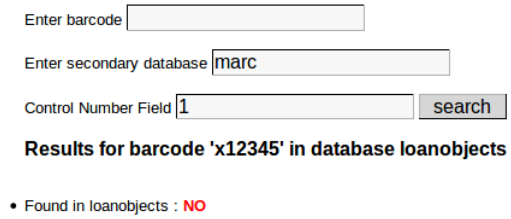


Note that the temporary export-file is created in the 'wrk' database-directory as we need to ensure there are writing-rights for that directory.

7.2.9. Barcode check

This tool is supposed to run from the LOANOBJECTS database in order to check whether the barcodes of the loanobjects-records 1) exist in the copies database and 2) the related title exists in the catalog. So this option will only show up when the active database is named 'loanobjects'.

The idea is to simply scan (or otherwise 'enter') the barcode of a book in the first box and - given the name of the catalog-database and the CN-field - click on 'SEARCH' to see if the result appears in GREEN (=OK) or in RED (not OK). So this tool can assist a 'stocktaking' effort but in an interactive way : librarians walking along the shelves and checking each individual book and correcting when e.g. a copy or title is missing for an existing object in either the loanobjects, copy or catalog-databases. In this sense this is different from another procedure for 'batch' stock-taking which is described in another section of this manual. In that procedure reports will be produced for all existing loanobjects to see if they exist on the shelves (as noted in a database with barcodes) or - vice versa - a report listing all inventoried barcodes (in a very simple dedicated database) with information on whether yes or not this barcode was found in the loanobjects database.



The illustration above shows the result for a barcode 'x12345' which was NOT found in the LOANOBJECTS database. Since it was not found there it can also not be located into the COPIES database or MARC-catalog. Once created in LOANOBJECTS, the check can be repeated to see if a corresponding record exists in the COPIES and MARC database.

7.2.10. Convert ABCD to Unicode

To assist migration from ABCD1.x to ABCD2.x with the Unicode capabilities, two additional utilities are available, to convert from ANSI (the 'old' classic ABCD) to UTF8 and vice versa. The interface is the same for both scripts but the 'direction' of the operation is the opposite.

Beware : this conversion does NOT convert databases, only the text-files of the system. So the first option 'Convert base marc' of the illustration above means to convert the text-files of the database, i.e. the files with extenstions FDT, FST, PFT, STW, DEF and TAB in the 'def' and 'pfts' subdirectories of the database. This operation will allow to use non-Latin characters in e.g. print-formats or field-names, which would otherwise completely disturb the display if kept in ANSI.

A separate option is available for doing this specifically for the 'operators with passwords' database 'acces'.

The list of extensions allows to select/deselect specific file types. In most cases it is best to leave them all selected, as there should be specific reasons to leave out one or more from this process.

When selecting the 'htdocs' folder the whole series of text-files with the selected extensions will be changed from ANSI to UTF8. For instance help-texts in iAH in HTML-format can be changed to UTF8 to allow non-Latin content etc. An important folder here is probably the htdocs/bases/documentacion folder with help-text ('documentation') for the databases. They can then also be written in the local alphabets of the users.

7.2.11. Convert ABCD to ANSI

This utility 'undoes' the previous action and is indeed converting all UTF8-marked text-files, with the selected extensions for the selected system-part (bases-folder for the active database or the whole htdocs-directory) to ANSI. This procedure will only be needed in very rare cases and assumes all non-Latin contents have been removed (or were absent in the first place).

The tool has the same interface as the previous one and results in the same detailed list of the file-names of files which were not-converted, which were converted with unknown previous encoding and converted with known previous coding.

Be careful, e.g. lots of .HTML files exist in htdocs (e.g. plugins for Secs-Web, so it can take a while.

8. ABCD Thesaurus

8.1. Thesaurus

In ABCD a thesaurus can be used as a 'guided authority' list of terms at two different stages : while assigning descriptors during data-entry and while defining keywords for searching in a database.

The thesaurus itself is a database (remember in ABCD 'everything is a database') with a specific structure of fields which cover the main standard-thesaurus elements of identifier, descriptor, scope note, use, used-for, broader/narrower term and related terms as well as 'non-descriptor' (a term to be avoided and replaced by a 'preferred' one). For more background information on this standard 'ISO 25964' check e.g. the Wikipedia-page at the URL https://en.wikipedia.org/wiki/ISO_25964.

8.1.1. Configuration of a thesaurus database for ABCD Central

In the file 'dr_path.def' of the associated database (e.g. your MARC-catalog) the following parameters have to be added :

- Thesaurus = , followed by the name of a thesaurus database, e.g. 'agrovoc' (without path information as that will be obtained, as for any database in ABCD, from its par-file in the bases/par directory;
- prefix_search_tesaurus = , followed by a prefix which is used to index the 'descriptors' field in your database, e.g. 'MA_' in the demo MARC-database for field 650. This prefix will be added by the software in front of the selected term in order to search it in the (catalog-)database.

In addition a file '*dbn*.dat' (where '*dbn*' is the name of the thesaurus database, e.g. 'agrovoc') needs to be created in the active language of the 'def'-directory of the thesaurus-database, e.g. in the case of English agrovoc : bases/agrovoc/def/en/agrovoc.dat. In that file 4 parameters need to be defined :

1. Alpha_prefix = : Prefix to use to retrieve the alphabetical list of thesaurus terms, as indexed in the FST
2. Perm_prefix = : Prefix to use to form permuted listing of terms. The prefix PER_ must be used and the line of the fst in which it is used must be indexed by technique 8 (each word of the term with addition of a prefix
3. Alpha_pft = : Format to use to display the term in the alphabetical or permuted list
4. Display = : Name of the format (without the .pft extension) to format the thesaurus record and display it in the thesaurus terms tab in full detail with links.

An example such configuration file looks like the following :

Alpha_prefix = TE

Perm_prefix = PER_

Alpha_pft = v8

Display = tab

The 'Display = ' PFT, in this example therefore named 'tab.pft', will require some additional elaboration in order to format the record in a useful and attractive way. Here is an example of such PFT, using the fields :

v8 = Base term

v12 = Use or 'see' (US)

v13 = Used for (UF)

v14 = Scope note (SN)

v16 = Broader term (BT)

v17 = Narrower term (NT)

v18 = Related term (RT)

v30 = Notes

`<a href='javascript:Search("`v8`")'> <img src = .. / dataentry / img / search.gif border = 0> </ a>` "<Strong>" V8 "</ Strong>" V45 / "
" v32, /

'<Table>',

If p (v12) then '<Tr> <td valign = top> Use </ td> <td>'

(If p (v12) then `<a href='javascript:Show("`v12`")'>` V12 " '/ fi), '</ Td>' Fi,

If p (v13) then '<Tr> <td valign = top> UF </ td> <td>'

(If p (v13) then `<a href='javascript:Show("`v16`")'>` V13 " '/ fi), '</ Td>' / Fi, <Td> </ td> </ td> </ td>

If p (v16) then '<Tr> <td valign = top> BT </ td> <td>' (If p (v16) then `<a href='javascript:Show("`v16`")'>` V16 '/'), '</ Td>' / Fi,

If p (v17) then '<Tr> <td valign = top> NT </ td> <td>' (If p (v17) then `<a href='javascript:Show("`v17`")'>` V17 " '/ fi), '</ Td>' / Fi,

If p (v18) then '<Tr> <td valign = top> RT </ td> <td>' (If p (v18) then `<a href='javascript:Show("`v18`")'>` V18 " '/ fi), '</ Td>' / Fi,

If p (v30) then '<Tr> <td valign = top> See also </ td> <td>' (If p (v30) then `<a href='javascript:Show("`v30`")'`` V30 '/'), '</ Td>' / Fi,

'</ Table>' '<P>' /

The initial statement `<a href='javascript:Search("`v8`")'> <img src = .. / dataentry / img / search.gif border = 0> </a>` will allow to launch a search when clicking on the term.

Note

the use of the special quotes ` in this statement to still allow the 'normal' quotes ' and " without disturbing the format

The statements calling a javascript 'Show', generalized as follows :

(If p (vxx) then `<a href='javascript:Show("`vxx`")'`` Vxx " '/ fi),

allows to use the terms of the relationships to navigate through the thesaurus by clicking on them ('Vxx' being the related field).

Note

The javascript codes for Search and Show are already inserted in the script tesaurus / show.php in charge of showing the full records of the thesaurus

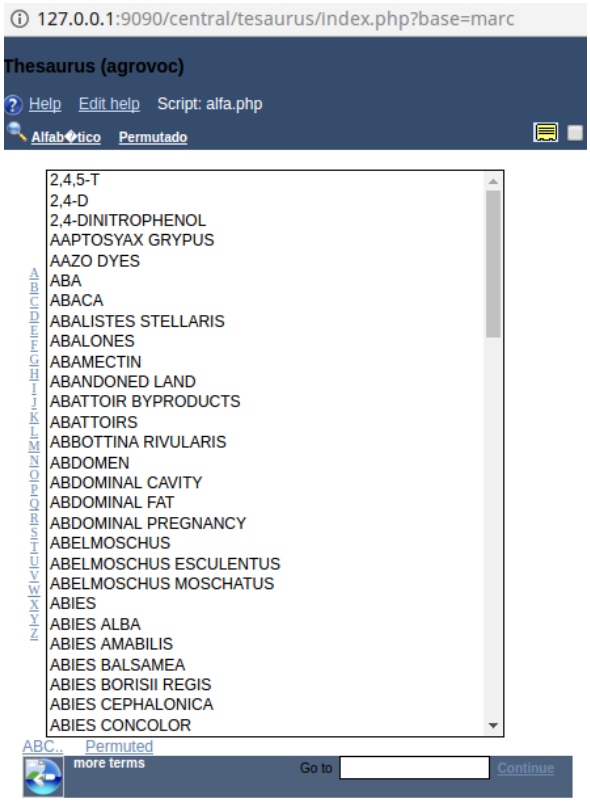
Finally, the descriptors field in the catalog FDT and FST has to be defined as a thesaurus-supported field as follows :

- FDT : in the 'descriptors' field (e.g. 650 in MARC) define 'thesaurus' as the picklist database-type and add the prefix used (e.g. MA_); put the field-tag in both 'list as' and 'extract as' columns, e.g. v650 | v650
- FST : if V8 contains the main term, at least two entries need to be present in the FST of the thesaurus database : 8 0 "TE_" v8 and 8 8 '| PER_ |' V8 for resp. the base-term listing and the permuted listing.

8.1.2. The use of a thesaurus in data-entry

When the previous configuration with its 2 parameters is found to be present, the Central toolbar will show the Thesaurus icon 

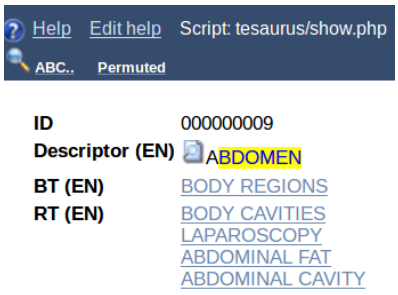
Clicking on that icon will open a new sub-window with the initial listing of your thesaurus terms, e.g. in the case of agrovoc :



The listing shown is the 'alphabetical' one ('ABC...') with all Latin alphabet characters available for quick repositioning in the thesaurus. Also the 'go to' box at the bottom can be used to enter a few characters and then immediately jump to the closest term by clicking on 'continue'.

An alternative way of showing this listing is using the 'permuted' listing - see the two options 'ABC...' and 'Permuted' at the bottom of the screen - which will show all terms starting with the selected character but with its context (other included words) before or after the word itself. In the illustration below we have opted for the 'D' character as the starting point and the list shows all terms with words starting with 'D' also if it is not the first words in the multi-word term :

Finally, when the tick-box at the upper-right corner is indeed ticked (active), selecting a term will first show the selected term with its relations, i.e. the full record formatted according to a pre-defined PFT (which is explained above in the configuration).



Whenever the 'thesaurus consultation' is finished (by navigating into it or simply selecting a term), the finally selected term will be inserted into the data-entry form (field), however only at the field level [the sub-field level still needs to be implemented].

8.1.3. The use of a thesaurus in searching

In fact exactly the same mechanisms of opening and consulting the thesaurus as above for data-entry apply for searching. After finally a term has been selected, it will be used as the current search-term and the resulting records will be shown.

Note that such search will only result in real records when they were entered using thesaurus-terms at data-entry stage. When a database is searched which was created without a thesaurus, only by co-incidence the same terms from the thesaurus might have been used as non-authoritatively assigned descriptors.

9. The Online Document Delivery Service (ODDS)

This service is added in version ABCD 2.0 to facilitate the organisation of a document delivery service for electronic documents. The idea is that end-users, from the link on the ABCD Site (now part of the demo Site), bring in the bibliographical data they know in a form as a request formulation. The form, when submitted, becomes a record in the ODDS database which is attended by a special library officer : librarians, having often better access and knowledge about how to locate electronic documents, identify the documents, put them on a library server and send out - semi-automated - an e-mail to the requesting end-user notifying her/him of the URL of the document and the time-span to download it.

We discuss this process in this section but will have to start with some few configuration issues.

9.1. Configuration of ODDS

The ODDS module is integrated into ABCD as from version 2.0, but can also be installed as an add-on in an existing ABCD-installation by unzipping the ODDS.zip archive and by doing so adding new files but also modifying existing files in your system. We explain these here.

9.1.1. new files

The following files are newly installed into ABCD for the purpose of the ODDS-module :

- the directory htdocs/central/odds, which contain the main scripts and files needed;
- htdocs/central/css/estilo_odds.css : a CSS style-sheet used in ODDS, needs to be available in the Central style-sheets directory
- /lang/odds.tab and bases/lang/odds_help_info.tab : messages and help text to be copied into the related language-directory of your ABCD bases/lang directory.
- /odds datase as a folder in your ABCD's database-directory; this is the database to store the requests. The database may have some test records which should be deleted ('initialize' database) before starting to use it locally
- /par/odds.par file to be copied to the directory bases/par and edited if necessary to indicate the correct path to the ODDS database for your ABCD installation

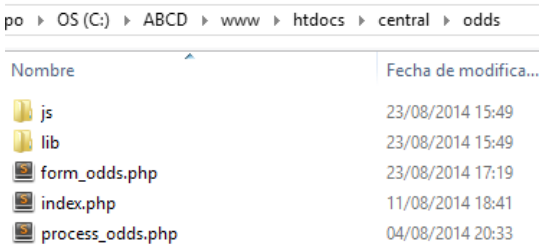
9.1.2. modified files

The following files exist in ABCD but have new contents added on behalf of ODDS :

- Central / iah / ver_documento.php
- Central / iah / configure.php
- Central / iah / ver_documento_ex.php
- Central / iah / view_document_ex-ODDS.php
- Central / iah / ver_documento_ex-WEBEX.php
- Iah / scripts / <lang> /ahhead.pft
- Iah / scripts / <lang> /ahfoot.pft

9.1.3. Structure of the ODDS-directory

After installation of ODDS, the following structure will exist in your ABCD Central odds-directory :

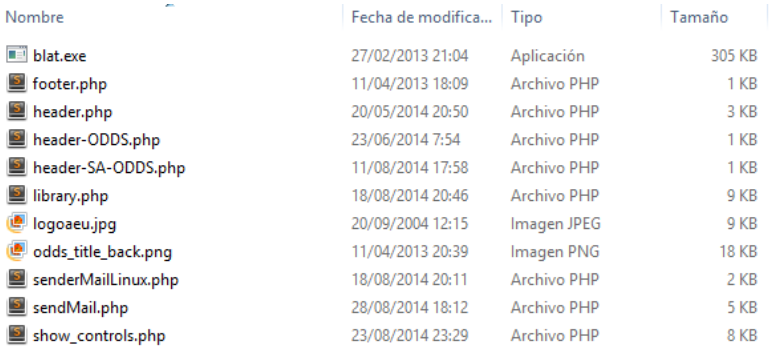


in which the following files side :

- Form_odds.php : the ODSS home-page which contains the code to display the main ODDS request-form.
- Process_odds.php : The data loaded in the form of the previous item are sent to this script for validation, processing and storage of the data in the ODDS database.

- index.php : test examples for invoking the form with parameters and without parameters.

The **directory odds/lib/** contains the following files :



Nombre	Fecha de modifica...	Tipo	Tamaño
blat.exe	27/02/2013 21:04	Aplicación	305 KB
footer.php	11/04/2013 18:09	Archivo PHP	1 KB
header.php	20/05/2014 20:50	Archivo PHP	3 KB
header-ODDS.php	23/06/2014 7:54	Archivo PHP	1 KB
header-SA-ODDS.php	11/08/2014 17:58	Archivo PHP	1 KB
library.php	18/08/2014 20:46	Archivo PHP	9 KB
logoaeu.jpg	20/09/2004 12:15	Imagen JPEG	9 KB
odds_title_back.png	11/04/2013 20:39	Imagen PNG	18 KB
senderMailLinux.php	18/08/2014 20:11	Archivo PHP	2 KB
sendMail.php	28/08/2014 18:12	Archivo PHP	5 KB
show_controls.php	23/08/2014 23:29	Archivo PHP	8 KB

where these files implement various functionalities, needed by the ODDS module, more specifically :

- Blat.exe Binary used to send emails (from ABCD) to the Windows operating system.
- Footer.php Foot of the pages included in the ODDS module.
- Header.php Head of the pages included in the ODDS module.
- Header-ODDS.php Head for the box in which user validation is requested (from iAH or Site).
- Header-SA-ODDS.php Head for the frame in which user validation is requested (from Alert Service).
- Library.php Set of functions used to load messaging dynamically. The text of the messages are read from files with an extension tab. For details please see below.
- Logo.jpg Logo sent in the head of institutional emails.
- Odds_title_back.png Background image for the title of the form.
- SendMaiLinuxl.php Implements the functionality of sending emails (from ABCD) to the Linux operating system.
- SendMail.php Used to centralize the sending of emails (regardless of the operating system). Validate the data required to make the sending and upload the text templates for the subject and the body of the mail according to the email to be sent (satisfied or canceled order).
- SendMail.php Implements the sending of emails from ABCD (via Ajax). Validate the data required to make the shipment and upload the texts according to the email to be sent (order satisfied or canceled).
  In order to send mails, the ODDS module is provided with the getOutput function implemented in the JavaScript language and located in the central file / odds / js / lib.js To be able to send (using a call to Ajax built by the getOutput function) must be invoked using the following parameters) if not used send empty strings) :

  ```
  getOutput(email,email_proxy,date,name,status,uploadFiles,notes,title)
  ```

  where :

  - Email: is the email of the recipient (can be sent to several separating the addresses of mails with a comma).
  - Email_proxy: if there is a proxy to which you want to send mail.
  - Date: date of application.
  - Name: name of the applicant.
  - Status: status of the request, used if the request corresponds to a satisfied request (2) or a cancellation (3).
  - UploadFiles: uploaded files separated by the pipe character.

  - Notes: additional notes.
  - Title: title of the work being served

- Show_controls.php : Implements the load of the controls dynamically according to the option chosen in the "Bibliographic Level" combo. In other words, it dynamically determines what data will be requested at the time of making the bibliographic request according to the type of bibliography chosen. To load the controls, this file is invoked via Ajax, thus avoiding reloading the page.

  In the base / odds / def / <lang> /odds_show_controls.tab file, the blocks of controls to be set corresponding to the option chosen in the "Bibliographic Level" combo box are configured. That is, each block of controls maps with a combo option "Bibliographic Level". For example, the control block started with "as" ("as" ignores serial analytics) is loaded when the "magazine article" entry is selected in the combo. This link between the value "as" and the entry of the "magazine article" combo is done in the file levelbiblio.tab (under bases / odds / def / <lang> /).

  In show_controls.php, each line in each of the blocks specifies which field to display and with what characteristics :

  ```
  <tagXXX> | <label_to_show> | <input_type> | <length> | <validate_method_1 validate_method_2>
  ```

  where :

  - <tagXXX>: XXX is the field number defined in the FDT file
  - <label_to_show>: Text to be displayed accompanying the input for data entry
  - <input_type>: type of input for data input, for now we only have two possible values: text or textarea
  - <length>: length of the input (only applies to input of type text)
  - <validate_methods> Methods that apply to validate the data field. The methods must be defined in htdocs / central / odds / js / JV.js

-

The **directory odds/js/** contains the following files as javaScript functions :

- jquey.min.js General javaScript ibrary jquery
- JV.js Validations and messages for all elements of the forms

  This file provides a list of validation methods provided by ODDS. All validations can be used by simply including the name of the validation function in bases / odds / def / <lang> /odds_show_controls.tab. In case you want to add or modify validations, as well as the messages they deployed, you must modify this file.

- Lib.js Auxiliary functions for sending emails and calling Ajax to sendMail.php
- odds.js Validations and call to show_controls.php

The **directory htdocs/central/iah**

The files listed below should be overwritten should they already exist. These files implement the functionality to request identification from the user. Configure.php

- View_document.php
- View_document_ex.php
- View_document_ex-ODDS.php
- View_document_ex-WEBEX.php

Clarification The parameterization of this functionality has not yet been completed; Therefore, some code fragments must be modified to be able to use it. See below.

Text and configuratoin of ODDS summarized :

Table 2.3.

FUNCTION	RELATED FILES
Text and subject of the e-mails that are sent for the notification of orders ODDS (accessible from the ABCD)	In bases/odds/def/<lang>/ : odds_success_mail_single_file.tab odds_success_mail_multiple_file.tab odds_cancel_mail.tab
Miscellaneous texts: Below the title, on the top bar Notification text of success or failure of the order (after completing the form), section REQUEST_MESSAGES Labels of the fixed controls, that is, they do not vary according to the "Bibliographic Level" chosen (for example, ID, name, email, etc.)	bases/lang/<lang>/odds.tab
Configuration of optional controls (those made visible according to the option chosen in the combo "Level Bibliographic")	bases/odds/def/<lang>/odds_show_controls.tab
Text of the help box, on the right, in the main form	bases/lang/<lang>/odds_help_info.tab
ABCD Text for ODDS Notification E-mail Buttons (Order Completed and Order Canceled)	bases/odds/pfts/<lang>/odds.pft
Options in combos (whether or not visible) "source" "Bibliographic level" "category"	bases/odds/def/<lang>/source.tab | bases/odds/def/<lang>/nivelbiblio.tab | bases/odds/def/<lang>/categoria.tab

9.2. The workflow of ODDS

The workflow or 'how to use' of ODDS in ABCD pertains to 2 parts : the request by the end-user and the processing by the 'information broker' or librarian. The idea is that an end-user creates a request to get an electronic document which is not already available and which cannot easily be retrieved directly, e.g. because of licensing issues. A librarian would offer the service of locating the document, creating a local copy of it (given that the library is authorized to access the document) and alert the end-user by e-mail about the temporary availability of the document.

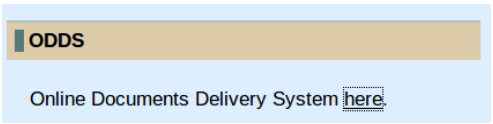
9.2.1. The request created from ABCD Site (or iAH)

The process starts when an end-user creates the request through a form, to which a link exists in ABCD, mostly from the ABCD Site (as that is exactly the idea of the Site). The ABCD 2.0 demo site already has such a link in the 3rd column. A component of type 'XHTML' was created there, and in the component (as can be verified using the ABCD Site Admin or CMS) the following code was entered, in this case referring to the PHP-script for the form on a 'localhost' server, obviously needing adjustments for other server-URL's :

```
<p>Online Documents Delivery System <a target="blank" href="http://localhost:9090/central/odds/form_odds.
```

This is nothing else but a simple HTML link to the ABCD Central script : http://localhost:9090/central/odds/form_odds.php

and showing up in the ABCD Site as follows (example taken from the default demo ABCD Site) :

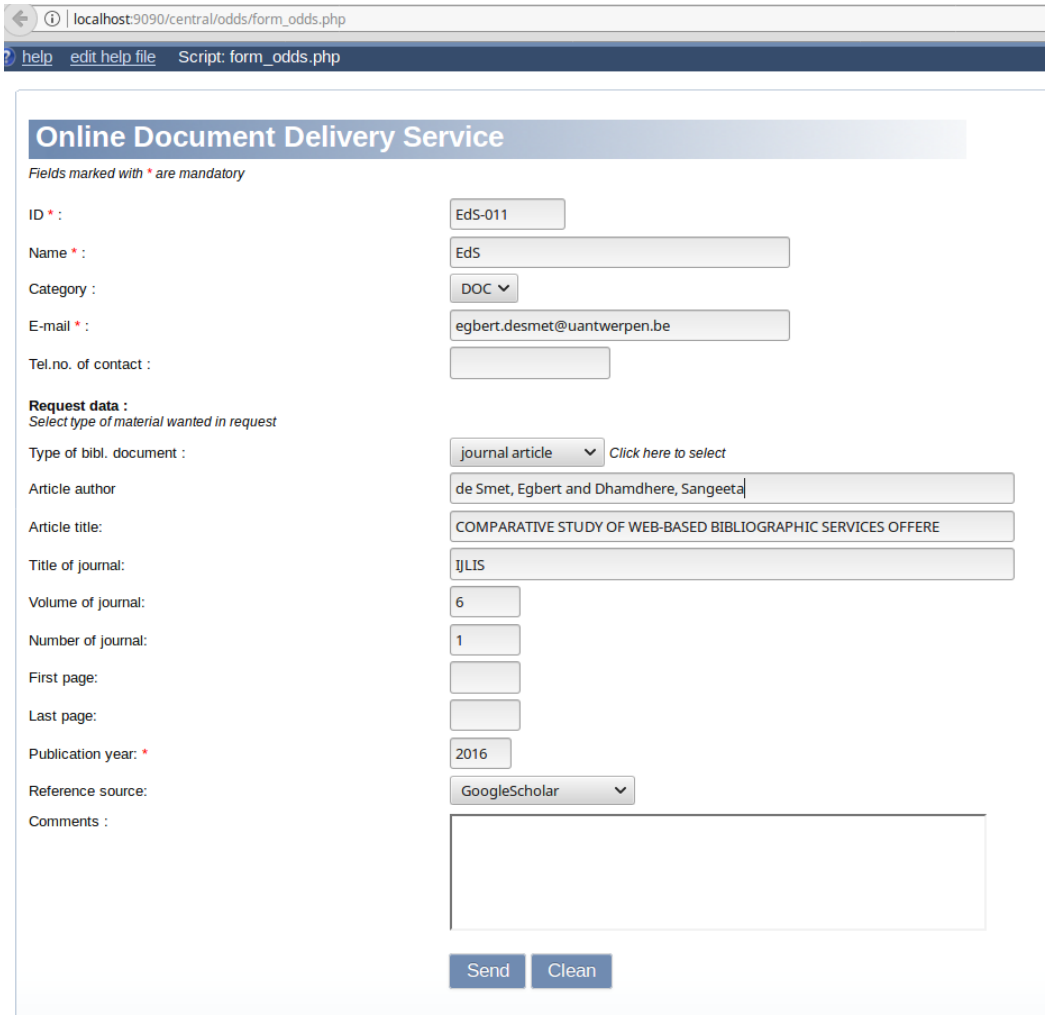


Creating the same link in the iAH OPAC is also possible, but requires quite some more skills in locating the right spot for such link. Most links (e.g. the 'shortcut'-links) in iAH are repeated for every single record of a result-set, which does not make sense here for a document request.

Note

The ODDS module is not a 'document request' feature for e.g. creating photocopies and having it physically sent to the requester. Such feature are however also quite possible in ABCD iAH by e.g. creating an ISIS PFT as a 'shortcut' which sends a request to the library's e-mail address requesting such copy. The initial developer of iAH (BIREME/PAHO) uses/used this service a lot as it was/is their main service.

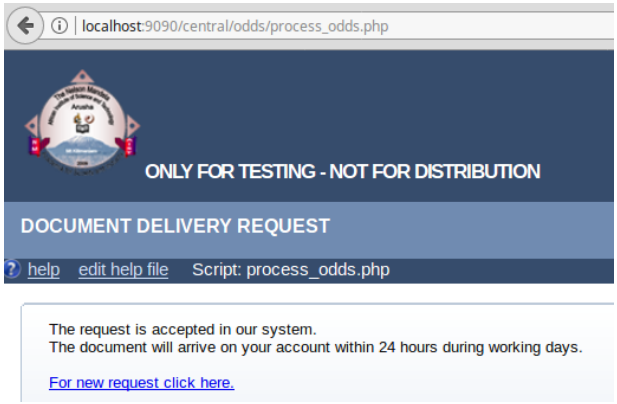
When clicking on the link given, a page will open in the browser with the request-form, which looks, with some demo-data already filled in, as follows :



One important observation here is that by selecting different 'type of documents' (here a 'journal article' is selected), the form will display and gather different fields to identify the bibliographic data of the document.

The picklists for this form are resp. 'categoria.tab' (for request categories, e.g. by library branch), 'nivelbiblio.tab' (for the bibliographic level of the requested document, changing the fields involved in the form), 'source.tab' (where did the user learn about the document), 'status.tab' (status of the request process), 'tipoliteratura.tab' (type of literature, e.g. book, article...) and finally 'topicarea.tab' (topics). All of these can be edited directly (n the directory bases/odds/def/*lang*/) or from the worksheet-editor as 'picklists' for the ODDS-database.

The requester, after filling in the mandatory and as many as possible the available fields, clicks on 'Send' and will then receive a confirmation :

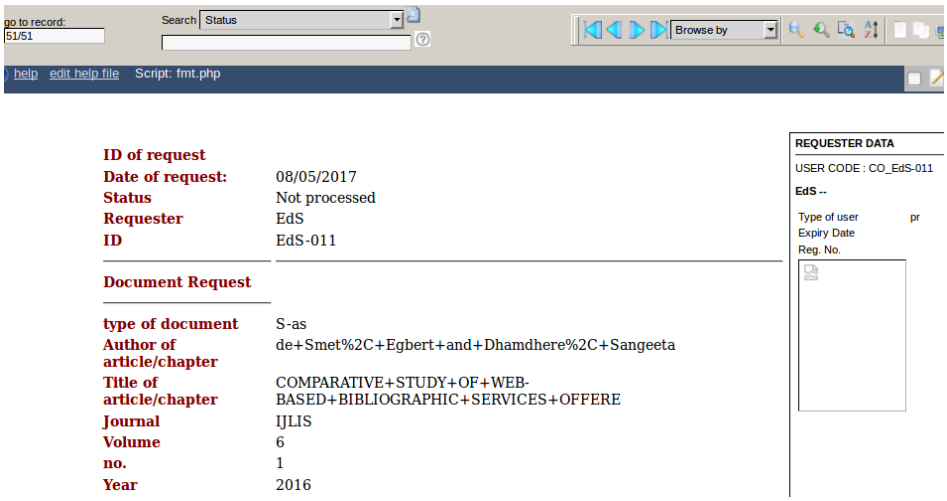


This ends the first phase, the 'end-user' request. The user now has to wait for an e-mail to be sent by the ODDS-responsible librarian about the availability of the document.

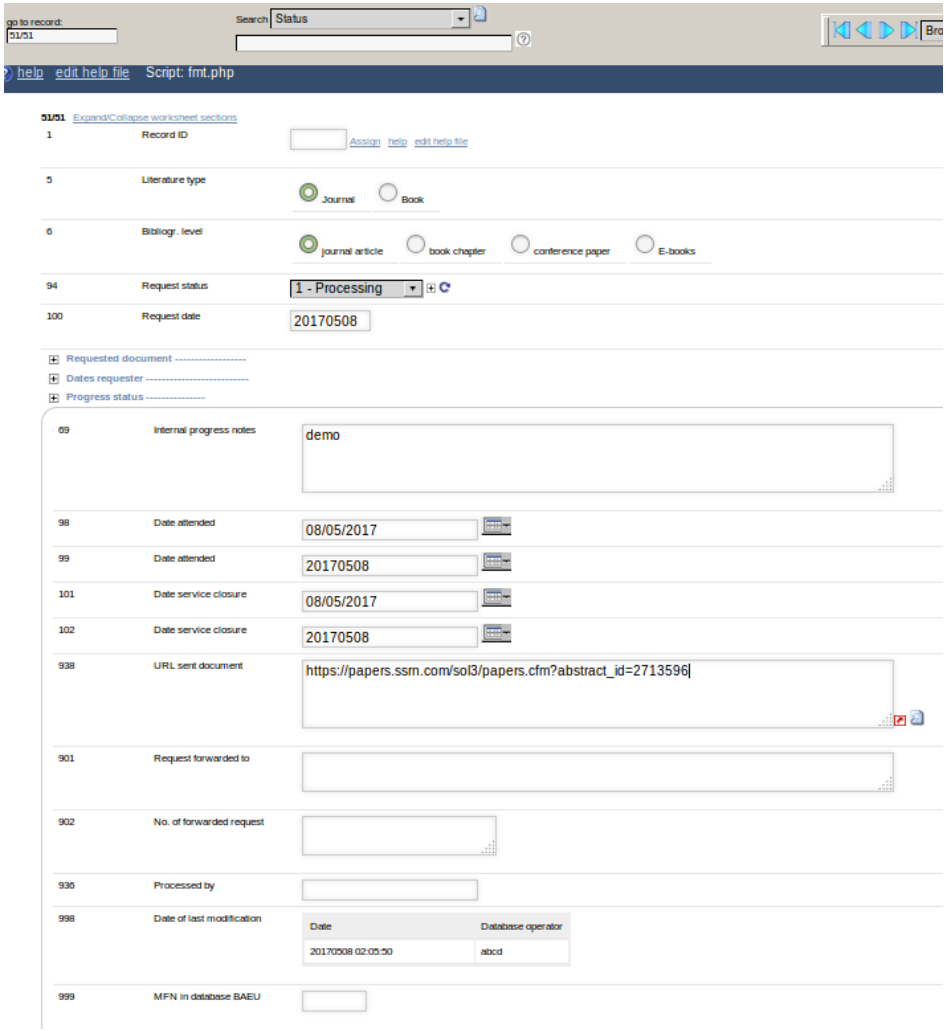
9.2.2. The ODDS processing by the library

The requests created by end-users are actually sent, by 'sending' the form, as records stored into the dedicated 'odds' database of ABCD. This means that the responsible officer or librarian needs to have administration access to this database in her/his profile. The database also needs to be included in the list of available databases (bases.dat in the bases-directory).

So in reality the ODDS-officer will check - on a regular basis, e.g. daily - whether any new incoming requests have been stored in her/his database. By opening that database and navigating to the end, the last submitted request e.g. can be opened and will be displayed with the default 'odds.pft' (which can be adapted if so desired), e.g. :

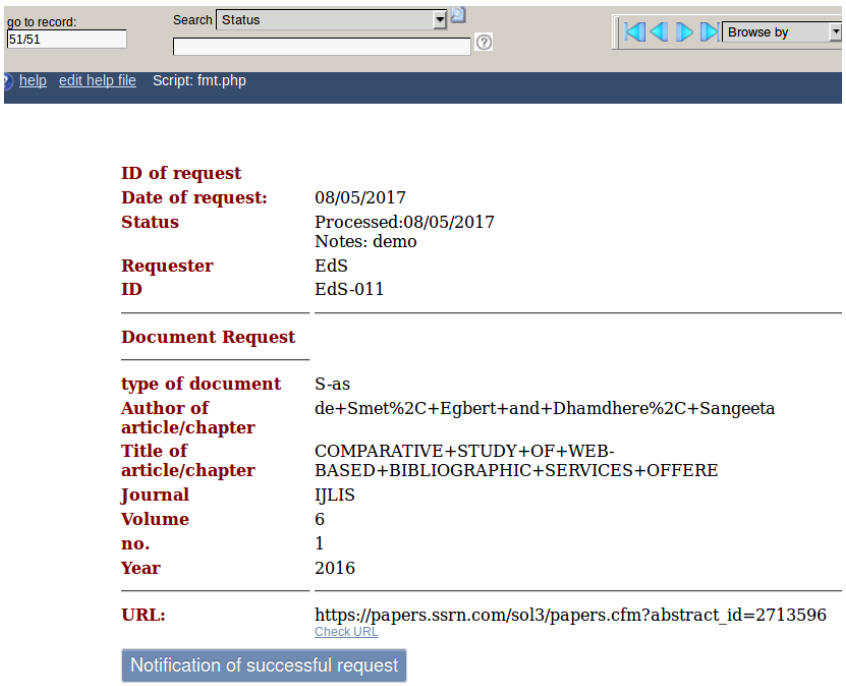


which then can be opened for editing, with an 'editor'-form, just as any other database record can be edited in ABCD Central :



The data in the record will hopefully allow the officer to indeed locate the document and download a copy on the local server. This is the main job of course and the responsibility of the ODDS-officer, who has - in principle - more or better tools available than the end-user to perform this service.

When done, the officer would then change the 'request status' from 'processing' to 'served' (or 'ready', the pick-list values of this field can be locally adjusted as with any other database-field in ABCD). When the 'request status' field is changed to the value '2', the PFT presenting the record will now show at the bottom an extra button, which actually will trigger an e-mail sent out to the requester (whose e-mail needs to be known from the user-database !).



The 'notification of successful request' will actually, when clicked on, pass on the field-values to the JavaScript performing the e-mail client command as defined in the PFT,

```
<a href="#" onclick="return getOutput(``,v528,`',``,v828,`',``,v100,`',``,v510,`',``, v94,`',``,v1938,`',
```

The javaScript 'getOutput' with its proper sequence of variables is included in the script 'sendMail.php' described above in the configuration section of this chapter. It is the responsibility of the system-manager to make sure the e-mail send-function is well tested and working. The sendMail script checks the configuration scripts either for Linux (senderMailLinux.php) and uses that configuration to send out the e-mail, or in case of using Windows, the executable 'blat.exe' (which is included in ODDS) will be executed by Windows with some e-mail fields pre-defined in the script itself, e.g. 'sender' (the name of the library), 'from', with possibly an embedded logo

The exact wording of the e-mail letter sent out is defined in either the file 'odds/def/$lang/odds_success_mail_single_file.tab' or 'odds/def/$lang/odds_success_mail_multiple_file.tab' ($lang being the code of the language used) depending on whether just one or more documents were requested and uploaded. The format is given as follows :

```
subject = Reference service - Reply to your request.
<html><head><meta http-equiv=Content-Type content=text/html; charset=iso-8859-1> </head>
<BODY>
<p>Dear <b>|name|</b>.
<br/>Your request dated |date| for the document |request_data| is available for |number_of_days| days
at the following URL: |url|</p><br>
<p>Sincerely,
<br> Library Administration<br>
<font color=red>Dept.</font><br><font color=#555555>ADDRESS:<br>Tel.:  Fax: <br>Address/Country</font><br
<font color=green>WWW: http://abcd.netcat.be </font><br></p>
</BODY></html>
```

As can be noted, the file uses 'variables' like |name| and |url| which will be substituted for the real values by the software.

If the e-mail wording needs adjustment, it can be easily done here by editing the text-file itself.

When a request is cancelled, the file 'odds_cancel_mail.tab' in the same 'def'-directory of the odds-database is used to define the contents.

10. Digital library features in ABCD

In this section we discuss some features and techniques to create and use digital libraries as 'full-text' databases in ABCD.

10.1. The Digital Library concept in ABCD

In ABCD we simplify the meaning of a 'digital library' as a 'full-text' database, in which the records either contain - in addition to 'metadata' - the full-text of a document or have a link to the full-text document while the words in that full-text can be searched.

Since ABCD is based on the CDS/ISIS technology, therefore focuses on text-retrieval, images and other non-textual elements of a document will not be processed but remain available of course in the original linked documents. E.g. illustrations in a PDF will not be part of the ABCD-record but by clicking on the document-links the PDF, including the figures, will be opened in the browser.

An important limitation is the absence of 'relevance ranking' : the indexing engine of ISIS can deal with words (full-text indexing) but has no provision for relevance ranking, e.g. storing a 'weight' denoting the relevance of a document based on the (statistical) frequency of the word occurring in the document. In ABCD 2.0 a word is there or not, and this has certainly consequences for the search efficiency. For 'ranked full-text indexing' we have to refer to the new-generation version of ABCD : version 3.0, which is based on J-ISIS and therefore the Lucene-engine which has provision for ranking.

Another limitation is the size of the documents : if the special CISIS version 'BigISIS' is used, records can contain up to 1 Mb of text. In our tests most documents (PDF's) fit within this limit except for the really long textual documents. Most bigger PDF-files are big because they contain a lot of images or binary elements (BLOBs) but the extracted text will mostly fit within the 1Mb limit. Putting the document text inside the record therefore should be only used - it is an option after all - if most of the collection consists of not-too-long textual documents. The script generating the ABCD-records of a digital library database will warn for documents not fitting (or exceeding the PHP-limit set for upload_max_filesize' parameter - which by the way has very low correlation with the actual text-size of the documents.

However, despite the limitations, using a database for a digital library in ABCD also has advantages :

1. The digital library can be managed like any other database, e.g. searching/editing in ABCD Central, including adding/editing metadata fields

2. The database or digital library can be presented in the ABCD Site and searched along like any other database (e.g. catalog, serials collection...) in the MetaSearch option of ABCD

3. If the document-text is - as an optional feature - stored inside the record, it can also be edited along with any other fields (metadata).

4. Numbers of documents in the collection hardly affect the speed of operations, e.g. CISIS can deal with almost 17 million documents without any speed-penalty in searching.

5. If the directories of the original files are structured into sub-directories, ABCD will generate and index these directory-names automatically as metadata in the 'sections' field of the records. This means searching can be based on these directory- or foldernames.

One special feature used by ABCD - as do many other similar dedicated digital library softwares - is to use the 'metadata'-extraction feature of the Apache Software Foundation TIKA library, which is a general-purpose text-extractor. This means that, provided (not obvious - most authors are not giving attention to this feature of their word-processor) that in the document properties the related fields were entered, available metadata such as title and author will be automatically entered into the fields of the database-record (see below for a sample record with such automatically extracted metadata). If defined in the Field Selection Table, these fields will also be searchable in both Central and the OPAC (or Site Metasearch).

A special instance of a digital library in ABCD would be a 'DSpace repository' created from an existing DSpace collection. For this we refer to the dedicated section in this manual.

10.2. Creating a collection in batch-mode

The batch-mode creation of a collection refers to the possibility to add a set of documents, pre-organized into one or more (sub-)folders, into an existing ABCD-database. We suggest to use the 'Dublin Core' demo-database as it

has already the basic Dublin Core fields available and the creation script presents these fields already as defaults (but they can still be changed).

A typical use case would be the initial creation of a collection of theses, where the theses as PDF-documents are stored in a folder, structured by e.g. faculty, department and year of submission. These folder-names will automatically be introduced as meta-data ('sections') in the database-records and can be used as search-criteria.

Another usage case is adding a series of newly arrived documents (e.g. journal-articles) into an existing collection. For additions of individual documents, either in a new or existing record of the collection, a different script will be used, see the next section 1.3.

10.2.1. Preparation of your collection

The following are the pre-required elements in order to allow creation of a collection in batch-mode :

- A database with pre-defined FDT, FST and PFT in which the documents will be added as records
- A new parameter 'COLLECTION=' in dr_path.def file of the related database : this parameter indicates the full path to the collection-subfolder in which the collection files will be stored. This subfolder can be anywhere in your system, but typically will be either a 'collection' directory in the 'bases' directory of your ABCD-system (e.g. ABCD\bases\collection in Windows) or a collection-subdirectory for a specific database (e.g. /var/opt/ABCD/bases/dubcore/collection in Linux). This folder will need full access/control for the script creating the document records.
- Optionally : a field in which the document text will be stored
- A special PFT (to be edited manually or copied form existing digital library database PFT), in which two special instructions are given :

  1. instruction to display the text-contents of the document into an 'Iframe' of the window :

     ```
     if p(v96) then '<tr><td width=20% valign=top><font face=arial size=2><b>Text-source</b>"</td><td valign=top><font face=arial size=2><iframe height=200 width=800 scrolling=yes src=http://localhost:9090/', replace(v96*1,'ABCD/www/bases/','collection/') ,'></iframe></td></tr>' fi
     ```

     Note

     This example works for 'localhost' with port 9090 since the 'source' of the file is referenced by 'http://localhost:9090', and the string 'ABCD/www/bases/' in the file-path and -name in v96 will be replaced by 'collection', so this supposes Apache has an 'alias' in which 'collection' refers to the real database-directory. The name of the database (e.g. 'dubcore') will need to be present as well as the 'ABCDSourceRepo' subfolder name.

  2. instruction to display the link to the original document file on the server :

     ```
     if p(v98) then '<tr><td width=20% valign=top><font face=arial size=2><b>URL</b></td><td valign=top> <div id=divurld',f(mfn,1,0),' name=divurld',f(mfn,1,0),'><a href=javascript:DisplayURL(',f(mfn,1,0),')>DISPLAY</a></div> <div id=divurl',f(mfn,1,0),' name=divurl',f(mfn,1,0),' style=display:none><font face=arial size=2> <A HREF="',v98'" target=new>'v98+|<br>|,'</A></div></td>' fi/
     ```

     Note

     This example is a bit more advanced since it includes the mechanism to hide the URL with only a link 'DISPLAY', which triggers a Javascript function 'DisplayURL' which will invite the user to log in as a library user and only if correctly logged in will actually display the real URL. That URL is indicated by the value

of v98 in this example. As an alternative to 'logging in' ABCD can also use a check on the end-users IP-number (or range) and only show the URL to users within the accepted IP-range :

The login Javascript code, to be added on top of the PFT, is listed under here :

<script language=javascript> function DisplayURL(id) { var loginto=document.getElementById("into").value; if (loginto=="no") { var posicion_x; var posicion_y; posicion_x=(screen.width/2)-(315); posicion_y=(screen.height/2)-(235); sel = window.open("/site/login.php?id="+id, "ABCD Log In Windows", "width=630,height=470,menubar=0,toolbar=0,directories=0,scrollbars=no,resizable=no,left="+posicion_x+",top="+posicion_y+""); } else { document.getElementById("divurld"+id).style.display="none"; document.getElementById("divurl"+id).style.display="block"; } } </script>

As an alternative the IP-range of the end-user can be checked to (dis-)allow the display of the URL, with the following code added into the PFT :

proc('<9000>',getenv('REMOTE_ADDR'),'</9000>'), s1:=(left(v9000,instr(v9000,'.')-1)), s2:=(mid(v9000,instr(v9000,'.')+1,size(v9000)))/, s2:=(left(s2,instr(s2,'.')-1)),

if p(v98) then '<tr><td width=20% valign=top><font face=arial size=2>URL</td><td valign=top>', if val(s1)=127 then if val(s2)=0 then '<A HREF="http://127.0.0.1:9090//docs/dubcore/collection/',v98, '" target="new">'v98,'</A></td>' else 'IP not allowed' fi else 'IP not allowed' fi fi,

Note

This instruction has two separate parts : at the beginning of the PFT a 'proc' is used to store the IP-number in a virtual field v9000 and then its parts into local variables 's1' and 's2' respectively. Then, at the location of the PFT where you want to display (or not) the URL, and if v98 (with the URL info) is present only, the value of s1 and s2 are checked to see if they are contained within the allowed range, in this case 'localhost' (127.0.x.x).

In principle the system manager (or librarian) needs to decide which method to restrict access to the original documents to use : either log-in based or IP-range based. However using normal PFT-instructions (e.g. IF...THEN...FI) one can combine them or simply use both, e.g. labeled 'Display after log-in' and 'Display within campus'.

- A special FST, named 'fulltext.fst' in the database 'data'-directory, which contains an indexing instruction (line) which uses the document-text as input for the indexing engine with the 'cat' instruction, e.g. in the case of the existing Dublin Core database ('dubcore') :

  99 8 '/TW_/',if p(v96) then proc('Gload/99/nonl='v96) fi,v99

  This instruction creates an index identified by '99', using the prefixed words-indexing method (8) of ISIS, by first creating the prefix 'TW_' (text-words, typically the index for the simple search of ABCD OPAC), then if the field 96 is present to indicate the path/name of the text-file, to load that file into v99 (without new-line characters) and use that text as input for the indexing engine. In addition to this special instruction all other meta-data oriented indexing instructions, e.g. to include title/author etc., can be used in this special FST.

- A 'normal' FST, mostly meta-data oriented, to be used whenever the record meta-data fields have been edited and are saved.

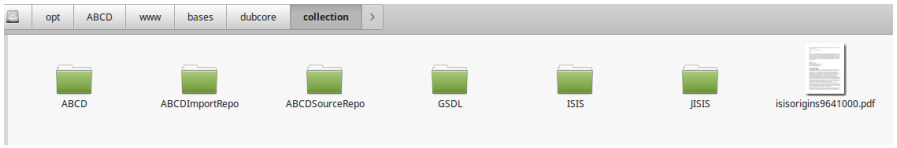
- A fixed-named 'collection' subdirectory of the related database-directory

- A fixed-named subdirectoy of the COLLECTION-directory 'ABCDImportRepo' in which the documents to be processed are located.

It is of utmost importance that both the collection- and ABCDImportRepo- subfolders have full access (or in Windows 'Full Control') to the users, since files will be moved from there and new files will be written into the collection-folder. *Make sure, using your Operating System's interface, that this condition is fulfilled. E.g. in Windows right-click on the folder and check if the 'security'-tab indicates 'full control' for the users. In Linux make sure the directory has '777' attributes (e.g. with the command 'chmod -R 777 collection'). Please also note that also higher-level directories or folders need to have full access since the OS will parse through all levels from the 'root' up to the related collection-directory and stop doing so if no full access it allowed.*

A typical structure will look like the following, where documents are related to either ABCD, ISIS, Greenstone (GSDO) or J-ISIS, so inside the ABCDImportRepo subdirectory the original documents were stored inside subfolders with corresponding names (ABCD, ISIS, GSDL, J-ISIS), except for a special document 'ISISorigins.pdf' which was left out of this sections-structure and therefore will end up in the 'root' of the collection.

Figure 2.3. Typical COLLECTION structure



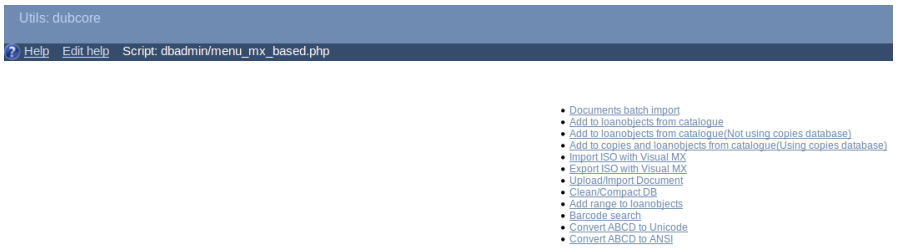
The ABCDSourceRepo subdirectory will contain all HTML-files with the extracted texts, all original documents (e.g. PDFs, Word-documents...) will be stored in the corresponding subfolders as they were present in the ABCDImportRepo directory, but with some random numbering added to the file-name. This is to ensure that documents with the same document-names still are individually identified. The resulting ABCD-record will have both the reference to the HTML-file (in ABCDSourceRepo) as the original document into its dedicated fields, in the demo DUBCORE database : resp. v96 and v99. In this example the'isisorigins.pdf' was not moved into a subfolder - only for demonstration purposes - because inside the ABCDImportRepo folder it also resided at the 'root'-level.

10.2.2. Using the creation script

After you have ensured all preparation steps are properly dealt with, the use of the script to create digital library records is rather straightforward.

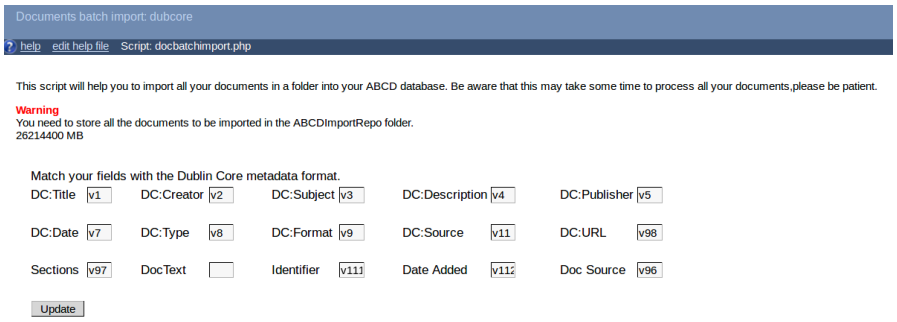
The script for batch-import of documents is the first one in the new 'EXTRA UTILITIES' submenu of the Central Utilities menu :

Figure 2.4. EXTRA UTILITIES menu



After the 'Documents batch import' option is selected, the main screen is shown, in the case of the Linux version after a count-down to load the Tika-server into memory (since the server-version, not used in Windows) takes longer to load) :

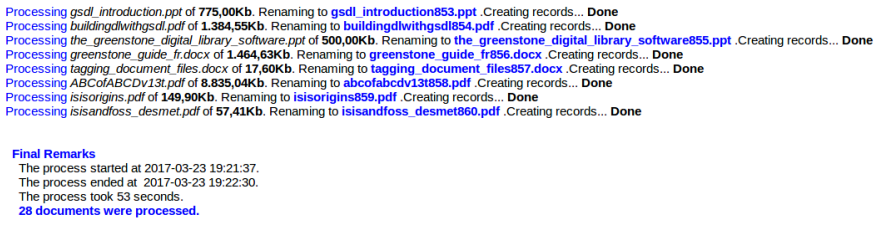
Figure 2.5. Main screen Batch Documents Import



In this screen the librarian has to 'match' the automatically extracted metadata (Dublin Core-based) to the fields of her/his database, along with some special fields : the field to contain the 'sections' (=subfolders of the collection), the document source, the document identifier and optionally the field to contain the full document text extracted by TIKA. Note that this is not used by default as it will require the BIGISIS CISIS-version in order to use larger records.

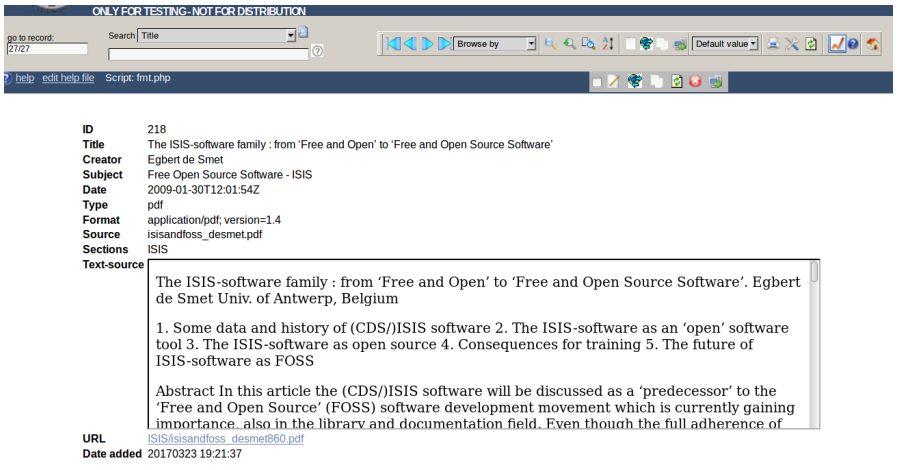
Clicking on the 'Update' button will launch the iterative script which processes all files in the ABCDImportRepo subdirectory and displays the progress on the screen until it is finished. The illustration below shows the last part of such listing when the script finished.

Figure 2.6. Results listing after running collection creation script



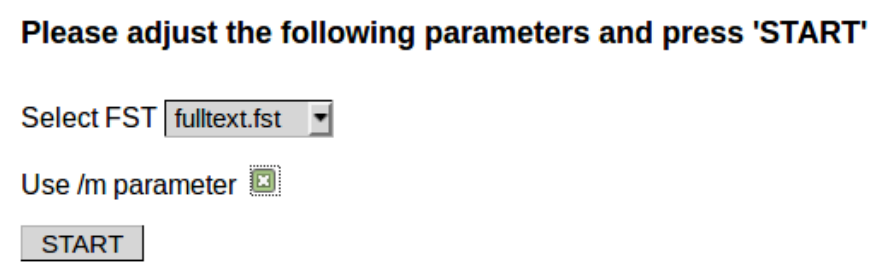
Now entering (still within Central) into the database and navigating to the last record will show a typical record as follows :

Figure 2.7. Display DigLib record in Central



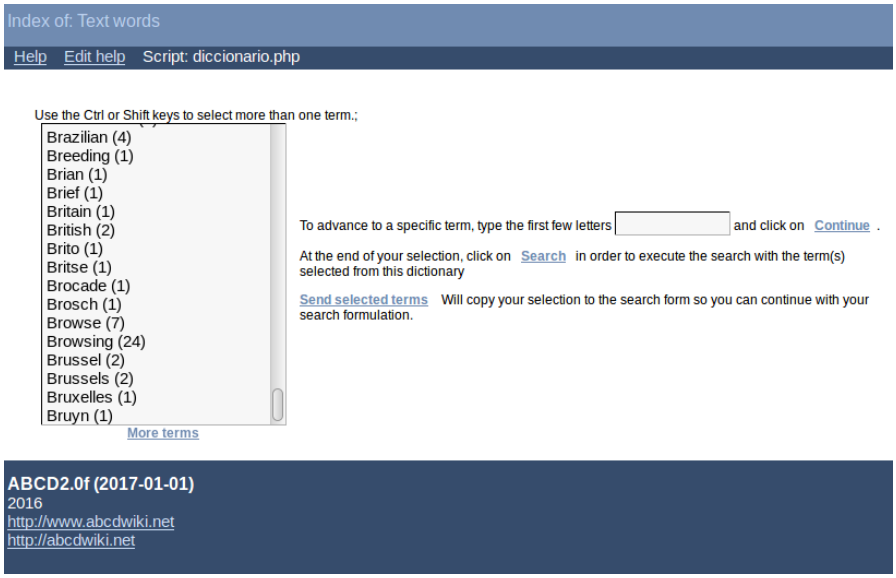
The database needs to be fully indexed ('full inverted file generation' in ISIS-vocabulary) using the earlier mentioned special FST 'fulltext.fst', but also using the special parameter '/m' in order to avoid storing all detailed position info into the index-postings :

Figure 2.8. Full text indexing of Digital Library collection



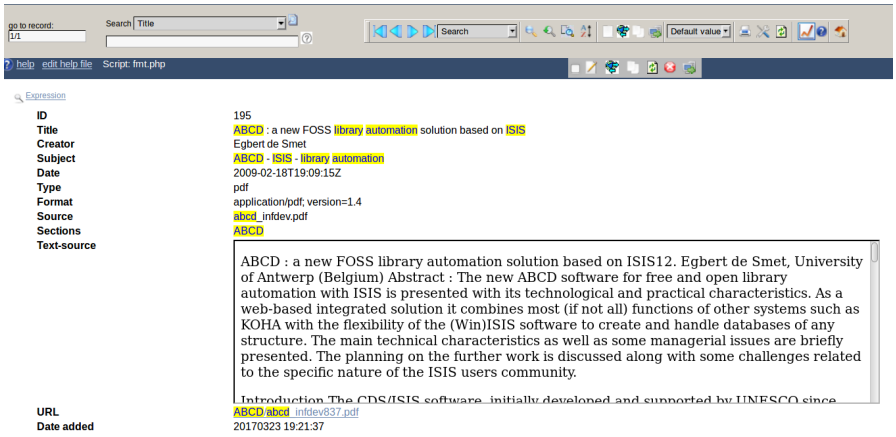
An example of - part of - a full-text index is shown in the next illustration :

Figure 2.9. Full text index listing



Selecting one or more words, as is also done in any other Central search action on any other database, results in displaying the record(s) with highlighting of search-terms :

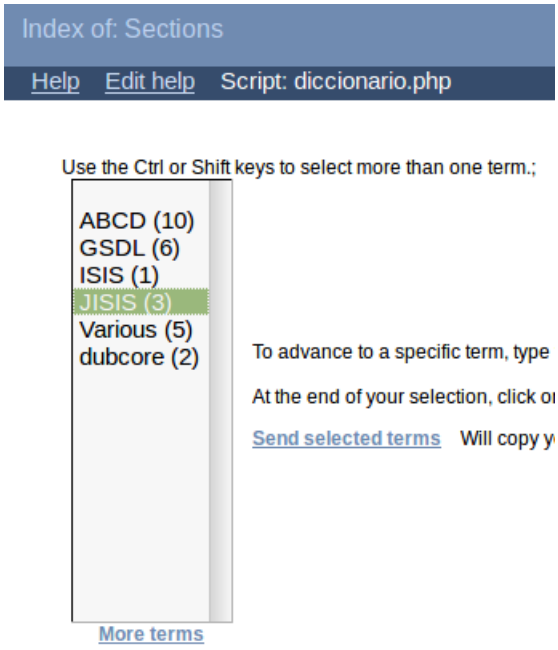
Figure 2.10. Digital Library search record display



The text-source is just shown as a first quick-check to assess the relevance of the document; this is of limited use but e.g. in the OPAC can help end-users to decide on whether or not they click on the URL-link to view the original document in full format.

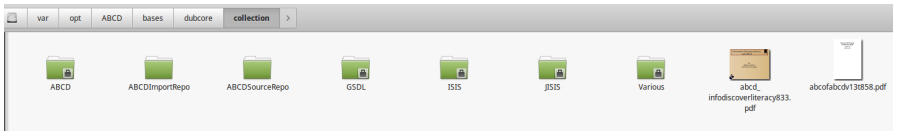
In order to illustrate the earlier mentioned 'sections' feature, where ABCD uses the pre-configured subfolders of the ABCDImportRepo directory to automatically assign 'sections' to the collection, here is the listing of the 'sections'-index in ABCD Central :

Figure 2.11. ABCD Digital Library Sections listing



This reflects the now adjusted folder structure of the collection-directory : subfolders from ABCDImportRepo were reconstructed into the collection-root folder with the original files moved there, and the ABCDImportRepo folder is emptied, only leaving unprocessed files there (e.g. too large) :

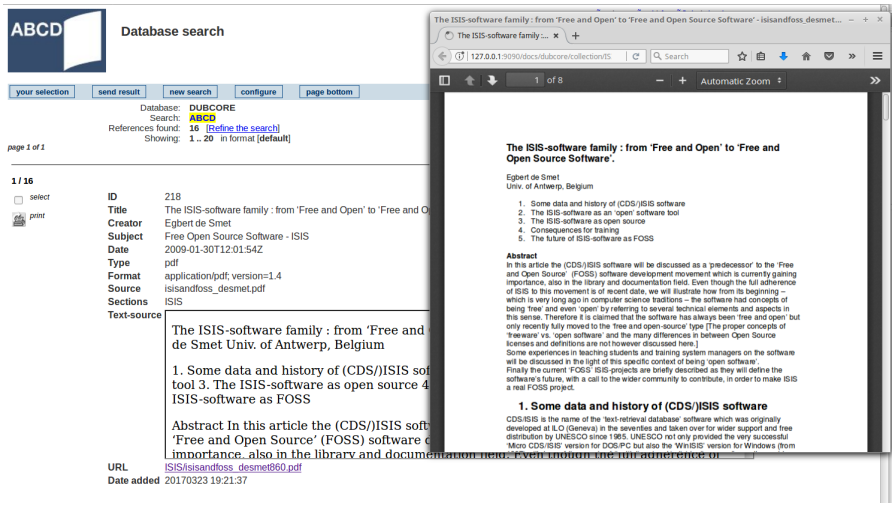
Figure 2.12. ABCD Digital Library new collection directory structure



As one can see : in addition to the 'ABCDImportRepo' folder (now emptied) there is a folder 'ABCDSourceRepo' containing all html-files with extracted texts, and there are 'section-'folders for ABCD, GSDL, ISIS, JISIS and 'Various', while two PDF's were not part of the sections and therefore remain at the 'root'-level of the collection without 'section' information.

The digital library database can be included in the ABCD Site MetaSearch (as is the case in the demo installation) and searched by end-users like any other database. The result display is very similar to the one above in Central :

Figure 2.13. DIgital Library search result in iAH-OPACh



This screenshot also shows at the right side the original PDF-document, after having clicked on the URL-link as a hyperlink.

Note

The 'login'-based protection method using the javascript 'Display_URL' does not work from Central but works - as intended - from the iAH OPAC.

10.3. Interactive upload and adding documents into a collection

11. The ABCD OAI interface

This section will present the ABCD OAI interface, which is an implementation of BIREME's 'ISIS-OAI-PROVIDER', a general tool for providing access to ISIS databases through the OAI-PMH protocol.

11.1. Short introduction to the OAI-PMH concept.

OAI, or 'Open Archive Initiative' is a world-wide initiative in the information communities to avail information in an open, free way.

For more informaiton on this organisation, the 'Openarchives.org', see http://www.openarchives.org.

The Open Archives Initiative develops and promotes interoperability standards that aim to facilitate the efficient dissemination of content. OAI has its roots in the open access and institutional repository movements.

The three current main areas of activity are :

1. OAI-PMH : OAI Protocol for Metadata Harvesting : a protocol to allow retrieving metadata on electronic documents with an open standard, meaning one does not need to use the software with which the metadata were/are managed.

2. ResourceSync Framework Specification : a protocol to make it possible for servers to synchronize their references to resources which might move or migratie on the WWW so as to always remain updated (pointing to the correct URL)

3. Open Archives Initiative Object Exchange and Reuse : a mechanism to allow 'rich information documents' (with images, movies etc.) to be presented as one 'aggregated resource' independent from the individual locations of the components. E.g. social media The intent is to develop standards that generalize across all web-based information including the increasing popular social networks of “web 2.0.

In ABCD the first of these three features is implemented : the Protocol for Metadata Harvesting or OAI-PMH.

The OAI-PMH is the main protocol developed and promoted by the OAI, and described as :

“The Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) is a low-barrier mechanism for repository interoperability. Data Providers are repositories that expose structured metadata via OAI-PMH. Service Providers then make OAI-PMH service requests to harvest that metadata. OAI-PMH is a set of six verbs or services that are invoked within HTTP. ”

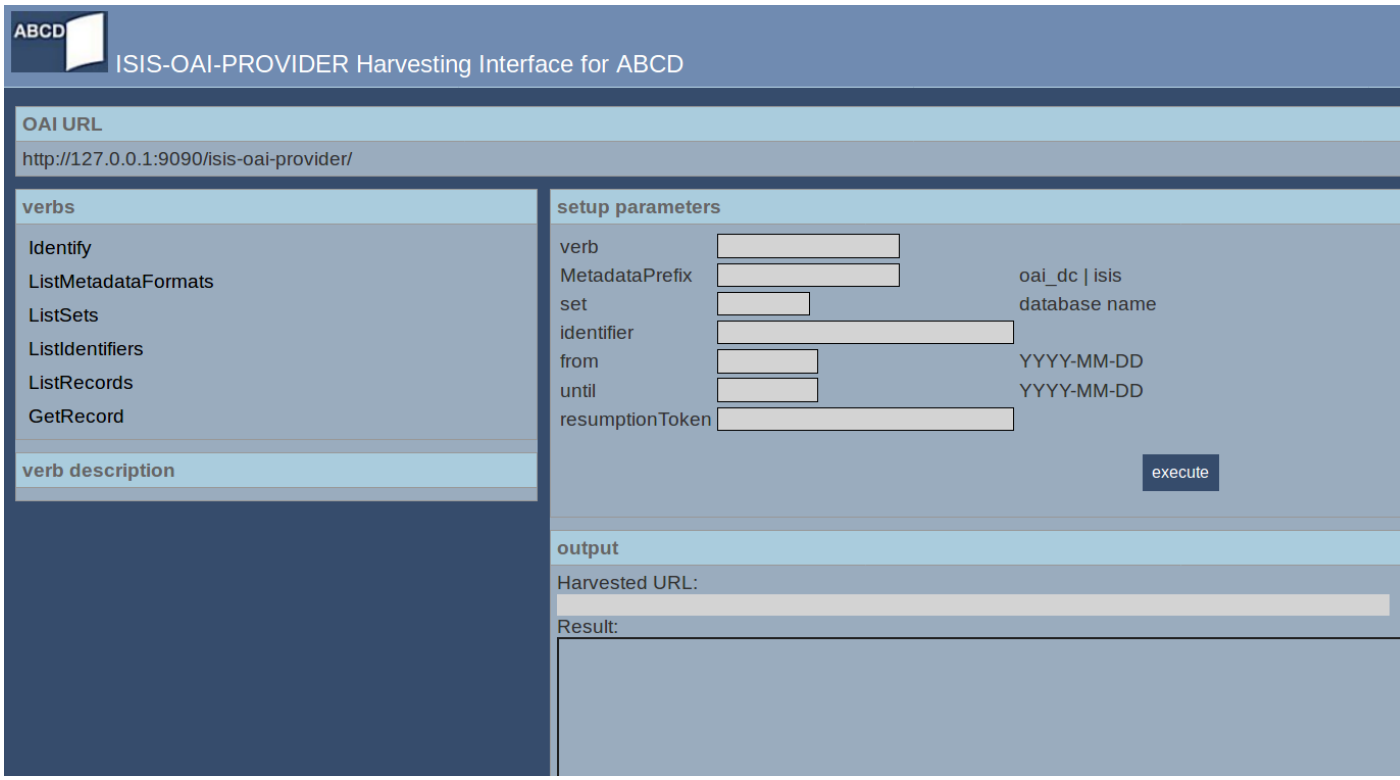
The idea is to provide access to 'open' meta-data based information in as much a 'generic' and open way as possible, meaning that any need of specific software should be avoided. For this reason the protocol agreed on 6 'verrbs' or commands which allow to identify properly the source of the information and retrieval of basic meta-data information of documents available, often in a 'repository', either in a list of document-identifiers or, based on one such identifier, a specific document. In order to facilitate easier update of repositories taking their info from other repositories, selections can be limited by dates.

11.2. Implementation and configuration in ABCD

The implementation in ABCD is fully based on the tool 'ISIS-OAI-PROVIDER' produced by BIREME as part of their Virtual Health Library (see e.g. the website about this tool :http://wiki.bireme.org/en/index.php/Isis-oai-provider and for downloading the original software the GitHub repository : :https://github.com/bireme/isis-oai-provider .

Since no ABCD or other ISIS-related software can be supposed to exist at the users' side, the mechanism is to provide a URL and present the interface as just a web-page which is freely accessible. Therefore no links from within ABCD (e.g. in 'Central' module) is provided but the URL should be entered directly in the browser's address bar (or referenced by a shortcut etc.). In the case of an ABCD 'localhost:9090' installation this URL would be : http://localhost:9090/isis-oai-provider/index.php.

The page opened by that URL should like like the following illustration :



As can be seen, in the left part of the interface access is given to each of the 6 main 'verbs' or operators, with a short description of each verb after selection of one of them, while at the right part the user-interacion has two parts : the criteria defined by the user at the upper part and the XML-output (to be 'captured' with copy/paste, in some browsers from the 'frame source' if the full XML-code is needed) as the result of the harvesting action.

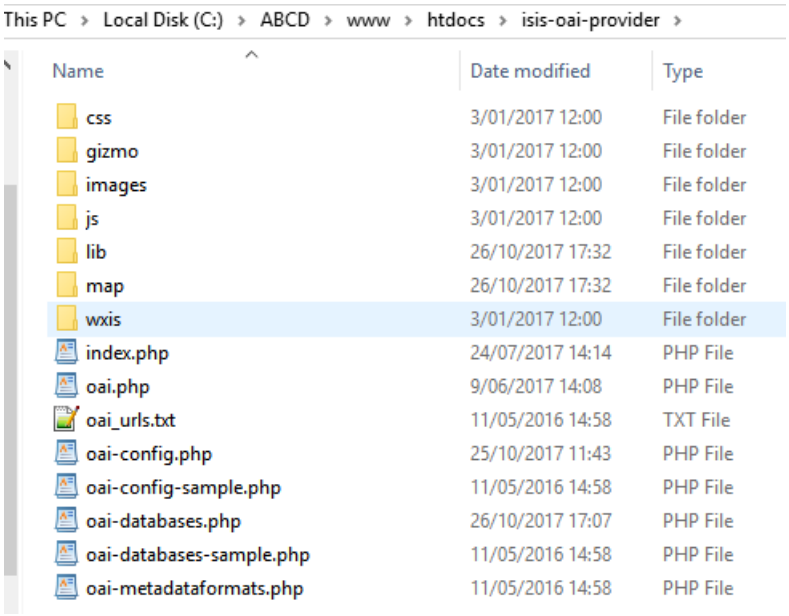
We will first briefly discuss the configuration necessary to make this work for local ISIS-databases in ABCD and then discuss each of the 6 verbs with some illustration.

11.2.1. Configuration of the interface

Configuring ABCD for the OAI-interface comprises two parts : a general configuration and a configuration file for all involved databases.

11.2.2. The general configuration

The file 'oai-config.php' is located at the 'root' of the Isis-OAI-Provder module in ABCD, which is stored, next to other modules like central, iah, secs-web and site, in the htdocs-folder of ABCD. A typical Windows file-manager view of this folder's content looks like this :



So this module uses its own CSS stylesheets and images, allows for gizmo's to be applied and uses one JavaScript file with 'functions' called from the scripts. The most important sudirectories here are :

- lib : this directory contains the main PHP-scripts for the interface, partly taken - inherited - from the general PHP-OAI libraries (e.g. OAIServer.php) and partly dedicated PHP-scripts to deal with resp. ISIS-databases (ISISDb.php) or one (ISISItem.php) or more (ISISItemFactory.php) ISIS records. The configuration parser-scripts are also here for resp. parsing the ABCD-OAI configuration or the 'mapping' of the database fields to the Dublin-Core standard.

- map : this directory contains at least one file for each database involved, i.e. following either one of the two allowed syntaxes : OAI_DC (files with the name of the database followed by the .i2x extension) or ISIS (more traditional ISIS PFT formats with th eextension .pft). This mapping will be discussed in more detail below.

- wxis : some ISIS-Scripts used by the main executalbe wxis (web-enabled ISIS). getidientifiers.xis, getrecord.xis and gettotal.xis. These are rather 'normal' ISISScript instructions, taking variables form the CGI-environment and then specifying the instructions to operate on the variables, e.g. printing the titles in a loop etc.

- In the main isis-oai-provider directory itself we find the opening PHP-script index.php (which contains the interface) and the sample and real configuration files as PHP-scripts.

For the configuration of the ABCD-OAI interface we need to consider two files : one for the general configuration, one for the databases.

A sample listing, as it comes with the demo-implementation, is given here, where lines starting with ';' are comments, e.g. to illustrate the Windows-alternatives :

```
[ENVIRONMENT]
```

```
; Set the directory of application. The effect will be http://your-site.org/DIRECTORY

DIRECTORY=/isis-oai-provider ; Full path to folder 'site' in folder bases. Windows users must
NOT put letter unit in path

DATABASE_PATH=/var/opt/ABCD/bases/

;DATABASE_PATH=/ABCD/www/bases/

EXE_EXTENSION=

;EXE_EXTENSION=.exe ; Set the directory configured to execute cgi-bin applications.

CGI-BIN_DIRECTORY=/cgi-bin/

[INFORMATION] ; Set the repository's name.

NAME=ABCD

; Set the email of the system admin

EMAIL=egbert.desmet@uantwerpen.be

IDPREFIX=be

IDDOMAIN=netcat

EARLIESTDATESTAMP=2011-01-01

MAX_ITEMS_PER_PASS=20
```

In the [ENVIRONMENT[section the following variables need to be defined :

- DIRECTORY : the folder of the module, starting from the 'ABCD DocumentRoot' (as defined in the webserver, e.g. Apache), so in the default case of ABCD in Linux where the document-root is defined as '/opt/ABCD/www/htdocs', defineing the DIRECTOY as 'isis-oai-provider' means that the OAI-module is located in /opt/ABCD/www/htdocs/isis-oai-provider.
- DATABASE_PATH : the directory where the databases are to be found
- EXE_EXTENSION : the extension to be added to the 'wxis' executable name, in Windows : '.exe', in Linux no extension is used.
- CGI-BIN_DIRECTORY : the directory as it should be recognized in the URL In the case of Apache the Apache-configuration explains which real directory corresponds with the URL-string '/cgi-bin/'.

In the [INFORMATION] section we deinfe :

- NAME : the name of your repository as you want it to be seen by the users
- EMAIL : the e-mail address of the administrator who can be contacted about the repository
- IDPREFIX and IDDOMAIN : together they identify the 'domain' of the server
- EARLIESTDATESTAMP : the starting data of the repository, i.e. the date of the oldest documents, in the format YYYY-MM-DD.
- MAX_ITEMS_PER_PASS : the number of records to be listed in the ListIdentifers or ListRecords verbs. At the end of the list a 'resumption'-code will be given which can be used to define the start of the next batch. Default here is 20 items.

11.2.3. The databases configuration

One or more databases can be used in the OAI-interface, where they are referred to, in the OAI-vocabulary, as 'sets'. The sample implementation in ABCD uses 2 such sets : MARC (a marc21 catalog) and DUBCORE. (a repository with papers, but this is just a sample full-text database which the ABCD-administrator has to implement !)..

The **database definition** :

Each database needs to be defined, but also will need at least one 'mapping' file which we will discuss afterwards. The parameters to be defined for each are rather self-explanatory, like name, description, path (to the database-files), database (the actual file.name to which '.mst' will be added to define the ISIS MST file), while some other parameters can be briefly explained additionally :

- mapping : the file where the 'mapping' (the ISIS-fields corresponding with the related DublinCore elements) are described in either a very simple 'DC' format (.i2x) or a more sophisticated format based on the ISIS Formatting Language (.pft), see further;
- idprefix : the string to be prefixed (added 'before') the actual identifier when listing the record-identifiers; this additional string is optional;
- cisis_version : this is the identifer of the specific CISIS-version which is used (as from ABCD2.0 different CISIS-versions can be used, e.g. BIGISIS or UTF8/FFI etc. The version specified here should reflect the (sub-)directory structure naming into the 'cgi-bin' directory where the correct 'wxis' (or in Windows : wxis.exe) is located to deal with the database concenred. This can be different for any database defined.

  > **Note**
  >
  > We kchanged the original BIREME-name of the variable 'isis_key_length' (e.g. 1030, 1660...), but indeed in ABCD the CISIS version is to be referred to here. This is one real difference with the original default BIREME-tool.

- identifier-field : the tag (number) of the ISIS-field which contains the identifier of the record;
- date-stamp-field : the tag (number) of the ISIS-field which contains the datestamp of the record.

We use the earlier mentioned 2 databases or 'sets' to illustrate the configuration of a database (in this case for Linux) :

```
[marc]

name=marc

description="MARC"

path=/var/opt/ABCD/bases/marc/data

database=/var/opt/ABCD/bases/marc/data/marc

mapping=marc.i2x

prefix=oai_date_

cisis_version=ansi ;

identifier_field=1

datestamp_field=980

[dubcore]

name=dubcore
```

description="DublinCore repository"

path=/var/opt/ABCD/bases/dubcore/data

database=/var/opt/ABCD/bases/dubcore/data/dubcore

mapping=dubcore.i2x

prefix=oai_date_

cisis_version=utf8/bigisis ;

dentifier_field=111

datestamp_field=507

The **database 'mapping'** explains how the ISIS-fields correspond with the DC or XML-elements. It is best to illustrate this based on the samples provided, starting with the very simple example 'marc.i2x' :

root=oai-dc

245 dc:title

500 dc:description

650 dc:subject

001 dc:identifier

980 dc:date

First the 'root' string is defined as it will be used in the interface records-display. In the case of the oai-dc prefix used (see the 'usage' section below) just put 'oai-dc.

This 'root' is followed by a list of fields, one for each line, consisting of the ISIS-field-tag folllowed by the Dublin-Core element-name, e.g. the MARC21 title field 245 becomes : 245 dc-title. Note that this list can be a selection of fields to be shown. Non-defined fields here will just keep the field-tag as the XNL-element, without translation to a proper DC-eleement.

When more powerful formatting is desired, e.g. on the level of subfields, the alternative more complicated syntax of an ISIS PFT can be used, then referring to a file with the .pft extension. Such PFT 'prints' the necessary XML-tags, i.e. the XML-leader info, as 'literals (unconditional quoted strings) :

```
'       <oai_dc:dc
            xmlns:oai_dc="http://www.openarchives.org/OAI/2.0/oai_dc/"
            xmlns:dc="http://purl.org/dc/elements/1.1/"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/oai_dc/
            http://www.openarchives.org/OAI/2.0/oai_dc.xsd">'/
```

followed by the individual ISIS-fields in the Formatting Language syntax used to add literals for the DC-XML output, in the case of e.g. the 'title' (v2 in Dubcore) :

```
if p(v2) then
(|  <dc:title><![CDATA[|v2^*|]]></dc:title>|/),
fi,
```

or, for the tile iand authors n MARC :

```
if p(v245) then
```

```
(|  <dc:title><![CDATA[|v245^*|]]></dc:title>|/),
|  <dc:title><![CDATA[|v246^*|]]></dc:title>|/,
fi,
```

```
if p(v100) or p(v110) then
   (if p(v100) then
|  <dc:creator><![CDATA[|v100^*|]]></dc:creator>|/,
   else
|  <dc:creator><![CDATA[|v110^*|]]></dc:creator>|/,
   fi),  fi
```

which should be read as : if the title is present then print the XML tags <dc:title> and </dc:title> with the actual value of v2 (any subfield, indicated by the ^* operator, in between the [CDATA[...] attribute.

Any more complicated PFT-statements can be used, e.g. for displaying the document-format, if it is indicated in v9 :

```
'  <dc:format>'select s(mpu,v9,mpl)
                        case 'A':'text',if p(v8^q) then '/'v8[1]^q fi,
                        case 'MATERIAL TEXTUAL':'text',
                        case 'G':'video',
                        case 'I':'audio',
                        elsecase if a(v9) then 'text',if p(v8^q) then '/'v8[1]^q fi, else 'other' fi
              endsel,
'</dc:format>'/,
```

Any sequence of ISIS-fields can be treated this way, but fields not mentioned here will be omitted from the output. Instead of fields also subfields or groups of fields can be used, using the power of the Formatting Language.

Best is to re-use an existing model, as provided with the demo-installation, for a new different local database, mostly adapting the field-tags to the local structure of the database. One has to be very careful, as always with the ISIS Formatting Language : missing a quote or any other syntax mistake will produce no output at all but some error messages in the XML-output !

11.3. Using the interface

When everything is properly configured - a lot of preliminary testing is advised, after all OAI means exposing your database to the whole world ! - the following outputs can be obtained, presented by each one of the six OAI-PHM 'verbs' :

Note

With most illustratoins under here, the XML-output is incomplete, in other words : the actual output contains more data but only part can be shown here.

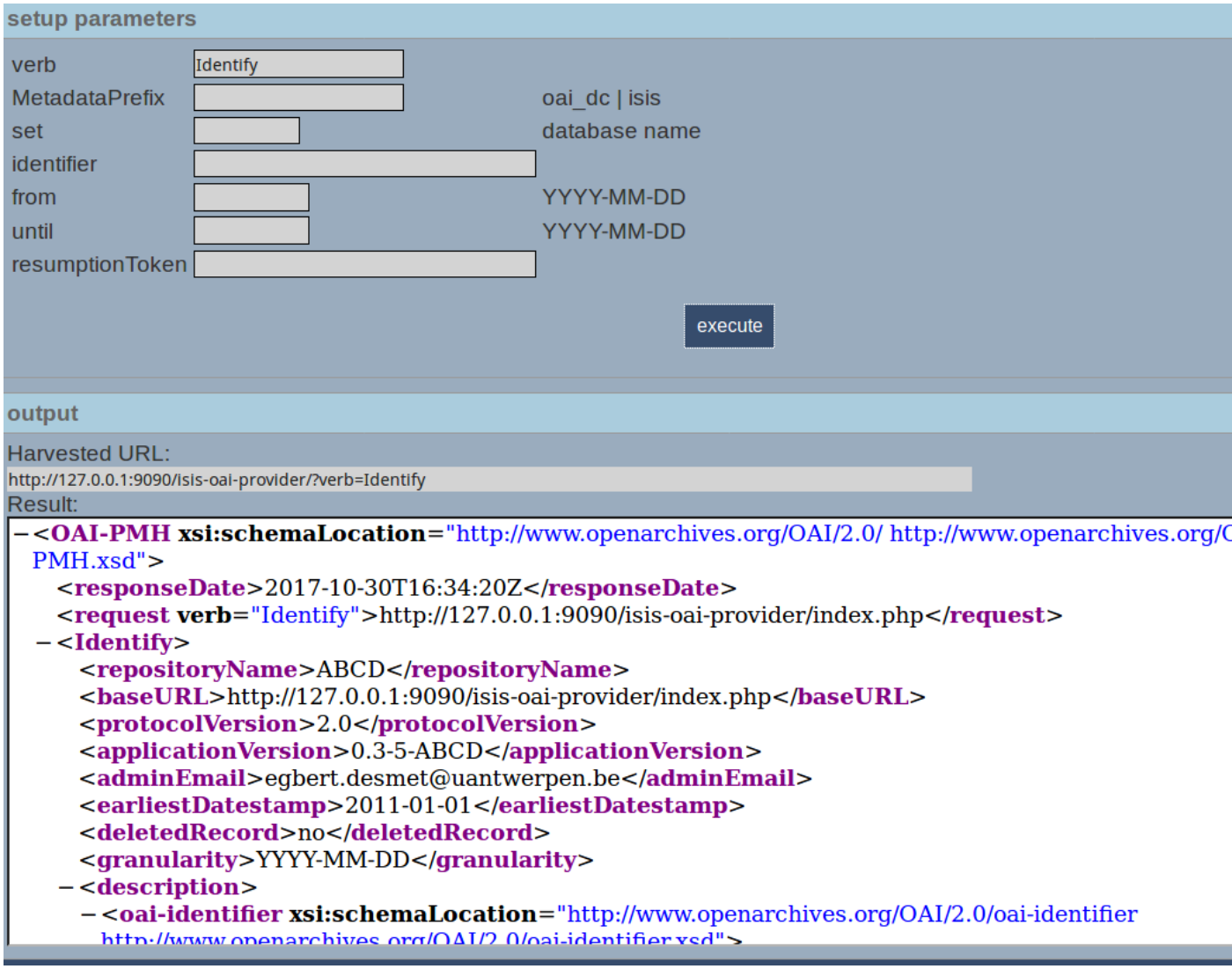
Note

The illustrations are generated with the Firefox browser, immediately showing the XML-output because no XML-style being available ('This XML file does not appear to have any style information associated with it. The document tree is shown below'). Other browsers might show just the text-output, the full-XML still being available by right-clicking and requesting the 'frame-source'.

11.3.1. Identify

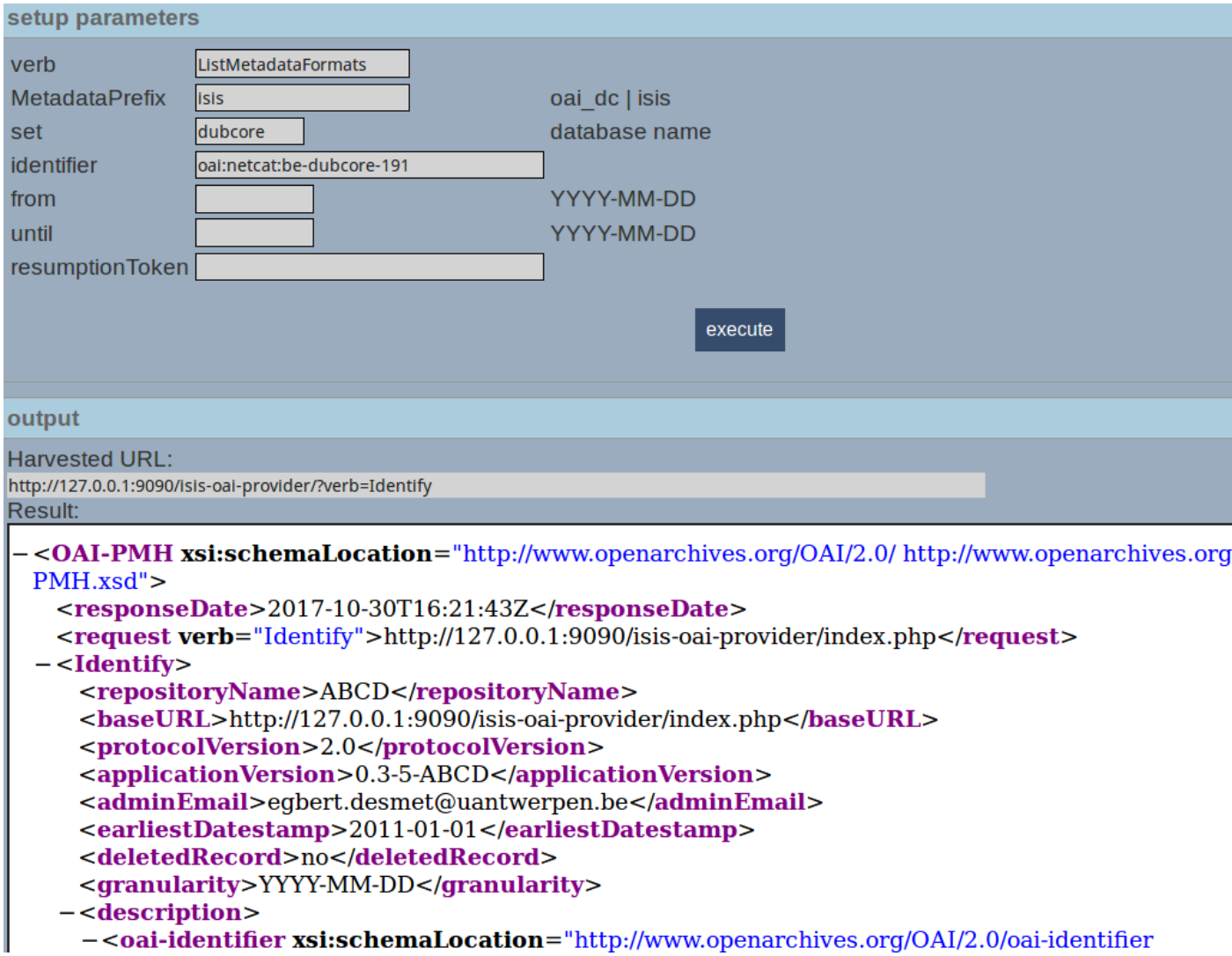
This verb returns the main protocol information. Compulsory parameters verb (automatcially entered by electing this verb).

The output contains all necessary information as agreed per the protocol, e.g. the 'name' of the protocol as indicated in the oai-config script.



11.3.2. ListMetadataFormats

This verb describes the metadata formats used in the protocol. Compulsory parameters verb In ABCD we use two different formats : oai_dc and isis. which will correspond with the 'mapping' file defined, either i2x (for oai-dc) or pft for isis.



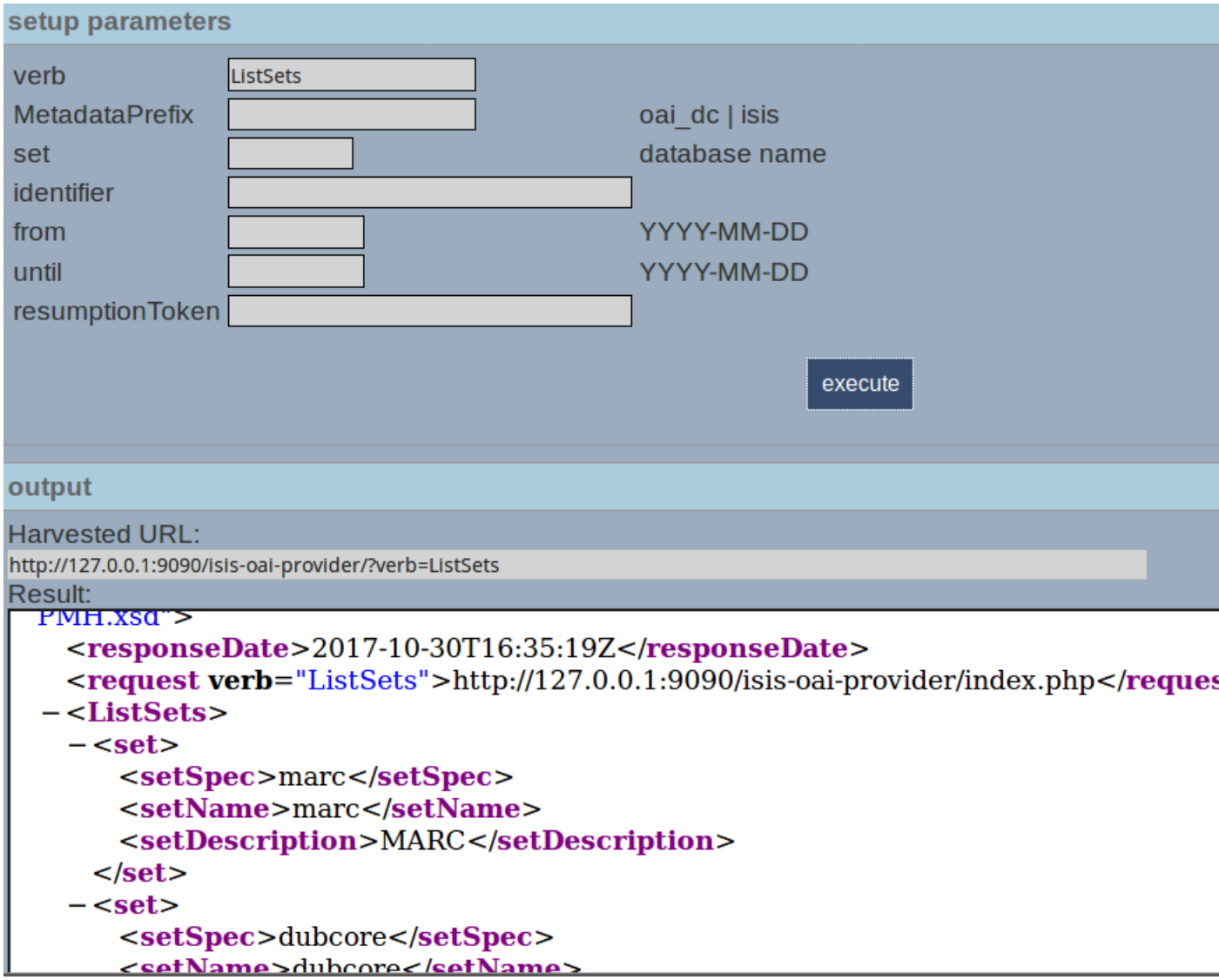
11.3.3. ListSets

Retrieves a list of all databases available on this repository. Compulsory parameters verb Exclusive parameter resumptionToken

Note

'exclusve parameter' means that this parameter is not allowed as it is incompatible with the idea of the verb itself. The interface will not allow to put an exclusive parameter.

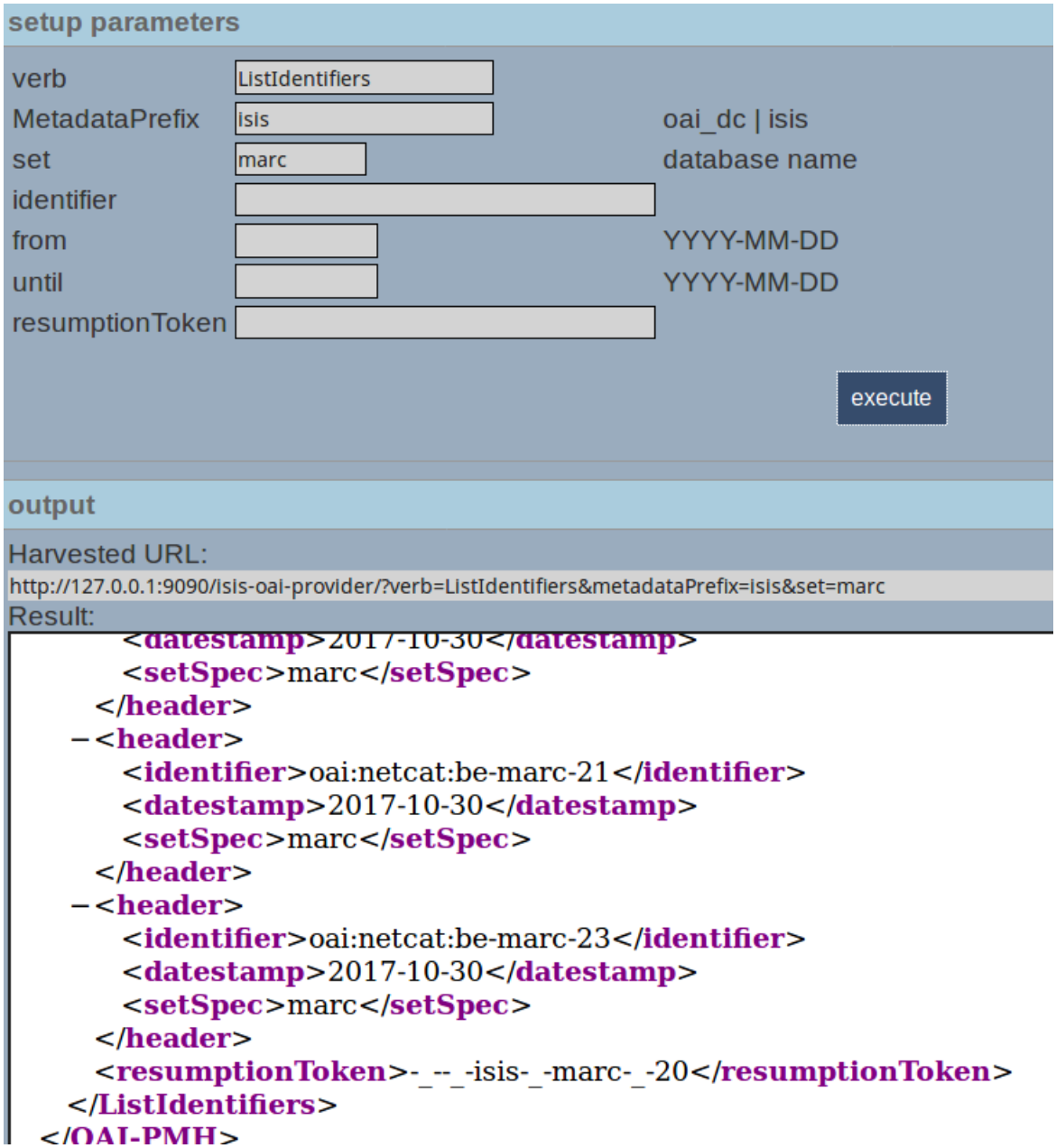
In case of our 2 demo-databases here (marc and dubcore) the output will look somethng like :



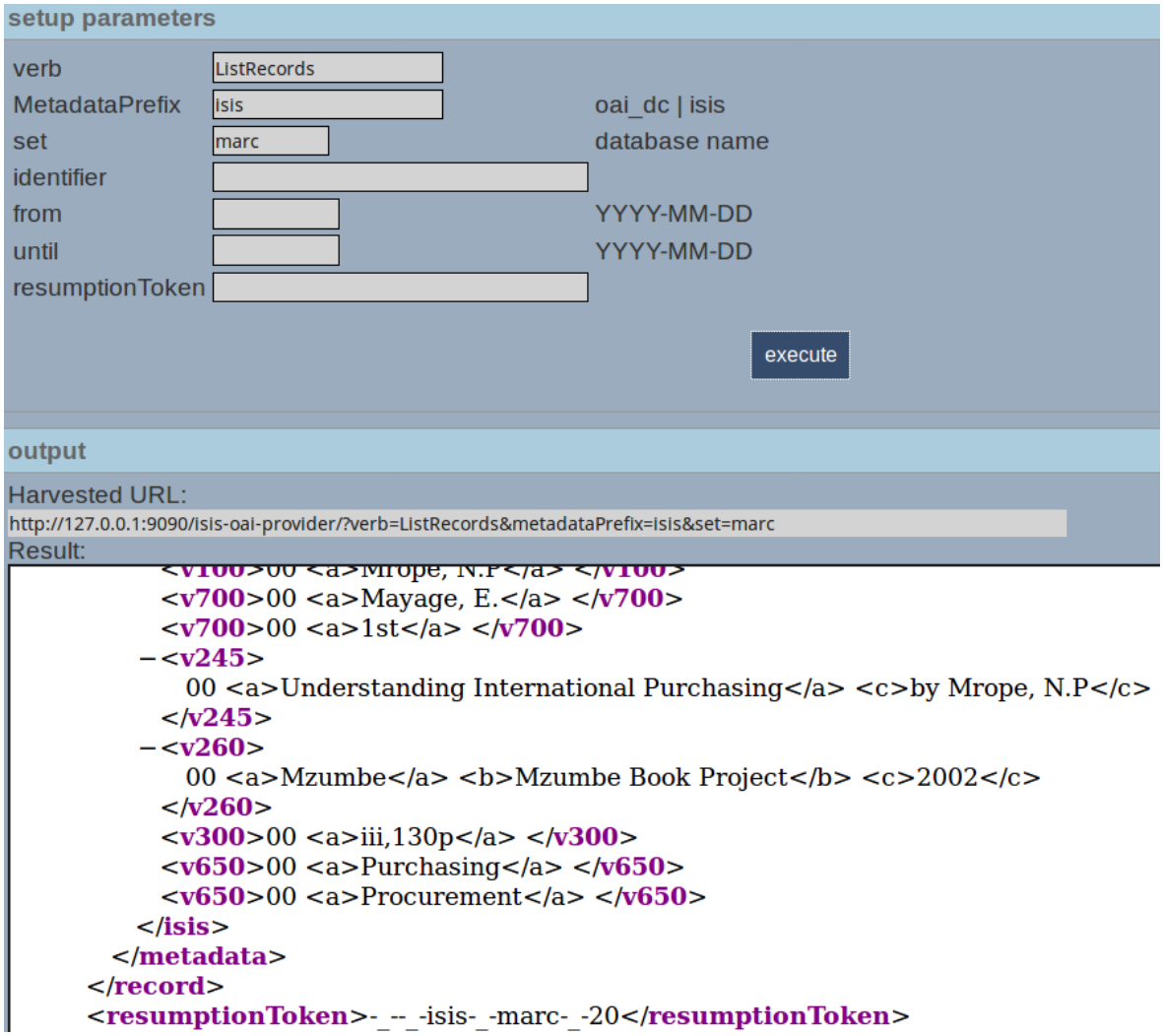
11.3.4. ListIdentifiers

This verb retrieves a list with the single identifier of each document published. Compulsory parameters verb metadataPrefix Optional parameters from until set Exclusive parameter resumptionToken retrieves a list with the single identifier of each document published. Compulsory parameters verb metadataPrefix Optional parameters from until set Exclusive parameter resumptionToken.

Depending on the field given in the database configuration as containing the identifier of the record, the records will be listed with that field :

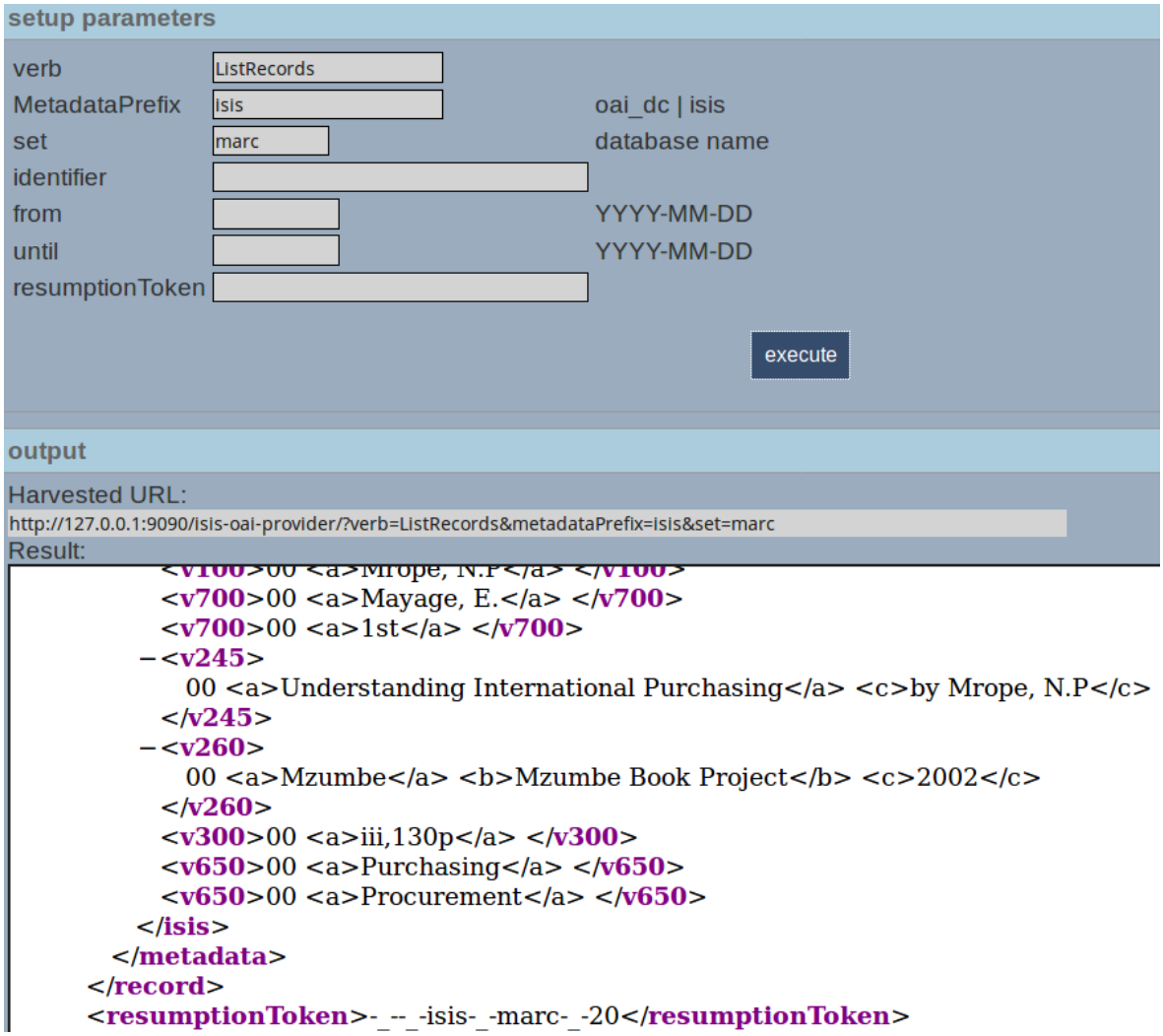


We are showing here the end of the output, because an interesting feature is visible there : the 'resumptionToken can be copied from there, to be filled in into the corresponding form-element in order to resume further listing with the given one as the first, so allowing subsequent batches to be retrieved (or 'harvested' in the IOAI-vocabulary).



11.3.5. ListRecords

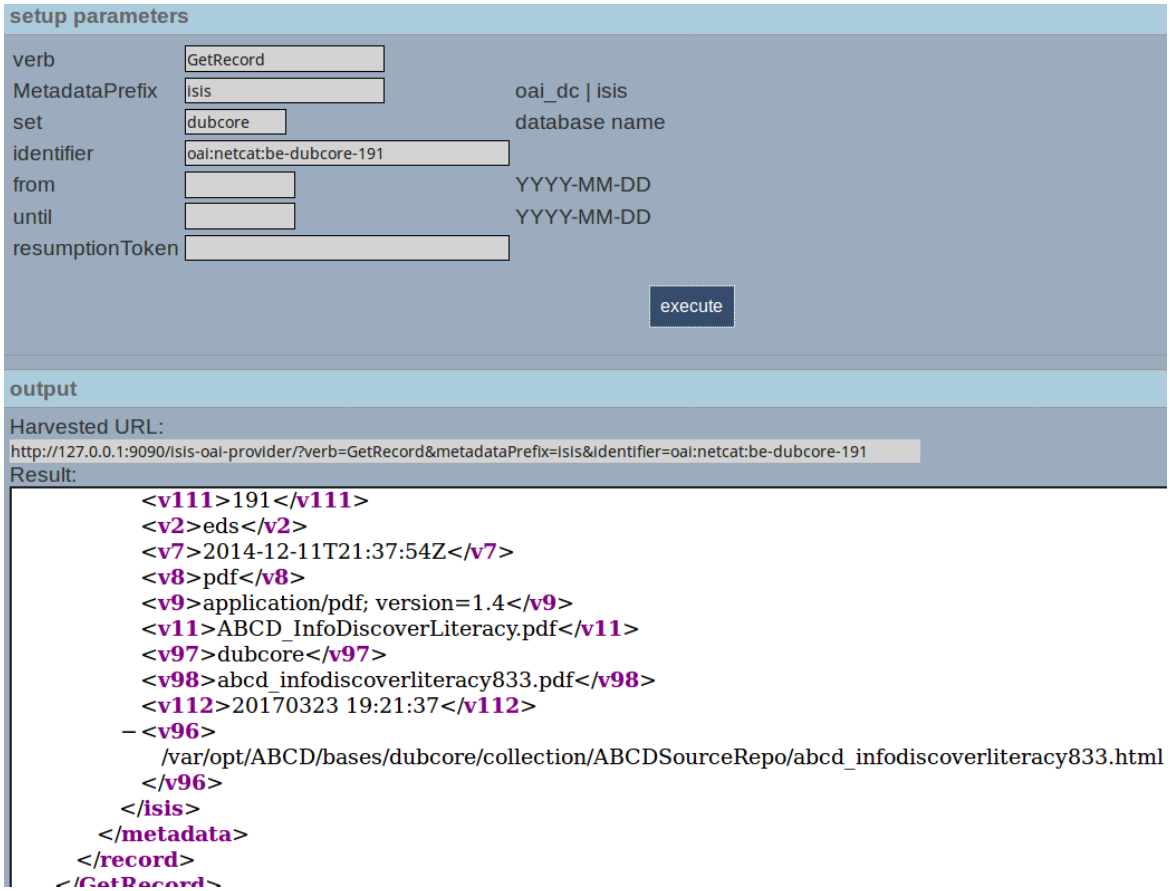
This verb retrieves the list of documents. Compulsory parameters verb metadataPrefix Optional parameters from until set Exclusive parameter resumptionToken .



As with 'ListIdentifiers' a resumptionToken can be used to continue listing more records starting from the last one shown before. Note that also 'from' and 'until' dates can be entered to limit the records listed to a given span of time.

11.3.6. ListRecord

Finally, the last 'verb' is to retrieve the metadata of a specific record. Compulsory parameters verb metadataPrefix identifier. Shown here is one record form the dubcore demo-database, using the dubcore_dc.pft 'mapping file' which is also part of the demo.



12. ABCD OPAC [THIS DOCUMENT IS AVAILABLE AS A SEPARATE PDF]

12.1. Concepts and files

12.2. the Site Editor

12.2.1. Philosophy of Components

12.2.2. Content managment

12.2.3. management of the Site

12.3. the Search Interface (iAH)

T

12.3.1. Configuration

12.3.2. Indexes

12.3.3. Help messages

12.3.4. Display formats

12.3.5. Plug-ins

13. ABCD Site [THIS DOCUMENT IS AVAILABLE AS A SEPARATE PDF]

14. ABCD Serials Control [THIS DOCUMENT IS AVAILABLE AS A SEPARATE PDF]

14.1. ISSN Standard

14.2. Concept of Kardex

14.3. Creation and edition of serial titles

14.4. Data entry issues

14.5. Configuration and templates

14.6. Union catalogues

14.7. Utilities: export/import, statistics, etc

Chapter 3. ABCD Unicode

1. ABCD Unicode and localisation

As from ABCD v2.0 both the software-interface as the data can work with non-Latin alphabet characters, based on the 'UTF8' standard of the Unicode-technology.

Unicode means that instead of the historically very limited 'ASCII-table' with 256 character codes (e.g. 65 for 'A', 66 for 'B' etc.), i.e. the basic interpunction (.,;-...), the numbers 0-9 and alphabetical characters in lower and upper-case only another limited non-standard set of characters could be defined. E.g. frequently used Greek characters like alpha (##) or sigma (##) were typically included in the encoding table of ASCII. Even with such 'available' space to redefine 128 characters in the non-standard part of the table (which was e.g. also used to make WinISIS capable of dealing with Arab), not all alphabets could be accommodated for (e.g. Chinese has many more characters), let alone combinations of alphabets. With Unicode the capacity of this table is enlarged to about 65000 characters, enough to include all known alphabets. In modern computer hardware and software storing such a table is no longer a real challenge.

Basically introducing Unicode to ABCD required making the (C)ISIS technology Unicode-compatible and add some functions (e.g. conversion from ANSI to UTF8 of text-files) in the interface. Both aspects will be explained in this chapter in the following two sections.

1.1. Unicode processing of the information

CISIS just stores 'characters' as bytes and in the original non-Unicode era (ASCII or its Windows-variant ANSI) each character was one single byte to represent it : single-byte. In order for ISIS to read the next character it simply needs reading the next byte. This changes drastically when multi-byte solutions as necessary for Unicode come into action : one byte can only represent 2^8 = 256 characters, so if more characters need to be encoded, more bytes will be necessary.

The UTF8 method was chosen for its backwards-compatibility with ASCII : UTF8 (but not UTF16 which always uses at least 2 bytes) allows single-byte encoding for the basic alphabets (Latin) and therefore ASCII-coded texts remain fully compatible with UTF8. This is of course very important for ABCD where such a wealth of ASCII-encoded information already exists and needs to remain available. However UTF8 or UTF16 can go up to 4 bytes for one character, so a sophisticated mechanism is needed to tell the software whether after reading the next byte still another (and another...) byte needs to be read in order to represent the next one character. These functions were implemented in the CIUTF8.c code of CISIS. Needless to state that these functions are very crucial since they are activated whenever bytes need to be read from a record.

Whenever a function of CISIS needs this multi-byte capability, such different mechanisms need to be used instead of the previous single-byte functions. In addition, the simple structure of the basic ASCII 'isisac.tab' just lists all up to 256 characters to be considered as 'alphanumerical' when ISIS is parsing data in order to identify 'words' which can be indexed for searching. For the upper-case translation table isisuc.tab two lists are needed : one with the lowercase values and another one with the corresponding uppercase values to which the lowercase ones will be translated. When - as it the case with UTF8 - characters no longer are defined with one single value (e.g. 65) they can only be defined with 'separators' in order to keep them grouped by character, so the UTF8 version of CISIS needs the ACTAB and UCTAB tables to use separate lines for each character, and each line can have, except for some optional 'comments' after a comment-separator '#', up to four columns.

So the original format of the ACTAB, which defines all characters considered to be part of a word for indexing, looks like this :

```
048 049 050 051 052 053 054 055 056 057 065 066 067 068 069 070 071 072 073 074 075 076
077 078 079 080 081 082 083 084 085 086 087 088 089 090 097 098 099 100 101 102 103 104
105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 192 193 194 195
196 197 199 200 201 202 203 204 205 206 207 209 210 211 212 213 214 216 217 218 219 220
221 224 225 226 227 228 229 231 232 233 234 235 236 237 238 239 241 242 243 244 245 246
248 249 250 251 252 253 255
```

whereas the UTF8 format now looks like (taking only some parts of the real table, the 'tab-signs' are represented to identify colums) :

065 # A

195 128 # A grave

216 128 # ARABIC NUMBER SIGN

224 164 132 # Devanagari #

224 182 133 #SINHALA LETTER AYANNA #

225 136 128 # ETHIOPIC SYLLABLE HA #

Note

1. decimal values are used while most Unicode-tables use hexadecimal coded values

Note

2. all values need to be listed in ascending order, otherwise the software will reject the table

In the case of upper-case translation a typical conversion for ABCD Unicode line will look as follows :

225 136 128=225 136 128 # ETHIOPIC SYLLABLE HA

Note

1. Most non-Latin alphabets don't have case-sensitiveness, so in this case (Amharic) the 'lowercase' is translated to the uppercase with exactly the same values. Such lines can as well be omitted and then no uppercase-translation will be performed.

Note

2. Adjustments in the FST are needed : since in non-Latin scripts the use of upper-case translation is not known in advance, UTF8-databases need to be indexed with specific instructions for the ’mode’-setting for uppercase, therefore : specifically add the instruction mode ’mpu’ (instead of mpl), ’mhu’ (instead of mhl), e.g. 245 5 ’/TI_/’,mpu, v245^a

Adding Unicode to ABCD mainly - or in fact 'only' - has relevance when indexing and searching a database : in these functions the 'bytes' stored in the MST need to make sense to indicate whether they are alphanumerical or not. But when indexing and searching is to be done, the differences in the basic tables of ASCII vs. UTF8 as illustrated above are very important, so a non-UTF8 version of ISIS cannot deal with a Unicode database properly and v.v. whenever indexing and searching are involved. In ABCD 2.0 both versions non-UTF8 and UTF8 are 'co-existing' for this reason, in the following way :

- the cgi-bin directory where the ISIS-executables are located has been completely re-organized : it now contains two basic sub-directories, one for ANSI and one for UTF8 and all executables are located there according to whether they are 'classic' (ANSI) or UTF8.
- all calls to the ISIS-executables (mainly wxis and mx) are now constructed in the PHP-scripts using a variable '$unicode' which defines whether the path /ansi/ or /utf8/ needs to be added to the cgi-bin folder when calling the executable.

Note

The same mechanism is used for the definition of special CISIS-versions, whether ANSI or UTF8, e.g. 'BigISIS' with the variable $cisis_ver

- in the base directory for databases (e.g. /var/opt/ABCD/bases in Linux) both the classic isisac.tab and isisuc.tab as well as the new UTF8-versions isisactab_utf8.tab and isisuctab_utf8.tab are stored. All PHP-scripts needing them (e.g. for indexing) will search for these tables according to whether the database is Unicode or not, first of all in the proper 'data'-subdirectory of the database itself, if not found in the base directory.

  ### Note

  This last mechanism allows for the use of adapted tables per database : if a catalog e.g. only uses in addition to Latin the Amharic alphabet, only the Amharic entries are needed in the table and no unnecessary large tables need to be loaded in memory and parsed.

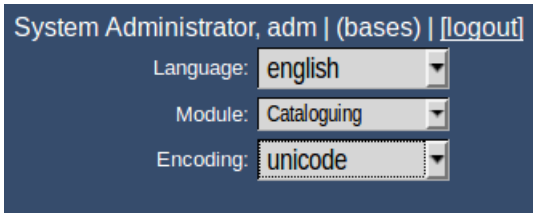
-

1.2. Unicode interface elements

After the explanation about the Unicode data-storage, the Unicode interface elements are discussed next.

1.2.1. General interface setting : Unicode or ANSI

In the Central main menu a (new) option '**Encoding**' is provided to select either ANSI or Unicode as the default charset for displaying ABCD interface pages. This only relates to the HTML-pages and text-files used in ABCD, not the databases and their contents. This is a system-wide setting, so it defines all pages of ABCD Central.

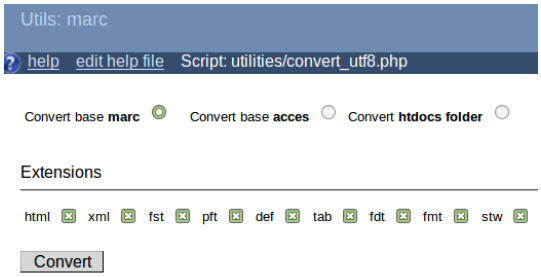


So when UTF8 encoding is used in e.g. configuration files like abcd.def, non-ASCII characters like diacritics, will be displayed wrongly unless the correct setting here is selected.

1.2.2. Conversion of text-files in htdocs and database-definitions to UTF8

When an existing ABCD (1.x) installation needs to be converted to a UTF8 environment, many files in the file-system of ABCD are encoded as 'ANSI', which will have consequences in how non-ASCII like diacritics are displayed - mostly wrongly ! Their encoding therefore needs to be converted to UTF8-encoding in order for such special characters (or any non-Latin characters by definition) to display correctly.

Tools in the extra ’utilities’-menu are provided to convert selected types of existing files (php-scripts, text-files, ISIS-formats…) from ANSI to UTF8 and vice versa. See also the discussion of the utilities in the chapter about ABCD modules.



An important thing to consider here is that as far as the databases themselves are concerned, only the database-definition files (which are text-files) will be converted, e.g. the FDT (so it can then use non-Latin alphabet field names), the PFTs, the FST, the stopwords-list etc. The database-files themselves, in fact the 'master' MST is not affected by this conversion. For the MST file with the actual data in the database, the following reasoning applies :

- if only ASCII-encoding was used, no conversion is actually needed, except for the indexes which need to be rebuilt

- if non-Latin characters were encoded e.g. with 'HTML-codes', a gizmo needs to be applied to convert these strings, which look like Unicode on the screen (because of the browser interpreting them correctly) but actually are not stored as UTF8. E.g. a string 'àé' could also have been stored with html-codes as 'ÀÉ' but the string 'À' will not have been translated to 'A' as per the ISIS-uppercase translation table (ISISUC.TAB) nor will be searchable as 'AE'. These two HTML-codes need to be converted to the UTF8 representation for 'àé' and indexed with the appropriate 'isisactab_utf8.tab' and if you want 'à' to be converted to 'A' for searching, the appropriate 'isisuctab_utf8.tab'.

- ANSI-databases can be exported to ISO2709 text-files and imported with the UTF8-version of mx; re-encoding the text-file itself to UTF8 will lead to problems because ISO2709 is very sensitive to the stored lengths of strings, so if a character which was stored in one byte is now stored in two bytes the ISO2709 file no longer can be read and reconstructed correctly.

The situation of still existing mixed ASCII, ANSI and Unicode charactersets is a problematic one anyway. E.g. the PHP-language for this reason has given up on becoming Unicode and skipped its version 6 which was meant to solve this... Characters displayed wrongly will be seen still for long on many screens, mostly in websites but even in subtitles for movies.

1.2.3. Dynamic adjustments of the selected 'charset'

The $unicode setting is taken from either the system-wide 'abcd.def' or the database-specific 'dr_path.def'.

1.2.3.1. ABCD Central

In HTML-pages an instruction can define which character-set (or charset) to use with the 'charset='-instruction. In ABCD Central some scripts therefore will check for the unicode-parameter and select a different charset accordingly. E.g. in the files htdocs/central/common/header.php and htdocs/central/common/display_header.php, which are creating the first part of the HTML-pages, the following code can be found :

```
if ($unicode==1) {
  echo "<meta http-equiv=\"Content-Type\" content=\"text/html; charset=utf-8\" />";
 }else{
  echo "<meta http-equiv=\"Content-Type\" content=\"text/html; charset=ISO-8859-1\" />";
 }
```

1.2.3.2. ABCD iAH

Also in iAh (the ABCD OPAC) pages can be dynamically adjusted to whether or not they use Unicode databases. For this the following elements are needed :

- DBN.def : a new parameter is added (at the end) : UNICODE=*n*, with AHINDX*n*.htm in iah/scripts/*lang*. When this parameter is set, i.e. is greater than 0, the value is put into the virtual field v5018^w, (with also an adjustment of the DBGIZMO in v5018^g) in the main script 'iah.xis', the actual script steering the whole OPAC. Different values, non-zero, can be used to activate different sets of alphabets, since the value set will define which file 'AHINDX.htm' will be used when displaying the buttons of the alphabets for end-user navigation into the related alphabet. E.g if UNICODE is set to 1, iAH will use the file AHINDEX1.htm of the related language folder in de scripts-directory, and this file can contain ’anchors’ to specific parts of the related alphabet. E.g. for Devanagari/Hindu :

```
<div class="rowCenter">
<input value="012..." name="indexRoot" type="submit"> <input value=" A" name="indexRoot" type="sub
[…]
type="submit"><input value=" Z" name="indexRoot" type="submit" />
<HR>
<input value="#" name="indexRoot" type="submit">
<input value="#" name="indexRoot" type="submit">
<input value="#" name="indexRoot" type="submit">
<input value="#" name="indexRoot" type="submit">
```

```
<input value="#" name="indexRoot" type="submit">
```

The resulting screen of the ABCD OPAC will then show, in addition to the Latin alphabet, alsom some anchors for Devanagari/Hindu characters :

By creating other files AHINDEXn.htm with other values for n (e.g. 2, 3 …) one can adjust this screen for any combination of special alphabets to be used.

- GIZMO databases : Since iAH uses optionally gizmo-databases, these need to be replicated in the proper version used for the main database searched, e.g. if the database is 1660utf8, then also the gXML-gizmo has to be transposed to 1660utf8 format, saved in a subdirectory of the gizmo-databasefolder or with a different name, and has to be referenced to accordingly (see next paragraph). If this is not done properly, 0 search results will be obtained.

  First the original 1660 gizmo-database has to be exported into an ISO-file and this then has to be imported to create the special database, e.g. creating the gXML gizmo for ffiutf8 :

```
mxffiutf8 iso=gXML.iso create=gXML_ffiutf8 now -all
```

  In order for the above mentioned gizmo-files to be found by the iAH OPAC, they need to be correctly referenced to in the .def file for the related database in the PAR-database directory, e.g. for a UTF8 database in the 'FFI' variant (ffiutf8) :

```
FILE gXML_ffiutf8.*=/ABCD/www/bases/gizmo/gXML_ffiutf8.*
FILE gXML=/ABCD/www/bases/gizmo/gXML_ffiutf8.*
```

- Dynamic adjustment of the correct charset in the iAH script htdocs/iah/scripts/*lang*/ahhead.pft, checking the above mentioned v5018^w to see if Unicode is to be used :

```
if val(v5018^w)>0 then
 '    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" /> '
 else
 '    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" /> '
 fi,
```

- Finally also the opening script 'index.php' for iAH needs now to check for the correct Unicode (and possibly also special CISIS-version) settings :

```
<?php
$base = ($_REQUEST['base'] != '' ?  $_REQUEST['base'] : 'rda');
$lang = ($_REQUEST['lang'] != '' ?  $_REQUEST['lang'] : 'en');
$form = $_REQUEST['form'];

//define database path according to OS
if (stripos($_SERVER["SERVER_SOFTWARE"],"Win") > 0)
 $db_path="/ABCD/www/bases/" ;
else
 $db_path="/var/opt/ABCD/bases/";

//unicode defined in abcd.def
$dbpath=$db_path."abcd.def";
```

```
$def2= parse_ini_file($dbpath);
if (isset($def2["UNICODE"]))  {
 $unicode=trim($def2["UNICODE"]);
 if (intval($unicode)!=0) $unicode="utf8"; else $unicode="ansi";}
else
 $unicode="ansi";

//cisis_ver and unicode defined in dr_path.def
$drpath= $db_path.$base."/dr_path.def";
 $def= parse_ini_file($drpath);
if (isset($def["CISIS_VERSION"]))
 $cisis_version=trim($def["CISIS_VERSION"]);
else
 $cisis_version="";
if (isset($def["UNICODE"]))  {
 $unicode=trim($def["UNICODE"]);
 if (intval($unicode)!=0) $unicode="utf8"; else $unicode="ansi";}
//echo "cisisver=".$cisis_version."<BR>";
//die;
if ($cisis_version!="")
$cisis_ver=$unicode."/".$cisis_version."/";
else $cisis_ver=$unicode."/";


//Path to the wwwisis executable (include the name of the program, in Windows add .exe)
$Wxis=$cisis_ver."wxis";
$hdr = "Location: /cgi-bin/". $Wxis . "/iah/scripts/?IsisScript=iah.xis&lang=" . $lang . "&base=" . str
header($hdr);
?>
```

1.2.3.3. ABCD Site

The ABCD Site module is mostly based on HTML- and XML-files to store the 'configuration' of the Site, defining the lay-out and contents of the main web-page of the Site and some sub-pages referred to from this page, e.g. warnings, explanations etc. If non-Latin is used in the website, make sure these files are encoded as UTF8, not as ANSI as is the case for the original ABCD-Site files.

While these text-files (.html and .xml in the 'site' database, for each language used in a sub-directory) can be edited directly, it is safer to use the Site Admin (e.g. http://127.0.0.1:9090/site/admin) CMS to now add the UTF8 path-parts (and similarly additional path-parts for special CISIS-versions) into the URL's of the metasearch-linked databases. E.g. the URL for a 'MARC'-catalog in UTF8 (MARCUNI) on a localhost ABCD-installation would have to change from :

```
/cgi-bin/wxis/iah/scripts/?IsisScript=iah.xis&lang=en&base=MARCUNI
```

to :

```
/cgi-bin/utf8/wxis/iah/scripts/?IsisScript=iah.xis&lang=en&base=MARCUNI
```

and the same additional 'utf8/' path-string has to be added to the other links for resp. the search- and 'show result' link URLs. Failing to do so will result in either a 'file not found' error (e.g. the previous wxis-executable is no longer found) or the database being searched by the wrong version of wxis, resulting in 0-results.

For a small cosmetic correction, in the file htdocs/site/xsl/public/metaiah/result.xslv of UTF8 ABCD installations, replace the string ' ' by :

```
<xsl:text disable-output-escaping="yes">&amp;amp;nbsp;</xsl:text>
```

2. ABCD Localisation

This section deals with the techniques and steps needed in order to create a 'localised' version of ABCD.

Localisation refers to the idea of presenting the software in a way suitable for the local users, i.e. using a language and alphabet but in some cases also 'style' of wording and presentation which is most suitable for the targeted audience. E.g. for a children's library one could imagine that all the mentioned elements : language, alphabet,

wording and presentation, are adapted to the main characteristics of the target users, being children who have another vocabulary, their own preferences re styling (e.g. more colours and funny icons) etc.

As an extension of this idea also the concept of a 'responsive' interface could me mentioned : the interface adapts to the device of the user. Generally this boils down to using other style-sheets when the user-device is a mobile phone with much more vertically oriented and smaller screens as compared to computers. Not everybody is fully happy with this concept, some even stating "don't do it", so we will refrain from going into this technique all the way. Let us simply refer to the idea that a lot of such 'responsive' behaviour can be facilitated by the creation of several style-sheets sets. For ABCD this has been done as a matter of fact.

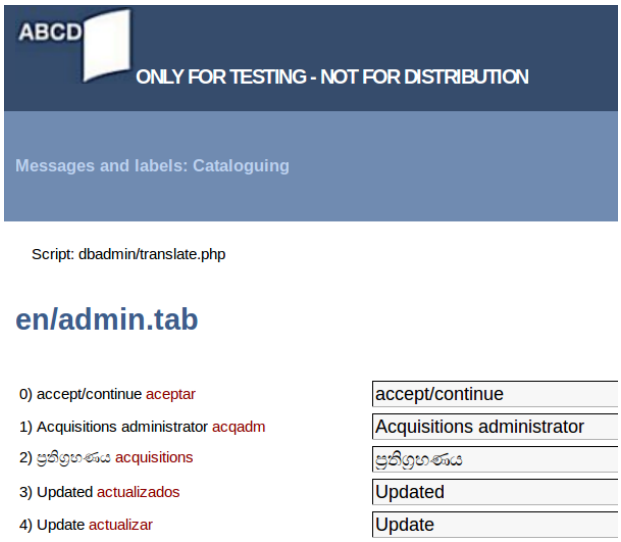
Here we will focus on creating a new interface based on the existing one but using a different - possibly new - language, if necessary written in a non-Latin alphabet. This not only needs to be done for the ABCD Central module but in fact more relevant for the other main end-user oriented modules iAH and Site. So when we 'localise' ABCD Central we consider the operators (librarians) also as end-users with their preference for a specific language, alphabet and vocabulary.

Since ABCD is a very much 'modular' system, each of the main modules has its own requirements and techniques to translate the interface to any other language. These modules operate independently (however accessing the same ISIS-databases) with their own software technology and also use different ways to display texts on the screens. E.g 'Central' uses text-files .tab with simple key-value pairs in the language-database-folder whereas 'Site' uses HTML- and XML-files. For the iAH OPAC a single text-file with most messages can be used with available scripts to immediately create (most of) the necessary text-parts in the required files (mostly php and pft scripts).

2.1. Localisation of Central

Except for some texts which could contain language-dependent elements, e.g. in the header and footers, like the institutional name, version, website links etc., all language-sensitive elements for messages are in 'tables', i.e. text-files with key-value pairs (key=value) saved by submodule (e.g. administration, cataloging, circulation…) in a file '.tab'. E.g. the file 'admin.tab' contains the messages of the Central-interface to deal with database-definitions. For each language these files are stored in a sub-folder of the 'lang'-folder in the bases-directory. This is not a real database-folder with ISIS-database files, like for other databases (e.g. MARC, CEPAL, USERS), but just the store of messages for each language used.

Under here we show how the same file admin.tab above will look like accessed from within the interface of ABCD Central; for illustration purposes we included a non-Latin entry (Sinhalese) already for the 3rd value (acquisitions) :



The first use of this interface therefore is to 'adjust' messages according to local taste or individual preferences. However the same technique can also be used to create a new language as follows.

2.1.1. Create new directories for a new language

Using the file-manager of your OS (either Windows or Linux) and navigate to the bases-folder, i.e.

- for Windows : \ABCD\www\bases
- for Linux : /var/opt/ABCD/bases

and locate the 'lang' sub-directory in the list of databases (each database has its own folder/directory). Enter into that directory and copy the one folder with the language you want to use as the 'from' (origin) language to be translated into a new one. E.g. if you want to create your new language based on the English, copy the folder 'en'.

Paste that folder back and (re-)name it with the code for your language, using preferably the ISO-codes for languages. E.g. to create a Sinhalese interface we will use 'si', so the original folder 'en' is copies and added as a new folder 'si'. In there all text-files still are English, by definition.

2.1.2. Add the new language in the languages lists

Still in your file manager, locate the file 'lang.tab' within your language folder and manually – e.g. with Notepad or Gedit – add your new language in the list, e.g. with Sinhalese the new list would look like :

```
en=English
es=Spanish
fr=French
pt=Portuguese
am=Amharic
si=Singalese
```

So if you use English as the default language for Central (see the file htdocs/central/config.php,in the line where the default language is defined, e.g. $lang="en"; for English), you have to edit the file bases/lang/en/lang.tab and add the new language as a new entry.

In Windows please check, esp. when using Notepad as the editor, that the extension of the file is maintained as '.tab' whereas Notepad will try to add .txt' to it as a new extension. ABCD will not recognize your languages list if it is not stored as lang.tab !

Now, up one level in the 'bases'-folder, another file 'lang.tab' is present which is used to list the available languages in the Central login-screen, where the previously edited file 'lang.tab' (in the language-specific subfolder) gives the display values (not the codes) for each language, so 'en' becomes 'English' etc. So here you simply have to list all language-codes you want to use, e.g.

```
en
fr
es
pt
am
si
```

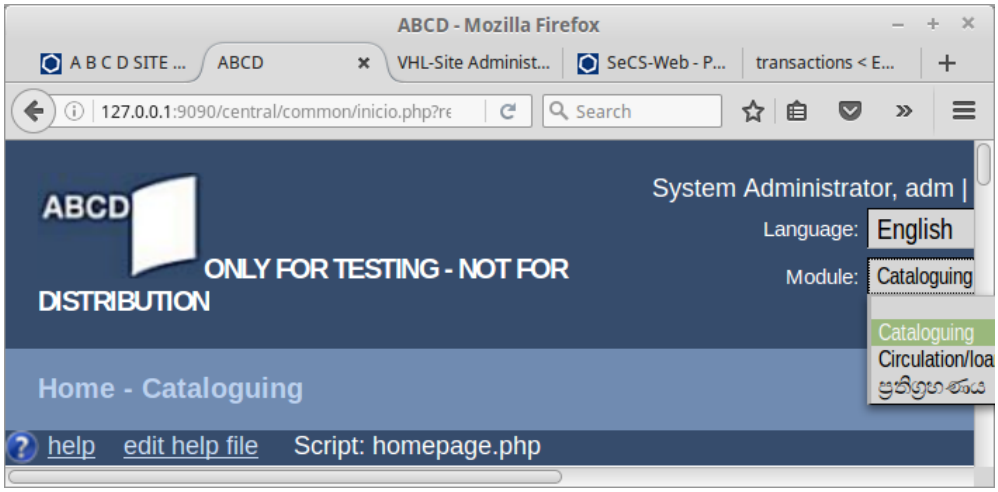
If you want users to be allowed to switch languages within Central, you should also add the new language in the 'lang.tab'-files for all active languages, so in their own sub-folders.

2.1.3. Translate all messages for all Central submodules using the interface

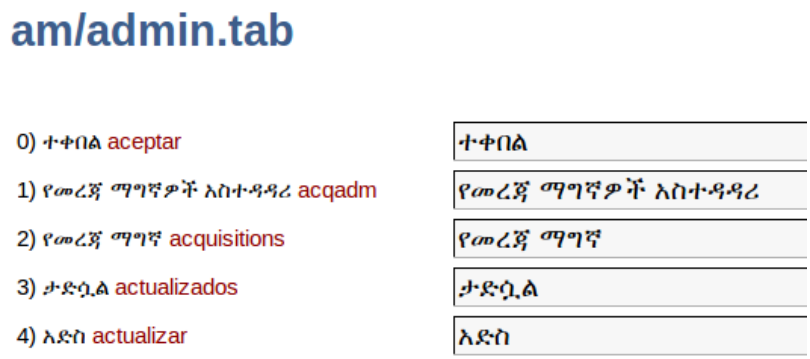
Now enter the ABCD Central interface into the language for which you have added the new language in the list. So the new language should appear in the languages-selection list of both the main login-screen and the language-selection list within the Central interface. Either way : make sure your new language is the active one. When then selecting the option 'Translate messages' and choosing one of the submodules, you will be looking at the 'origin'-language (e.g. English) at the left side fixed, and their values editable in boxes at the right side. So this is the big and main job : translate/transliterate all values to reflect as much as possible the correct language equivalent for the given message. This might require checking official glossaries e.g. maintained by your Ministry of Science and Technology, esp. for IT or even 'library science' in order to get the most appropriate authority files.

In the screenshot above one can see how the 3rd entry (no. 2) for 'acquisitions' has been translated into the Sinhalese word (and alphabet) '##############'.

Once the table is saved the next time (after refreshing the screens) the word 'acquisitions' in the interface will show as '##############' :



E.g. for the 'Amharic' interface (for Ethiopia) the same list of first few 'cataloging' interface messages will look like :



One can also, with an extra option in the ABCD Central main menu, 'compare' the existing translations in different languages to make sure the real correct meaning is taken. E.g.

The list of languages displayed in this overview should be manually edited in the script 'compare_admin.php' in the htdocs/central/lang sub-folder. Look for a section like :

```
if (isset($msg_path) and $msg_path!="")
    $a=$msg_path."lang/am/$table";
else
 $a=$db_path."lang/am/$table";
```

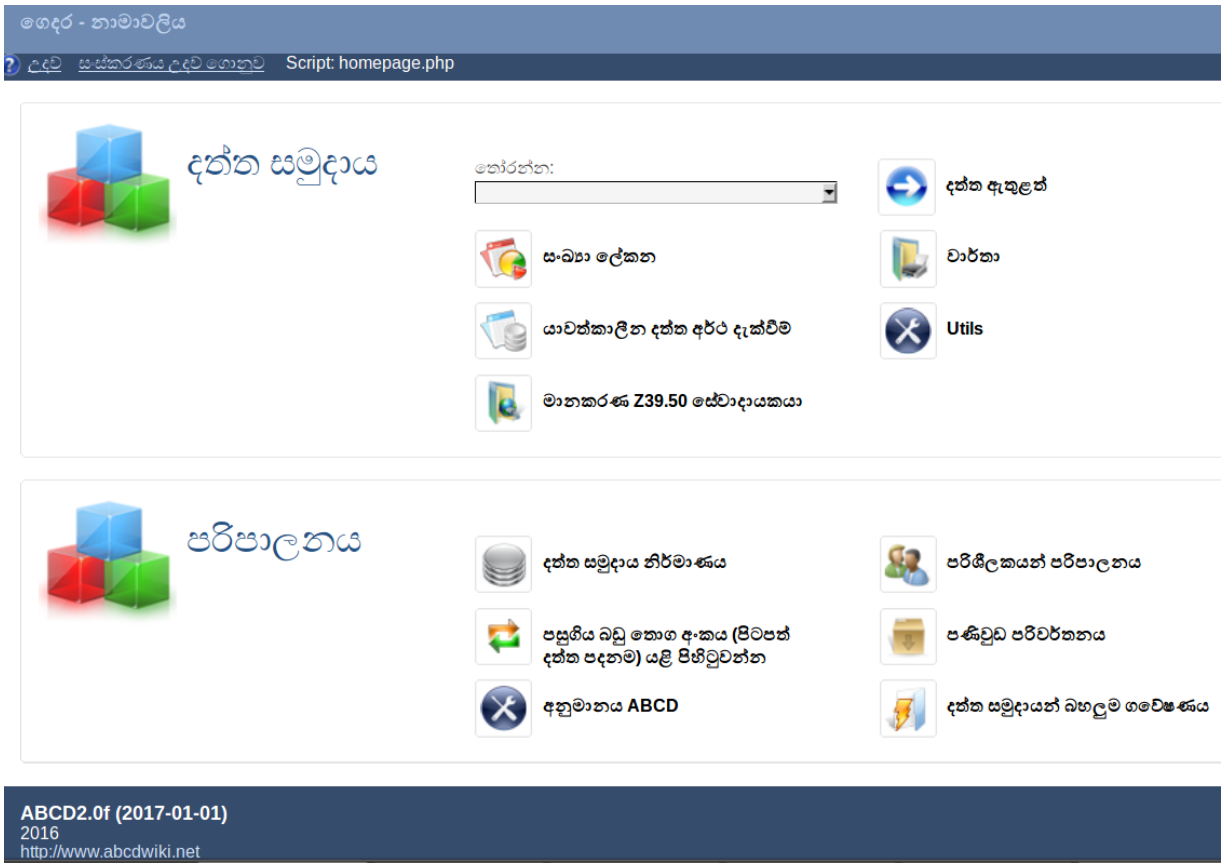
and change '/am/' (Amharic) for the language you want to see – at the right-most side – instead of Amharic, or any other previous language if so preferred. E.g. '/si/' .

Alternatively, once this mechanism with the underlying system of files and (sub-)folders is well understood, one could also directly edit with a text-editor all .tab files immediately within the file-system, without using the ABCD-interface. ABCD indeed is nothing more – but also nothing less – than an 'interface' to the multitude of files, scripts and databases of which the system consists.

Note

NOTE : when saving text files with non-Latin scripts, like in the examples above, make sure you save the text-files as UTF-8 files, not the mostly default 'ANSI' characterset. In Windows there is an additional complication : some software will want to add a hidden 'BOM' (binary order mark) as a few characters to indicate that the file is of Unicode-type. Since there has never been reached a real agreement on this – and quite some experts do not accept this to be a good solution – it is better to avoid this BOM additional code. If present the file won't be correctly processed by e.g. JavaScript or PHP as in ABCD.

When all messages of all modules/functions of ABCD Central have been translated, one can open the related language-version and the interface would look like for instance :



2.1.4. Translate all help-messages for the new language/interface

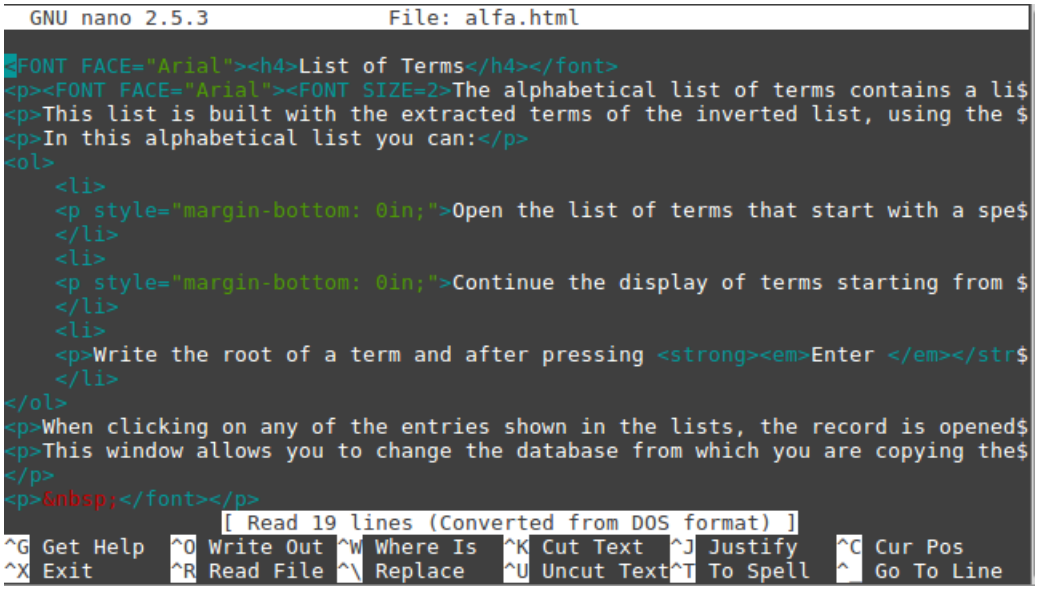
The ABCD Central help-pages are located as HTML-files in the bases-directory named 'documentacion'. For each language used a subfolder needs to exist.

So in the case of adding a new language (e.g. Sinhalese), copy the folder 'en' to a new folder 'si'. Then inside this new folder you will find all English help-files in their original English version.

Note

There exist sub-directories for some specific interface parts : acquisitions, circulation, copies and stats. All other helpfiles are directly positioned into the main subdirectory.

Then, using a suitable text-editor - try to use one which is more powerful than the dreaded Windows 'Notepad' but e.g. in Linux the basic 'nano'-editor will do well - translate the visible text-contents of the HTML-files. In case a good text-editor is used, these parts will have their own colour so as to make them quite easy to distinguish from other parts which should be left untouched as they constitute HTML-codes. Be careful ! As an illustration we show how the help-file 'alfa.html' for English is displayed in the Linux nano-editor : the while-coloured texts as the ones to translate into the new or local language.



2.1.5. Create language folders for each database

Since each database you want to use in ABCD also contains some language-dependent elements, e.g. the names of fields in the 'Field Definition Table' or in the display formats (PFT's), you should also copy/paste and rename an 'origin' language-folder to a 'destination' (your new language) folder in the bases-folders. Once done, editing the 'database definitions' in the ABCD-Central interface will be done on the correct language version for that database. E.g. for the MARC-database, the default catalog database for a library in ABCD, and a new language 'Singalese', do the following :

- copy the folder bases/marc/def/en as 'origin' to 'bases/marc/def/si as destination
- copy the folder bases/marc/pfts/en as 'origin' to 'bases/marc/pfts/si' as destination.

Then when in the MARC-database in the new language (Singalese) and using the ABCD interface to create/edit database definitions, you will be actually editing the files in bases/marc/def/si and bases/marc/pfts/si, so all language-dependent parts like field-names can be translated into Singalese.

As mentioned above, it is also possible to edit these files directly with a text-editor within the file-system without using ABCD, but then be careful because some understanding of the file-structure is necessary. E.g. deleting a pipe '|' - which serves as a 'column' separator in ABCD Central – by accident might spoil the whole file.

2.2. Localisation of the iAH OPAC

When checking the folder 'htdocs/central/iah', which contains all Central iAH scripts, one can see that again each language has its own sub-directory.

2.2.1. The general language elements in the subfolders of htdocs/iah

These are mostly html-files, e.g. for the help related to searching the database 'lilacs', there is a file note_form_lilacs.htm with contents :

```
<hr><font face="verdana" size="2"><b>Notas :</b><br><ul>  <li><font face="verdana" size="2">This option r
    <font color="Navy">title words</font>, <font color="Navy">words from the abstract</font>, and <font c
</font></font>
<li><font face="verdana" size="2">Use the truncation symbol <font color="#FF0033">$</font>(dollar sign)
     to search words with the same root. Example: <font color="Navy">educ</font><font face="verdana" size
retrieves educaci&oacute;n, education, educa&ccedil;&atilde;o, etc.     </font>
<li><font face="verdana" size="2">Do not type boolean operators (AND,     OR ou AND NOT) between words. <
<li><font face="verdana" size="2">
Select the option <font color="Navy">All words (AND)</font> to link the words (reduce the scope of the re
<font color="Navy">Any word (OR)</font> to add words (enlarge the scope of the retrieval).</font>
<li><font face="verdana" size="2">To search over other fields or to specify the field of the search use t
```

```
</font> </ul> </font><hr>
```

If one understands HTML-coding it can be easily seen where are the HTML-codes in between brackets < and > and where we have pieces of text to be translated.

If you want a new interface in iAH in another language, copy an existing 'origin' language folder and rename it under the new language-code. Then simply edit with a text-editor the pieces of text in your desired language/alphabet. Again when using Unicode scripts don't forget to 'save as' UTF-8 encoded file or use the 'encoding' or 'charactersets' option of your editor to assure the characters can be correctly stored.

2.2.2. The script-generated language elements

Within the htdocs/iah/scripts folder one will again note the existence of language-coded subfolders. In here iAH stores the HTML- and PFT- files or scripts it uses, with at many instances small language-related entries. While it would be possible to translate all such files in the same way as above, after having created a specific sub-folder for the new language by copying an 'origin' language folder, there are some scripts available in the folder 'htdocs/iah/scripts/translate', which works as follows :

- the 'translate' folder contains all the HTML- and PFT-templates used, as well as
- a simple key-value list for each language saved under the language code with .txt extension, e.g. en.txt contains all keys and values for English.

For creating the Amharic scripts of iAH the file 'am.txt' contains sections e.g. like :

```
database=### ##
database_search=### #### #####
config=###
```

again stored in UTF-8 format.

From the 'model' translation script 'translate2en.sh' (or translate2en.bat in Windows) easily a dedicated script can be generated for the new language, in the case of 'am'-haric it became :

```
#!/bin/sh
./mx seq=am.txt= "proc='d*a1~{{'v1'}}~a2~'v2'~'" -all now create=to_am
for i in *.pft
do
 ./mx seq=$i£ gizmo=to_am lw=9000 pft=v1# now > ../am/$i
done
for i in *.htm
do
 ./mx seq=$i£ gizmo=to_am lw=9000 pft=v1# now > ../am/$i
done
```

where basically the string 'en' was changed to 'am' and 'to_en' was changed into 'to_am'.

When running this script in a terminal (or CMD in Windows) some smart use of the CISIS mx-tool will read in the key-values text for your language and process the lines to add special brackets {{ and }}. Then these will be replaced, with a gizmo-parameter using the database created in the previous step, in the existing script-templates by their actual values, both for the PFT and HTM extension files.

The resulting scripts then are stored in the language-specific subfolder one level up in the file-system (from the 'translate' subfolder).

Some manual checking might remain necessary after this process, but most of the cumbersome work has been done intelligently by the above scripts using mx, the main CISIS-tool. So in ISIS-environments many problems can be solved by using the power of the software itself, albeit good understanding of the command-line tools will be necessary to create such solutions.

The list of languages to be used in the interface is derived from the file 'iah.def.php' in the folder htdocs/iah/scripts.

There is a line :

```
AVAILABLE LANGUAGES=pt,es,en,fr,am
```

clearly listing the codes for the languages (and their sequence, which is quite important as it needs to correspond to the list of subfields in the iAH-configuration files for each database !). In other words and in this example : since we added 'Amharic' as the fifth language for the OPAC, each configuration line in dbn.def needs to get a subfield ^5 added with the appropriate term for the language.

With the parameter

```
MULTI-LANGUAGE=ON
```

one can also switch if 'off' so as to only use the default language, set in the line

```
$lang = ($_REQUEST['lang'] != '' ?  $_REQUEST['lang'] : 'en');
```

of the file 'index.php' in the folder htdocs/iah.

2.3. Localisation of the ABCD Site

The ABCD Site again uses a different mechanism of using language-dependent elements. When one checks the folder 'htdocs/site' it immediately becomes clear no separate language subfolders are present as in iAH. Rather as in Central a 'database'-folder will be used, named 'site' even if as with Central there is no real (ISIS-)database involved : rather this base-folder contains XML- and HTML-files for each language.

So, as above with Central, one starts by copying/renaming an existing language-subfolder in the 'bases/site' folder, each for the XML and HTML-files.

Then inside the newly created folder each of the files – about 30 but small in size – need to be manually edited with a text-editor. This is a job to be done carefully so as not to 'touch' on any real XML- or HTML-tag.

The files are named with numbers, according to an intricate internal naming system managed by the Site-scripts. E.g. the file '22.xml' has this contents :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<warning id="22" lang="en" available="yes">
  <item available="yes">Warming<description>This VHL is under development </description>
  </item>
</warning>
```

If one understand XML one can see that only the parts 'Warning' and 'This VHL is under development' are to be translated, everything else is to be left untouched as it is actual XML-code.

A more illustrative file is e.g. 'metasearch.xml' which contains all labels used in the 'metasearch' interface :

```
<?xml version="1.0" encoding="iso-8859-1"?>
<metasearch id="" lang="en" available="">
  <text id="search_title" available="yes">ABCD Search</text>
  <text id="search_entryWords" available="yes">Entry one or more words</text>
  <text id="search_submit" available="yes">Search</text>
  <text id="search_howToSearch" available="yes">how to search?</text>
  <text id="search_allWords" available="yes">All words</text>
  <text id="search_anyWord" available="yes">Any word</text>
  <text id="search_error" available="yes">Please use a search expression.</text>
  <text id="search_advancedSearch" available="yes">search filter</text>
  <text id="search_freeSearch" available="yes">by words</text>
  <text id="search_results" available="yes">Result</text>
  <text id="search_method">method</text>
  <text id="search_demo">demo</text>
  <text id="conceptSearch_title" available="yes">by relevance</text>
  <text id="conceptSearch_entryWords" available="yes">Search documents more related to the concepts</text
  <text id="decs_mesh" available="yes">Search by DeCS/MeSH terminology</text>
</metasearch>
```

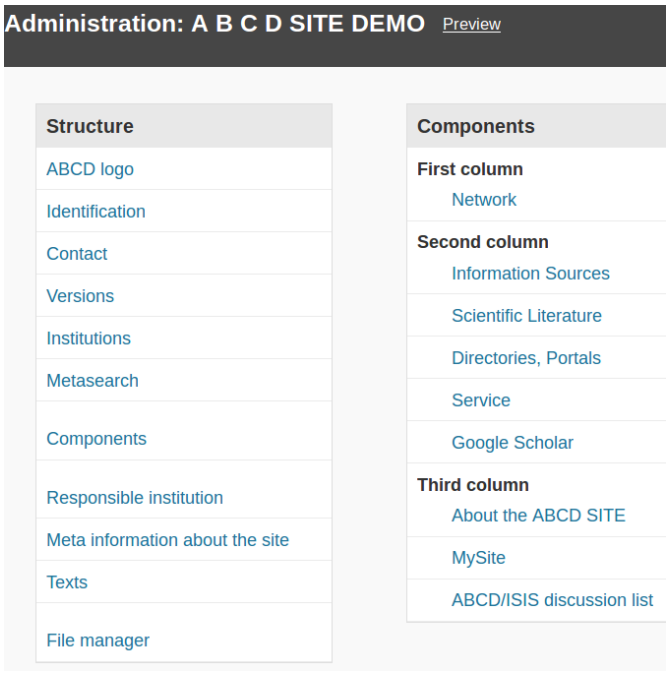
Here it should be clear that again not the parts inside < and > can be changed, only the actual texts as displayed outside the brackets.

Using a 'better' text-editor than the very basic 'Notepad' in Windows, e.g. Notepad++, or in Linux : Gedit, one can easily benefit from the syntax-recognition features of these softwares : whatever is plain text will be shown in a

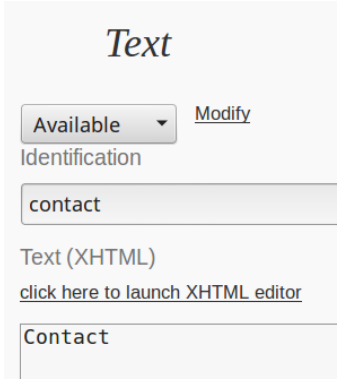
black color whereas anything 'code' (e.g. XML, HTML, PHP…) will have colors. E.g. in the following screenshot we show the same file 'metasearch.xml' for the ABCD-site when opened in such a richer editor, now all text-parts to be edited shown in black colour :

```
<?xml version="1.0" encoding="iso-8859-1"?>
<metasearch id="" lang="en" available="">
  <text id="search_title" available="yes">ABCD Search</text>
  <text id="search_entryWords" available="yes">Entry one or more words</text>
  <text id="search_submit" available="yes">Search</text>
  <text id="search_howToSearch" available="yes">how to search?</text>
  <text id="search_allWords" available="yes">All words</text>
  <text id="search_anyWord" available="yes">Any word</text>
  <text id="search_error" available="yes">Please use a search expression.</text>
  <text id="search_advancedSearch" available="yes">search filter</text>
  <text id="search_freeSearch" available="yes">by words</text>
  <text id="search_results" available="yes">Result</text>
  <text id="search_method">method</text>
  <text id="search_demo">demo</text>
  <text id="conceptSearch_title" available="yes">by relevance</text>
  <text id="conceptSearch_entryWords" available="yes">Search documents more related to the concepts</text>
  <text id="decs_mesh" available="yes">Search by DeCS/MeSH terminology</text>
</metasearch>
```

Finally some buttons which are really fully embedded in the Site interface need to be edited for your language using the 'Site Admin' (the CMS part of the Site) interface, selecting the option 'texts' from the main menu of the Site Admin module :



After selecting 'Texts' one can simply translate the option into the desired translated term :



Here the element 'contact' (the part of the screen where the Site will display contact-data) is simply translated as 'contact' but this could be done in another language/alphabet as well.

Within all other ABCD Site Admin editing boxes there are still many more elements which can or should be translated, like each 'structural component' (the 'components' option of the main menu) when created also gets a name which could be yes or no translated into a specific language.

The list of languages in Site is derived from the script 'functions.php' (in htdocs/site/adm/php), where the following section contains the list of languages to be used :

```
function loadLangs($xmlpath){
    $dirs = array();
    return array('es','en','pt','fr','am');
}
```

It is easy to see where to add/delete languages as listed by their codes in the array.

Additionally, ABCD needs a stylesheet named as 'style-xx' where xx is the language code, in the directory htdocs/site/css/public/skins/classic (which is the default 'skin' for the ABCD Site, but as in the VHL Site this can be changed so as to create 'local' or 'regional' skins). A dedicated language folder also needs to be created even if it only contains one empty file 'redefine.css'.

In htdocs/site/admin/defaultXML for the new language a new folder needs to be created and the texts in the file 'texts.xml' also translated with the new language added. However as this is really administration domain translation of all terminology might not be opportune, necessary or even possible.