adlib

WWWOPAC reference guide

Adlib Information Systems

Contents

Introduction

The Adlib Internet Server module enables you to make your catalogue available to the public on the internet and/or intranet. The visitor of your web site can use search forms and other web pages to search the catalogue directly. The Internet Server module mainly consists of the wwwopac.exe program and a web application called Adlib Internet Server.

Wwwopac.exe is an executable without a graphical user interface that, together with a web server works as a Common Gateway Interface-program. It takes care of the interaction between the web application and the database. This comes down to translating http search queries put together by the web application, to an Adlib search statement, the actual execution of that search in the database, and then sending back the search result as a list of records or terms in XML format.

The Internet Server web application transforms this XML search result with XSLT stylesheets to a piece of HTML, and combines this with other pieces of HTML which are produced by the aspx page and (.ascx) user controls present in there, which all program a separate piece of graphical interface (a button, a tab, etc.) of the entire HTML form. The HTML document further uses CSS styles to assign layout (colours, fonts, etc.) to the different graphical elements. The complete document is put together on the server, and then sent to the browser on the client which only needs to display the HTML page.

The logic of the Internet Server web application, which is necessary for the operation of the user controls and putting together the http search query, has been programmed in .NET, and is stored in compiled form in several dlls which you can find in the *bin* of *executables* folder. Therefore, you cannot change this code.
But since the visible part of the Internet Server web application, the HTML pages and forms which are displayed in the browser, is completely built up via user controls (representing independent pieces of HTML), you *can* adjust the layout of the entire web application. You can simply move user controls to change the structure of a web page, or change the user controls themselves to even further customize the web application to represent company logos and styles.

The wwwopac.exe is installed as an auxiliary program on an existing web server (IIS 5.0 or higher for Windows 2000, XP, or higher), and has to be prepared for use within your network or system by making some custom settings in the *adlibweb.xml* file. For the installation procedure of the Internet Server module and its system requirements, see the *Adlib Internet Server installation guide* on our web site.

This reference guide describes in general all functionality offered by wwwopac.exe (most recent version), and refers where applicable in brief to its implementation in version 3 (applies also to 3.2) of the Internet Server model web application.
In principle, you can build an entire web application of your own for wwwopac, but the Internet Server model web application is very flexibly and you can largely customize it to your liking, or you let AIS do that for you; so, using the model web application is more efficient and cost-reducing.
You can view the Internet Server model web application on:
`http://demo.adlibsoft.com/internetserver3` In the opened web site, click ? for a Help text about the functionality on the currently displayed tab.

■ Required skills

For the basis of learning to handle all wwwopac functionality, all you need is this manual. For more information about the application of external technologies and standards, you'll be referred to other sources when necessary.

In order to be able to customize the Internet Server model web application, knowledge of HTML, XML, XSLT, CSS and ASPX is required. This is because the search result from wwwopac is in XML format, and the web application uses XSLT and CSS stylesheets to apply a graphical layout to that data. The HTML web pages as a whole (of which the search result may be a part) are put together via ASPX. But this subject is not a part of this guide. AIS offers a training course to learn how to make substantial adjustments to the web application (which extend beyond the necessary settings during installation, of course).

1 Configuring WWWOPAC

1.1 Web configuration files

In a web configuration file you initialise the use of wwwopac. There are two variants: *adlibweb.xml* (one XML file), and www files (one or more text files with *.www* as extension, which are not being used for Internet Server 3 and higher). *Adlibweb.xml* was introduced with Internet Server 4.7 and is meant as a replacement for the www file which is now out-of-date. In stead of plain text in the www file, XML is used, which has the advantage that you can work with configuration groups. The www format is still supported by wwwopac 5.0 or higher, but the use of *adlibweb.xml* is preferred; for the application of OAI (see further in this guide), *adlibweb.xml* is mandatory.

In a web configuration file minimally a database path (`DATABASEPATH`) should be provided, and a database name (`DATABASE`), possibly followed by the name of a dataset separated by a greater-than sign (if that is not mentioned in the search query already).

In principle, you can also search without a web configuration file. All the information that is necessary for the search query, which you would normally enter into the web configuration file, must then be entered into the CGI request. But there are still a number of wwwopac commands that can only be used in a web configuration file (see the list of wwwopac commands).
We do advise you to use a web configuration file (like it is applied in Internet Server 3) if you want to guarantee the safety of certain data. Only those databases that you specify in the web configuration file can be accessed by the web application and by command-line search queries entered directly into the *Address* entry field of a browser. Without a web configuration file, visitors may be able to use command-line search queries and access all Adlib databases in the data folder on the web server.
So if you want to access a certain database without a web configuration file, you should refer to a database in the CGI request using the `DATABASE` variable, as you would normally do in the web configuration file. Wwwopac will first check whether this variable in a CGI request refers to a www file with that name, and if not, whether it refers to a database. With this, the `DATABASEPATH` variable is also available for use in CGI requests, because otherwise, without a web configuration file, you could not tell wwwopac where to find

the database.
When any www file or *adlibweb.xml* file is present, in a CGI-request no other databases can be accessed than the ones specified in the web configuration file(s).

If wwwopac cannot find a web configuration file (or none has been specified), then the program will start looking for the *default*.*www* file. And if this is not found either, you will receive an error report in your browser: *WWWOPAC ERROR: unable to find a default.www file.*

In a web configuration file, you can/must set a number of things. For this you can use wwwopac commands. These are commands that can sometimes be placed in the HTML-form (and with that also in the command line of the URL-input field of the browser) as well as in the web configuration file. These wwwopac commands are used by wwwopac to find the locations of the necessary files, and to set some parameters for searching and the retrieving of data.

For safety reasons, you can opt to place as many settings as possible in the web configuration file: contrary to the HTML-form, users cannot access this file. In the web configuration file, you can enter both absolute paths and relative paths . Preferably, only use one absolute path.

Below, you see an example of a simple *adlibweb.xml* file:

```
<?xml version="1.0" encoding="UTF-8" ?>
<webConfiguration>
<!--  Global settings -->
<globalConfiguration>
  <databasepath>D:\demo.adlibsoft.com\IS3\data</databasepath>
  <xmltype>structured</xmltype>
  <adlib_smarthost />
  <highlight>on</highlight>
  <maxlimit>400</maxlimit>
  <logfile>default.log</logfile>
</globalConfiguration>

<!--  Default group for library  -->
<groupConfiguration group="defaultLibrary">
 <brieffields>
  <field>lead_word</field>
  <field>title</field>
  <field>author.name</field>
  <field>corporate_author</field>
  <field>year_of_publication</field>
  <field>digital_reference</field>
  <field>copy.number</field>
  <field>shelf_mark</field>
 </brieffields>
 <detailfields>
  <field>lead_word</field>
  <field>title</field>
  <field>author.name</field>
  <field>corporate_author</field>
  <field>illustrator.name</field>
  <field>statement_of_responsibility</field>
  <field>edition</field>
  <field>place_of_publication</field>
  <field>publisher</field>
  <field>year_of_publication</field>
  <field>print.place</field>
  <field>print.name</field>
  <field>pagination</field>
  <field>illustrations</field>
  <field>dimensions</field>
  <field>material_type</field>
  <field>accompanying_material</field>
  <field>series.title</field>
  <field>series.number</field>
  <field>subseries.title</field>
  <field>subseries.number</field>
  <field>subseries.issn</field>
  <field>isbn</field>
  <field>binding_method</field>
  <field>parent</field>
  <field>child</field>
  <field>object.object_number</field>
  <field>keyword.contents</field>
  <field>geographical_keyword</field>
  <field>person.keyword.name</field>
  <field>copy.number</field>
  <field>shelf_mark</field>
  <field>site</field>
  <field>loan_category</field>
  <field>digital_reference</field>
  <field type="richText">abstract</field>
  <field type="richText">notes</field>
  <field>source.title.article</field>
  <field>source.title</field>
  <field>source.day</field>
  <field>source.month</field>
  <field>source.volume</field>
  <field>source.issue</field>
 </detailfields>
</groupConfiguration>


<!--  Database Full Catalogue   -->
<databaseConfiguration database="ChoiceFullCatalogue" groups=
 "defaultLibrary">
  <database>document>fullcatalogue</database>
</databaseConfiguration>
<!-- note: a databaseConfiguration tag may have only one groups
attribute -->
```

```
<!--  Database Books  -->
<databaseConfiguration database="ChoiceBooks" groups=
 "defaultLibrary">
  <database>document>book</database>
</databaseConfiguration>

<!--  Database Audio Visuals  -->
<databaseConfiguration database="ChoiceAudioVisuals" groups=
 "defaultLibrary">
  <database>document>avm</database>
</databaseConfiguration>

<!--  Database Articles  -->
<databaseConfiguration database="ChoiceArticles" groups=
 "defaultLibrary">
  <database>document>article</database>
</databaseConfiguration>

<!--  Database Serials  -->
<databaseConfiguration database="ChoiceSerials" groups=
 "defaultLibrary">
  <database>document>serials</database>
</databaseConfiguration>

<!--  Database Dublin Core  -->
<databaseConfiguration database="ChoiceDublinCore" groups=
 "defaultLibrary">
  <database>document>resource</database>
</databaseConfiguration>


</webConfiguration>
```

■ Virtual directories

To prevent web users from having access to the entire hard disk of a computer, web applications work with 'virtual' directories that are separated by forward slashes. A virtual directory refers to the real directory. You can create a virtual directory in the management of IIS 5; only assign the absolute required access rights to a virtual directory. Only the directory with *wwwopac.exe* needs execute rights . For the other directories, read-only rights are sufficient. You could make e.g. the virtual directory '*/adlib*' for the real directory: '*c:\bibl\wwwopac\wwwroot*' and assign read-only rights to it.
You can incorporate these virtual directories in a web configuration file instead of the real directories (then use forward slashes).

1.2 WWWOPAC commands

In a web configuration file you can use any of the commands listed below. These commands are not case-sensitive.

`<!-- comments -->`

This is how you insert comments in *adlibweb.xml*.

`ADLIB_PTR``=<path name to folder with pointer files>`

If you want to use wwwopac.exe to open pointer files that are not in the standard place in a sub folder of the *data* folder in your application, use this variable to enter the path name of the folder in which the pointer files are located.

`ALLOW_ALL_DBS``=true`

`<allow_all_dbs>true</allow_all_dbs>` lets all available databases be searched.
Normally, you would use a web configuration file in which every database that can be searched must be initialised. This offers security to those databases that may not be accessible through the web. But if this is not necessary, then `ALLOW_ALL_DBS` is a good way to bypass this security. `ALLOW_ALL_DBS` can only be used in *default.www* or *adlibweb.xml*.

So if you want to allow searching in all databases you either specify this in *default.www* or *adlibweb.xml* with `ALLOW_ALL_DBS`, or by not defining either *default.www* or *adlibweb.xml* at all.

`BRIEFFIELDS``=<taglist or field list>`

Use the `brieffields` variable to specify the fields you wish to see in the list display of the search result, for example:

```
<brieffields>
  <field>lead_word</field>
  <field>title</field>
  <field>author.name</field>
  <field>corporate_author</field>
  <field>year_of_publication</field>
  <field>digital_reference</field>
  <field>copy.number</field>
  <field>shelf_mark</field>
</brieffields>
```

You can mix tags and field names. So far as they are known by Adlib, tags are converted to field names in the XML-output. If you wish to include all the fields from the data dictionary in the brief display of records, you do not have to sum up all those fields; you can use:

```
<brieffields>
  <allDataDictionaryFields/>
</brieffields>
```

If you also want to include fields that have not been defined in the data dictionary, you can add the tags for those fields as normal to this variable, e.g.:

```
<brieffields>
  <allDataDictionaryFields/>
  <field>au</field>
  <field>ti</field>
</brieffields>
```

If no `brieffields` are provided, then `SCAN` uses the following tags:`%0`, `te`, `uf`, `us`, `rt`, `bt`, `nt` and `et`. For a normal search (searching without `SCAN`) all fields will be used (except the tags that are not identified as a field).

For optimal performance of Adlib it is definitely recommended to provide only the `brieffields` which you intend to actually use. This is because each submitted field will take up some processing time, and when you retrieve multiple records for a brief display, this will certainly add up.

DATABASE`=<FACS-name>`

Use this variable in the web configuration file to identify the name of the database. A variable with the same name can also be used in CGI-strings, but this then refers to the www-file at first, and only when that is not found, it refers to a database with that name. Example:
`<database>document>fullcatalogue</database>`

DATABASEPATH`=<disk station:>\<Adlib-folder>\data`

May occur in the web configuration file and in CGI-strings. This setting refers to a *physical* location, seen from the web server. For best performance of the web server, the data should be stored on a local disk of the web server. (It is sensible to only enter one absolute path in a web configuration file.) Example:
`<databasepath>C:\ourorganisation.com\IS3\data</databasepath`

DETAILFIELDS=<taglist or field list>

Use the detailfields variable to specify the fields you wish to see in the detailed display of the search result. You can mix tags and field names. So far as they are known, tags are converted to field names in the XML-output.

When no detailfields are specified, then all fields are used (except the tags which are not identified as fields). detail fields do not apply to SCAN.
For example:

```
<detailfields>
  <field>lead_word</field>
  <field>title</field>
  <field>author.name</field>
  <field>corporate_author</field>
  <field>illustrator.name</field>
</detailfields>
```

For optimal performance, it is recommended to provide only the detailfields which you intend to actually use.

HIGHLIGHT=<1, on, 0, off, or desired XML-tag>

Often it is desirable to mark in the search result the (partial) terms a user searches on in a web application, to make it stand out. This happens by displaying that (partial) term in, for instance, a different colour, the so-called highlighting.
Before wwwopac version 5.0.2 it was a time-consuming task to manage this functionality in a web application, because the Internet Server ASP script had special code to search the term in the contents of the retrieved (application-dependent) tags and lay out it through XSL.
Therefore, from 5.0.2 the implementation of this functionality has been simplified: highlighting has been built in to wwwopac itself. In a search query or configuration file you can switch this function off via HIGHLIGHT=0 and respectively <highlight>off</highlight> (by default highlighting is on).
The default setting puts a searched on (partial) term in the XML result between <highlight> and </highlight>. To have the Internet Server web application do something with these highlight tags, HIGHLIGHT=1 must have been set in the *globalsettings.xml* file as well.

In Internet Server 3, the style of highlighting is defined in the *highlight* class in the *adlib.css* file:

```
.highlight
{
  font-weight: bold;
  color: #cc1010;
}
```

You can easily adjust this to change the layout style.

If you cannot use the *<highlight>* XML tag for this purpose, for whatever reason, then you may specify another name for this XML tag when you switch highlighting on. You do this via the command `<highlight>desired_XML-tag</highlight>`. With this, you switch highlighting on and you define the XML tag at the same time, for instance: `<highlight>marking</highlight>`. In the XML result, the searched (partial) term will now be enclosed by *<marking>* and *</marking>*. Subsequently you have to adjust the *highlight.xsl* stylesheet accordingly.

For truncated searching there is a difference between the automatic highlighting of field contents from term and word indexes. After normal (right) truncated searching in term indexes, only the partial term at the beginning of the term is highlighted (so the part on which the term was found), not any further occurring partial terms. So for instance, if you search on `a*` and one of the terms found is *adrenaline*, then only the first *a* will be highlighted. For left truncating an analogous description can be given. When your indexes have been prepared for left truncation, and you search on for instance `*e`, and a found term is *adrenaline*, then only the *e* at the end will be highlighted.
In the search result from a word index however, every occurrence of the searched (partial) term will be highlighted.

Remarks:

- Searching an OR-combination in a word index of one field, only highlights the first (partial) term on which you search, not the second. For an extensive search on multiple fields, per field only the first (partial) term on which you search is selected for highlighting.
- Highlighting doesn’t work for numerical, logical, and date indexes.
- Highlighting only works with *equals* (=) or *contains* queries.

LOGFILE=`<path to logging file>`

Enable logging by entering a logging file with this variable, e.g.:

```
LOGFILE=../data/webstat.csv
```

The indicated file is created and grows with every normal search wwwopac.exe executes. The format of this file is comma-separated (csv) and can be created and saved anywhere. (Because it is a comma separated file, it can be opened by a broad range of programs, including Microsoft Excel.)

The information which is stored concerns the date and time of the search, the IP address* from which the search came, which database was searched, and the number of records that was retrieved.

* In the old two-layer structure, wwwopac had direct access to the IP address and logged it in the third field of the logging file.
In a three-layer web application such as Internet Server 3, wwwopac has no direct access to the IP address of the visitor, and logs in the third field of the logging file always the IP address of the computer on which this ASP application runs, because that is the machine that calls wwwopac. That is why you should adjust the ASP web application as follows: to every query that is sent to wwwopac, the web application must add `&callerIP=<IP address or DNS name of visitor>`. This so called *callerIP* is the last field in the logging file of wwwopac. (But wwwopac also keeps on filling the third field, although you can't use it in the three-layer web application.) If the *callerIP* is not sent included in the query, a copy of the third field will be logged, to ensure compatibility with two-layer web applications.

There is also a convenient way to have a new log file created automatically every month (or every day or each year), so that these files do not get too large. You can do this by literally including for example: `LOGFILE=log%m%Y`.txt in the web configuration file. When wwwopac would create a log file in April 2005, the name of this file would become *log042005.txt*. Every time an event is written to the log file, the current date will be checked, and as soon as a new month begins, a new log file will be created for that month. You can also use `%y` (with a small letter) if you only want to include the last two digits of a year. And `%d` may be applied to have a new log file created automatically every day. (Note that this *.txt* file will be filled like a *.csv* file.)

RUNADAPL=<myadapl>

In the web configuration file or in the CGI string you can use `RUNADAPL=<myadapl>` to execute a certain adapl right away, for instance for writing to the database.

TIMEOUT=<seconds>

With `TIMEOUT` you set the time after which the search query is stopped, regardless whether the search is still going on. This parameter prevents the computer from 'hanging' when there are problems with the search query. And searches that take too long are also aborted. The default is 300 seconds.

XMLADAPL=<path to pre-XML-generation adapl>

Before an XML-generation of the search result by wwwopac you can have an adapl executed per record. This adapl then applies to all types of search queries: `SEARCH`, `FLD`… etc., `SCAN` and OAI-requests. Do not let this adapl generate XML or HTML.
Such an adapl is useful when you are using OAI-requests, to edit or combine values from fields that do not have a real one-on-one relation with Dublin Core-elements, before they are presented on Dublin Core.

XMLTYPE=<UNSTRUCTURED, STRUCTURED, DIAGNOSTIC or RAW>

The Adlib wwwopac automatically generates XML-output, with which adapls are no longer necessary to convert a search result into HTML or XML.

Depending on the search method, an index or record list is generated in XML: `SCAN` produces an index (a list of keys), while a search produces a list of records. (By “search” we mean all search queries that do not use `SCAN`.)

XML can be executed in various types, depending on the type of data that is saved with it. You can specify this type in the web configuration file, or only in a CGI string too:

`UNSTRUCTURED`:	like adlwin.exe generates standard XML output;

`STRUCTURED`:	a type of thesaurus-tree. Groups, and internal and external links are arranged in a tree structure;
`RAW`:	the complete record is shown; any specified `brieffields` or `detailfields` are ignored.
`DIAGNOSTIC`:	This only returns a number of hits, not records. It can offer extra and more efficient functionality for web applications, such as showing the number of hits for various databases first, before users choose to access the resulting records of one of those databases.
`SCAN`:	The result of a `SCAN`, an index, is produced by the `XMLTYPE` with the same name, in a tree structure with only internal links. This also displays the tags `%0`, `te`, `bt`, `nt`, `et`, `us` (use), `uf` (used for), `sf` (semantic factor), `fv` (semantic factor for) and `rt` (related term).

These XML-types can be combined with one and the same style sheet.

1.2.1 Redundant commands

With the progressing development of the Internet Server web application, some existing wwwopac functions become redundant. Although these now dated wwwopac commands can still be used, they are no longer applied in the Internet Server 3 web application. It concerns the following commands:

*** or #**

You can place comments in a www-file (not in *adlibweb.xml*), e.g. to document the selected settings in the file itself. Start a comment line with a `*` or a `#`; both characters may be preceded by spaces, as these are ignored. Comments have to be placed on separate lines. So you cannot place a comment right behind a setting; place it on the line before or after.

`<spaces>`

You can add spaces to a www-file as you see fit (this does not apply to an XML file), to make the code more clear. During the processing of the file, all spaces are ignored. (Carriage returns are important, however.)

ADLIB_DIRi=<language-specific error report directory>

Use this to indicate which folder holds error report files in one language, in www file syntax for example: `ADLIB_DIR1=F:\adlib\dutch\errors`. A couple of files with standard names which you can place in this directory are *norecs.html*, *syntax.html*, *nofield.html*. Use the variables `NORECS`, `SYNTAXERROR` and `INVALIDFIELD` to refer to other than the default error report files if you have given them other than the standard names.

ADLIB_OUTPUT=<server disk station:>\temp

With this, a path could be provided to where temporary files were to be saved, and was only relevant in sorting. From wwwopac 4.7, sorting search results has become quicker though, because there is no longer need for a temporary file. This means that you no longer have to specify an `ADLIB_OUTPUT`-directory in the web configuration file.

BRIEFADAPL=<path to list-adapls>

This refers to an adapl in which the presentation of a record list is specified. Valid from the query by form and in the web configuration file; is used for nearly all selections, also after `SCAN`.

WWWOPAC automatically generates XML unless you specify a briefadapl. So only use a briefadapl if you want to generate HTML through adapls, not in an ASPX-environment.
For a possible data conversion of search results before they are automatically converted into XML, you can specify an `XMLADAPL`.

So when you specify a `BRIEFADAPL` in the web configuration file, the search results for list display are not produced in XML, but will be translated to e.g. HTML by the provided adapl. However, this `BRIEFADAPL` won't be used for a search query with which you retrieve a specific record, for instance:`%250=5`. Instead, the specified `DETAILADAPL` will be used. In this example, record no 5 will be retrieved and the result will be processed by the adapl for detailed presentation. (%0 is the tag for record number, and the hexadecimal value of the %-character is 25, preceded in the CGI-string by a %-character.

BRIEFSTYLE (see STYLE)

CONTENT-TYPE=<`text/xml or text/html>`

Specifies the type of content. It is only necessary to specify `CONTENT-TYPE=text/xml` when you use adapls that write XML. The default when adapls are specified is `text/html`. Adlib then assumes that the specified adapls write HTML. When no adapls are used, the default is `text/xml` because wwwopac without adapls automatically generates XML.

DETAILADAPL`=<path to detail-ADAPL>`

Valid from query by form and web configuration file. Refers to an adapl in which the presentation of a single record is determined.
(If an application is made in which the result is always 1 record, then the adapl for the list screen can be skipped by entering the name of the detail-adapl with '`BRIEFADAPL`'.)

Wwwopac automatically generates XML unless you specify a detail-adapl. So only use a detail-adapl if you want to generate HTML through adapls, not in an ASPX-environment.
For a data conversion of search results before they are automatically converted into XML, you can specify an `XMLADAPL`.

DETAILSTYLE `(see STYLE)`

INVALIDFIELD`=<HTML or XML-file>`

Indicates the HTML or XML-file you wish to show the visitor when the specified search query contains an invalid tag or field name.

LANGUAGE`=<language number>`

The `LANGUAGE` parameter has not become redundant, but was applied slightly differently in the www file of older web applications, namely as follows: `LANGUAGE` specifies the standard language that is used, and indirectly determines in which `ADLIB_DIRi` directory Adlib should look for error report files (i=language number); the default language is English (language number=0). In web pages however, you can offer the user a selection from various languages. With this selection, you can also take error reports from the corresponding `ADLIB_DIRi` directory by submitting the language number to wwwopac via the search query.

Behind `language=` you may also fill in a text instead of a number. You can replace the first four language codes in Adlib by the following terms:

0	`en` or `eng`
1	`nl`, `nld` or `dut`
2	`fr`, `fra` or `fre`
3	`de`, `deu` or `ger`

For each term different from these, the accompanying language code will be looked up in the web configuration file. (If that definition doesn't exist, language 0 will be used.) You create said definition by inserting for example the following assignments:

```
es=4
spa=4
```

Whenever you now use `language=es` or `language=spa`, Adlib knows you mean language 4.

Often used abbreviations for languages are laid down in ISO-standard 639-2 and 639-1 (see http://lcweb.loc.gov/standards/iso639-2/englangn.html). For definitions of language codes greater than 3 you are advised to use the standard language abbreviations, but that is not mandatory.

LISTADAPL

`LISTADAPL` is a synonym for `BRIEFADAPL`.

NOEXPIRES=TRUE

Specify `NOEXPIRES=TRUE` in the www file of your web application to prevent that an "Expires: 0" warning from the Internet Server is ignored in poorly functioning `POST`-requests from NetScape 4.x browsers.

Only set this option if a lot of your web site visitors use a NetScape browser and complain about problems with browsing to next or previous pages in search results, while your web application uses `POST`-requests.
Do not set this option by default, because it's mainly a workaround for this particular NetScape problem.
As an alternative to setting this option, you could use `GET`-requests in your web application, in stead of `POST`s. To avoid complete request URLs appearing in the browser, you should make the requests from a page in a frames page.

NORECS=<XML or HTML-file>

Indicates the HTML or XML-file you wish to show the visitor when the specified search query yields no records. If you do not specify a NORECS-file or if the path to the specified file is not found, wwwopac will generate a standard error report file for this purpose. The standard *norecs.xml* e.g., looks like this:

```
<adlibXML>
 <diagnostic>
  <hits>0</hits>
  <error>
   <code>0</code>
   <message>no records found</message>
  </error>
 </diagnostic>
</adlibXML>
```

Any UTF-8 header present in an XML error report file is recognized and skipped. (wwwopac will recognize Unicode encoded text files in UTF-8. This means that in e.g. *norecs.xml* diacritics etc. may appear.)

STYLE=<mylayout.xslt>

In older web applications, the location of an XSLT file was specified in the `STYLE=<mylayout.xslt>` command. The Internet Server web application used this style sheet to transform the XML search result to HTML, in www file syntax e.g.:

```
STYLE=layout.xslt
```

Instead of `STYLE`, use `BRIEFSTYLE` and `DETAILSTYLE` if you wish to specify different styles for the brief display and the detailed display.

These three parameters may appear both in the web configuration file and in the CGI string.

When using one of these three parameters, the style sheet itself has to generate special output if the query doesn’t yield any records. (The *norecs.html* file is not used in this situation.)

SYNTAXERROR=<HTML or XML-file>

This indicates which file you want to show the visitor if the specified search query contains a syntax error.

TRANSFORMED=1

In older web applications, `TRANSFORMED=1` (in www file syntax) meant: transform the XML search result on the server. If you did not make this setting, the value behind `…STYLE=` would have been transmitted directly to the web client, and therefore had to be a complete URL (otherwise the client would not be able to find the path). Then the web client is responsible for the transformation, something not all clients are capable of. In Internet Server 3, `TRANSFORMED=1`, has become redundant: all transformations automatically take place on the server.

This parameter may appear both in the web configuration file and in the CGI string.

When you want to make HTML when using `transformed=1` (which will often be the case), then in the web configuration file you'll have to set `content-type=text/html`, and the XSLT output will have to start with `<html` or `<!doctype`. If it is not done this way, the output will be interpreted by the browser as XML.

By the way, the `TRANSFORMED=1&DEBUG=1` combination is not supported.

1.2.2 Switching off wwwopac functionality selectively

From wwwopac version 6.1.0 you can switch off wwwopac functionality selectively through settings in your web configuration file. This gives you more control over which information can be requested through this software and in which way, and thus enforces your security policy.

The following two commands for this purpose can be used as many times as needed: `allow` and `disallow`.
Per database (a *databaseConfiguration* section in the web configuration file) you can set different permissions. Make the default settings in the *globalConfiguration* section.
So per *databaseConfiguration* section you can give multiple `allow` and `disallow` definitions. This is necessary because you may assign some twenty different values to these two commands, and because the list with permissions will be executed cumulatively in the order of listing. This way, you have a lot of control about what is allowed and what isn't. Take for instance the following definition list:

```
<disallow>all</disallow>
<allow>search</allow>
<allow>scan</allow>
```

This listing allows users and/or the web application to search normally and to perform an index scan, but nothing else.

The functionality values that you may assign to the commands, are the following:

- **Status** - requesting server information (via `?status=1` in the CGI call).
- **Scan** - searching through an index, which results in a list of search keys.
- **Search** - normal searching, without `Scan`. This results in a list of records.
- **OAI** - OAI 2.0 functionality.
- **SRU** - the CQL-SRU 1.1-protocol.
- **SDI** - SDI functionality (only relevant for some web applications).
- **CheckPerms** - for ADW. Only when you use the ADW protocol, to approach an Adlib database, it is necessary that you `allow` all "for ADW" functionality, otherwise it won't work. The meaning of these ADW functionality values, won't be explained here.
- **LoadFile** - for ADW.
- **Locking**- for ADW.
- **CheckIndex** - for ADW.
- **PFile** - for ADW.
- **FreeList** - for ADW.
- **Sync** - for ADW and for writing of records in web applications in which that is possible.
- **RunADAPL** - being able to start an adapl (only relevant for some web applications).
- **Thumbnail** - displaying thumbnail images (for web applications).
- **ImageMetadata** - requesting image metadata (for web applications).
- **Delete** - deleting records (relevant for some web applications).
- **Test** - use for internal test purposes by Adlib.
- **All** - all above functionality together.

If you do not set `allow` or `disallow`, then the default permission applies, which implicitly is equal to:

```
<allow>all</allow>
<disallow>test</disallow>
<disallow>sync</disallow>
<disallow>delete</disallow>
```

This is fairly safe, though users can request quite some information through `Status` or `Checkperms` for instance.

Also note the following:

- These settings are not case sensitive.
- When a function is switched off, and the user tries to use it anyway, then wwwopac produces an error message 19.
- If you use these permissions in your web configuration file, you can remove any occurrences of the settings `WRITE_ALLOWED`, `SYNC_ALLOWED`, and `DELETE_ALLOWED`, since these have become obsolete. These three options are still supported nevertheless, so you can leave them in the web configuration file, if you don't want to use `allow` or `disallow`.
- If safety is important in your implementation of Adlib Internet Server, you are advised to use the following settings:

  ```
  <disallow>all</disallow>
  <allow>scan</allow>
  <allow>search</allow>
  <allow>thumbnail</allow>
  <allow>imagemetadata</allow>
  ```

1.2.3 type="richText" in <field> tags

In *adlibweb.xml* with Internet Server 3, each listed field in `BRIEFFIELDS`, `DETAILFIELDS` and `OAIFIELDS` (see chapter 6 for more information about OAI) is separately enclosed by XML `<field>` tags. This has the advantage that you may insert the XML tag attribute `type="richText"` for specific fields, to copy line breaks in the contents of a those fields (\n) as XML line breaks (`<br/>` tags); this way, all text from that contents is not concatenated in the same line, as would normally be the case.
Note that normally you will only want to use this attribute for non-RTF fields: if you would use it for RTF fields then RTF codes from the field contents would be included in the wwwopac.exe search result, but by default those codes are not recognized by the Internet Server as being

layout characteristics and therefore the codes will be displayed as if they were part of the text.

An example of an adjusted field list:

```
<OAIFIELDS>

      <field>be</field>
      <field>ob</field>
      <field type=”richText”>av</field>

</OAIFIELDS>
```

To include all fields from the data dictionary in such an XML structure at once, and apply `type=”richText”` to them, you can use the `<allDataDictionaryFields/>` tag, for instance:

```
<OAIFIELDS>

      <field>be</field>
      <allDataDictionaryFields type=”richText”/>

</OAIFIELDS>
```

In here, `be` is a tag that is not defined in the data dictionary.

Note that `<br/>` tags in the XML search result must still be handled by a style sheet to actually create a line break in the layout. To this end you may add the following template to a stylesheet:

```
<xsl:template match="br">
  <br/>
</xsl:template>
```

1.2.4 Adlib error reports

Any error messages from wwwopac that normally are displayed on the screen of the server and require a confirmation of that message by clicking *OK*, are automatically intercepted and reported in the XML search result in the `/adlibXML/diagnostic/error/info` node, so that the web application on the screen of a user on the internet cannot apparently freeze because there the error messages are not displayed on screen.

2 Submitting a search query

2.1 Commands and parameters in the CGI-string

The Adlib wwwopac executable is a server-side CGI program. This means that CGI-strings with an embedded search query (on the client-side) are sent from the browser to the server through the 'application/x-www-form-urlencoded' media type (MIME-type); this is also the standard encoding that is generated by HTML-forms .

CGI stands for Common Gateway Interface. This interface makes it possible to transport data between a web server and a CGI program. A CGI program can be any kind of program designed to accept and process data complying with CGI-requirements.

In principle, a CGI-string is put together (invisibly) by the Internet Server web application, using the search terms entered into an HTML-form or through a mouse-click on an (image) hyperlink to known records. (This is done in compiled code in which you can't make any adjustments.)
A CGI-string can also be typed directly into the URL-input field of the browser. (Then certain characters, like spaces and quotes, do need to be replaced by an escape code; see the bottom part of the following list.) Doing so, you can deep-link to the Internet Server 3 web application itself, for instance, using the following syntax:

```
http://<webdomain>/<virtual_directory>/dispatcher.aspx?action=
search&database=<database_name>&search=<search_query>
```

For example:

```
http://demo.adlibsoft.com/InternetServer3/dispatcher.aspx?action=
search&database=ChoiceFullCatalogue&search=priref=28
```

But you can also submit your search query to wwwopac directly, thereby working outside of the Internet Server web application. Then the XML search result won't be transformed/processed to a nice web page. The syntax then becomes:

```
http://<webdomain>/<virtual_directory>/<wwwopac-
folder>/wwwopac.exe?database=<database_name>&search=<search_query>
```

For example:

```
http://demo.adlibsoft.com/InternetServer3/wwwopac/wwwopac.exe?
database=ChoiceFullCatalogue&search=ti=ko*
```

or

```
http://demo.adlibsoft.com/InternetServer3/wwwopac/wwwopac.exe?
database=ChoiceFullCatalogue&FLD1=ti&VAL1=ko&TRC1=on
```

So by default the "CGI"-string is made up of a URL that refers to an aspx file of the Internet Server web application, or directly to *wwwopac.exe* (CGI), and a search query which may either be built up according to the syntax of Adlib's *Expert search system* search language*, or via an alternative syntax be compiled of (field)name/value-pairs (which is not being used in Internet Server 3, by the way). (See paragraph 2.1.1 and the description of "=" below, for more information about the different syntaxes.)

* Special search language syntax, like the comma, + or - as a short notation for OR, AND or AND NOT combinations, is accepted by *wwwopac.exe* but not by *dispatcher.aspx*. However, for such special syntax there is always a simpler equivalent which is correctly handled by *dispatcher.aspx*. A good way to obtain such a simple equivalent is to actually execute your query in the *Expert search system* of an Adlib application: Adlib will automatically change a special search query in a more elaborate search statement in a simpler syntax. Then use that search statement as the alternative for use with *dispatcher.aspx*.

In general you have the following possibilities:

Reserved words, characters, variables and commands	
=	Use this to assign a value to a variable, for example: `database=ChoiceFullCatalogue` or `FLD1=au` or `SEARCH=ti=computer*`. A name/value-pair, as in the second example, consists of a reserved word or variable, followed by a logical index number (i), an =-character, followed by the parameter or value, e.g.: `FLD1=keyword` (the `keyword` field name is assigned to the `FLD1` variable), or `VAL1=bicycle`. The index number indicates which pairs belong together; so the order in which the pairs occur in the CGI-string is not important. This syntax is not used in Internet Server 3. The syntax used in the Internet Server 3 web application is: `tag=value`. Tags are case sensitive. You cannot use field names in this syntax. In this syntax, the search query must begin with `SEARCH=`. In this case, the rules for constructing extended search queries are the same as in Adlib's *Expert search system*; so you can use

	brackets and Boolean operators (without index number) like normal in the search language.
?	Separate the URL (to the server-side .aspx file, wwwopac.exe or adlibweb.dll) from the search query by a question mark.
&	Separate name/value pairs with an ampersand.
BOOL	A search query in the name/value-pairs syntax may be extended with `BOOLi`. Assign (through =) one of the Boolean functions as a parameter: `OR`, `AND`, `WHEN` or `NAND`. (`NAND` stands for 'and not'.) `AND` is the standard operator when none is assigned.
DATABASE	Use this in the CGI-string to refer to the name of the database in which you want to search. This name must have been set in the *adlibweb.xml* file as well. Use this variable directly behind the question mark in the CGI-string. (In older web applications, the same variable can also be employed to indicate the www-file, the predecessor of *adlibweb.xml*, to be used. Then do not add the extension .www to the function call of the www-file.)
DEBUG	Enable (`DEBUG=1`) or disable (`DEBUG=0`) debug-mode; very useful in application development. The default for `DEBUG` is off.

DOM	You can use the domain variable `DOMi` to specify the domain that has to be searched in the current index, e.g.:
	`FLD1=name&VAL1=simmons&DOM1=author`
	
	You can also search on domains without using `DOM`. For this, include the domain behind `VAL` in the search string, followed by the searched term, and separated by two colons, for instance:
	`FLD1=name&VAL1=author::simmons`
	
	If truncation is on, you can also provide a domain with `DOM`, without entering a `VAL`; all terms in that domain of the index will then be retrieved.

FLD	Use `FLDi` to specify the (database) field name you want searched with value `VALi`. Adlib database tags (two-character field codes) are also admitted.
	
	It’s also possible to search simultaneously on multiple fields without having to make combined indexes first. You can do this by providing multiple fields or tags, separated by commas, in your CGI string.
	The command `FLD1=title,author,keyword` for instance, searches the title, author, and keyword fields simultaneously. Actually, these multiple search queries are converted by wwwopac, to an Adlib search query with ‘or’ statements:
	
	`http://myhost/myapp/wwwopac.exe?FLD1=`
	`title,author,keyword&VAL1=captain`
	
	results in the following Adlib search query: (*title=captain or author=captain or keyword=captain*).
	
	Although search time increases, you can search all types of fields this way, indexed or non-indexed, term indexes and word indexes.
	
	By the way, this name/value-pairs syntax is not being used in the Internet Server 3 web application.

LIMIT	Specify how many records of a search result will be retrieved and shown each time, e.g.: `LIMIT=10`
MODE	With `SEARCHMODE` you specify the default way in which multiple terms in one entry field are combined. If you want to specify a divergent search mode for one or some entry fields, then use the variable `MODEi` in the CGI request to set a search mode per `FLDi/VALi` pair. To `MODEi` apply the same possible values as to `SEARCHMODE`.

OCC	Indicate which occurrences you want to sort on with `OCCi`: possible values are `All` (sorting is done on all occurrences: this record will be placed in the search result as many times as there are occurrences) and `First` (only the first occurrence of a field is included in the sort).
OPR	Use `OPRi` to specify an operator, unlike a Boolean: `contains, generic, narrower, %3E or >, %3C or <, %3E%3D or >=,` and `%3C%3D or <=`. Note that the hexadecimal value should be used (or `&lt;` and `&gt;` for resp. < and >) for the hard coding of a search query in HTML. (Because < and > are reserved characters in HTML.) `contains` only works in the search form .
SCAN	Generate an index (list of keys) with `SCAN=field_name_or_tag`. This may also be integer and date fields. Some pre-defined tags and reserved variables* with regard to the retrieved records will temporarily be available for use in adapls. In the standard XML-output, the values from those variables are placed between XML-tags.
SCANDOM	Use `SCANDOM` when you are using `SCAN`; `SCANDOM=domain_name` limits the search with `SCAN` to a specified domain.
SCANEND	You can use `SCANEND` to specify up to which value you want to search (with `SCAN`), for example: `SCANEND=value`.
SCANUNIQUE	`SCANUNIQUE=1`: by default, so without setting this option, `SCAN` filters all double terms from an index before it is displayed. Although the name of `SCANUNIQUE` can be somewhat confusing, with this option you specify that filtering of double terms should *not* be done; all terms will be retrieved from the index. (At the moment, this option is only used in ADW applications.)

SCANVAL	Specify the value with which the list of terms will start, with `SCANVAL=value` (automatic truncation). If you do not specify anything, the list will start with the first value in the index. Some pre-defined tags and reserved variables with regard to the retrieved records will temporarily be available for use in adapls. In the standard XML-output the values from those variables are placed between XML-tags. But you can also use `SCAN` without `SCANVAL`. `SCAN` and `SCANVAL` do not work in combination with `SEARCH`.
SCANXMLFORCE	`SCANXMLFORCE=1`: when you perform a `SCAN`, you normally have no choice as to the `XMLTYPE`: the XML is formatted in a 'scan xmltype' in which you mainly find metadata. For some applications (at the moment only ADW), it is an advantage if a `SCAN` result contains more information from the records, to, for instance, be able to display a hierarchy and to retrieve enough information at once to display all records for all keys. For this it is necessary that you can force an already set `XMLTYPE` in the result for a `SCAN`: you can do this with `SCANXMLFORCE=1`.
SEARCH	In Internet Server 3, submitted search queries are composed in the syntax of the Adlib *Expert search system*; in the URL entry field of your browser precede the search string with `SEARCH=`. Do use the hexadecimal forms of punctuation marks and reserved characters. Being able to structure search queries like in the Adlib search language, also means that you've largely got the same possibilities. When coding a CGI string through name/value-pairs, the search query is already implicitly put together as an Adlib search, so here you may not use the `SEARCH` command.
SEARCHMODE	Since wwwopac 4.5.3, a search with name/value-pairs allows you to enter more than one term in each `VAL`, separating the values with spaces or plus or minus characters (see further below for an explanation).

The purpose of something like this is to let visitors of your web application search (also) through a singular entry field, as they might do in search engines on the Internet. These search engines allow users to enter several terms into one search field, separated by spaces and/or plus or minus characters.
Before wwwopac 4.5.3 you would use `WQY` instead of `VAL`.
From wwwopac 4.7 on, this is different again. The `WQY` variable has been removed, and replaced by the (optional) variable `SEARCHMODE`. This variable can be used as follows:

Syntax	**Description**
`SEARCHMODE=weband`	All specified values must appear in one or (spread out over) more occurrences of the field of a record. You cannot use plus-characters, but minus-characters are allowed.
`SEARCHMODE=webor`	At least one of the specified values has to appear in the field of a record. You can use plus or minus characters to refine your search (see the overview underneath).
`SEARCHMODE=`	If you specify an empty `SEARCHMODE`, the entire entry is regarded as one long string, and that string will be used in the search. (This is the 'classic' search method with `VAL` terms for wwwopac.)

Chars in entry field	**Description**
+	The values before and after + have to appear in the search result simultaneously. Those values may be from one or more occurrences.
-	The value after - must not appear in the search result.
`<space>`	Use a space to separate individual values.
‘ ’	Use single or double quotation marks around a value to find an exact text containing spaces, such as when a term consists of several words, e.g.: `“OAI repository”`.

You can use `WEBAND` and `WEBOR`, just as in ‘classic’ searching, both for truncated and non-truncated searches.

If you want to set a different search mode for one or some entry fields, use the `MODEi` variable for this.

On your web site you may offer visitors the option of selecting a search method with a web search engine syntax, through `SEARCHMODE` and some button or selection field. So `SEARCHMODE` applies to all name/value-pairs.

(When searching multiple values in one field, you have to apply the escape coding of +, -, spaces and quotation marks in the CGI-string in the browser entry field *Address*.)

SEQ	The standard order of records is `ascending`. In a search with name/value pairs, select the alternative via `SEQi=descending`. The index number here, is separate from `FLD` and `VAL` index numbers. So, `SEQ1` is not associated with `FLD1`, for example. You do group `SEQ`, `SRT` and `TYP` with this index number. With `SRT` you could indicate that the search result must be sorted on author, for instance, and then set the order of the listing with `SEQ`. `SEQ` cannot be used without specified `SRT`.

SRT	In a search with name/value pairs, sort the search results on a field name or tag via `SRTi`. Note that sorting considerably slows down a search query!
	
	The index number here, is separate from `FLD` and `VAL` index numbers. So, `SRT1` is not associated with `FLD1`, for example. You do group `SEQ`, `SRT` and `TYP` with this index number, although `SRT` can also be used on its own. With `SRT` you could indicate that the search result must be sorted on author, for instance, and then set the order of the listing with `SEQ`. For example:
	
	`…/wwwopac.exe?database=ChoiceFullCatalogue&`
	`FLD1=ti&VAL1=ko&TRC1=on&SRT0=author.name&`
	`SEQ0=descending`
	
	In the `<diagnostic>` of the XML search result, you'll find this search query indicated as follows:
	
	`<search>'ti' = "ko*"</search>`
	`<sort>sort 'author.name' descending</sort>`
	
	If you want to sort on more than one value, the first `SRT`-variable could get index number '1', the second the number '2', etc.

STARTFROM	In the web application, usually only a part of an index is displayed. To retrieve the next or previous part of the list, you should use `STARTFROM`. Specify the numerical value from which the search should start, in which '1' is the first key after searching on `SCANVAL`. You can search forwards (`STARTFROM` = positive value) or backwards (`STARTFROM` = negative value). Use the `LIMIT`-value to calculate the `STARTFROM`-value for the new page.

TRC	The default setting for right truncation is off. Switch it on for name/value pair i via `TRCi=on`. You can also truncate a value by placing an * behind the (first part of a) word.
	The `TRCi=all` command, in combination with a search for an empty `VALi` retrieves all records from the database. `TRCi=all` only works for fields with term indexes.So you should use e.g. the following search query:
	`FLD1=author&VAL1=&TRC1=all`
	However, the following search query will not retrieve any records at all:
	`FLD1=author&VAL1=&TRC1=on`
	So `TRCi=all` or `TRCi=on` determines whether users are allowed to access records in the database through an empty search key.

TYP	To be used next to `SRT`, to indicate the type of sorting. Sort the search result alphabetically (default), numerically, or on date (for a date field) via `TYPi=t`, `TYPi=n`, or `TYPi=d` respectively. The index number must match the one from the intended `SRT`. This is not associated with the index numbers for `FLD` and `VAL`.
VAL	With `VALi` you specify the value you want to search on, in the field `FLDi`. See the explanation for `FLD` and `TRC` for more information.

Always enter reserved words in the search query in capitals, unless indicated otherwise. Apart from this, only the field names and tags are case-sensitive; copy these exactly as they appear in the database. For the other parameters and values the use of upper or lower case makes no difference.	
`<%HH>`	Reserved characters and punctuation marks (all non-alphanumerical characters) in names and values entered into a CGI-string must be replaced (escaped) by a percentage sign with two hexadecimal numbers; see the following examples:
`<space>`	Spaces in form and/or field names and values have to be replaced by a +. But %20 instead of + also works. Preferably, do not use spaces literally.
\n	In values with multiple lines of text, replace \n (CR LF: hard return) by %0D%0A.
&	The &-character (ampersand) in names and values has to be replaced by %26
,	Commas in names and values have to be replaced by %2C.
%	A %-character in text is encoded as %25. So the Adlib tag for record number (%0) must be encoded as %250: `FLDi=%250`. But you can also use the field name: `priref`.
+	A plus sign (e.g. for use in `SEARCHMODE`-input) in a CGI-string must be encoded as %2B.
-	A minus sign (e.g. for use in `SEARCHMODE`-input) in a CGI-string must be encoded as %2D.
‘	A single quote is encoded as %27.
“	A double quote is encoded as %22.

*** SCAN-results**

The pre-defined tags and reserved variables in the temporary file created by the scan function are (for each field):

&B[1]:	the total number of records in the selection (after any applied limit). In XML-output this value is placed between the tags `<hits_on_display>` and `</hits_on_display>`.
&B[2]:	would normally give the total number of hits, but because wwwopac stops searching when it reaches the provided limit, this variable will always be equal to the limit. In the XML-output this value is placed between the tags `<hits>` and `</hits>`.
&B[3]:	contains the value from `STARTFROM`. In XML-output this value is placed between the tags `<first_item>` and `</first_item>`.
&B[4]:	answers the question "are there any values in the list preceding the first value in this selection?" with 0 (no) or 1 (yes). In XML-output this value is placed between the tags `<backward>` and `</backward>`.
&B[5]:	answers the question "are there any values in the list following the last value in this selection?" with 0 (no) or 1 (yes). In XML-output this value is placed between the tags `<forward>` and `</forward>`.
&I:	(&capital i) `&I` gets a sequential number for every record retrieved in the selection (after the limit). For the first retrieved record after find with `SCAN`, `&I=1` is next `&I=2`, etc. When retrieving a new selection (via e.g. the buttons *Next* or *Previous*), `&I` is assigned a sequential number for the new records the same way: so for the first retrieved record from the new selection, `&I` will become 1 again. In the XML-output of wwwopac 6.2 and higher this sequential number no longer appears.
<TE>:	`TE` contains a term or name that was retrieved during a `SCAN`.

<NT>:	narrower term, may contain multiple occurrences, that can be approached through `NT[i]`, in which `i` is the number of the occurrence.
<BT>:	broader term, may contain multiple occurrences, that can be approached through `BT[i]`, in which `i` is the number of the occurrence.
<US>:	preferred term, may contain multiple occurrences, that can be approached through `US[i]`, in which `i` is the number of the occurrence.
<UF>:	non-preferred term, may contain multiple occurrences, that can be approached through `UF[i]`, in which `i` is the number of the occurrence.
<RT>:	related term, may contain multiple occurrences, that can be approached through `RT[i]`, in which `i` is the number of the occurrence.
<priref>:	contains the priref, the number of the term or name record in the linked database, in which the retrieved term can be found.

2.1.1 Three different syntaxes

It does not matter for the coding of the complete URL whether the search query is entered into the *Address* field of a web browser, or if the query is composed out of user input, or if it is hard-coded in HTML. But you can choose from three different syntaxes:

1. As shown above, you can work with variable-pairs `FLDi=name&VALi=value&…`
   When this syntax is used (and the second one below), Boolean operators are executed front to back; you cannot use brackets to indicate a different order.
2. You can also assign values directly to tags (not to field names): `tag=value&…` Tags are case sensitive. Syntax 1 and 2 can be mixed.
3. If you prefer to build up search queries the way they are constructed in the Adlib *Expert search system*, then that is also possible: precede the search logic with `SEARCH=`. E.g.:
   `…/wwwopac.exe?DATABASE=mydb&SEARCH=all`
   or e.g.
   `…/wwwopac.exe?DATABASE=mydb&SEARCH=(author.name%20%3D%20jones*)`

You can also use brackets, open pointer files and nest Boolean operators and have them executed in the desired order. Do replace punctuation marks and reserved characters with their hexadecimal values preceded by a %-character.
So with the `SEARCH`-statement you can do anything you can also do in an Adlib-application, such as:

- creating combined selections;
- selections with brackets;
- selections with the `$today`-variable;
- print (on the server);
- open pointer files.

Note again that some special search language syntax is accepted by *wwwopac.exe* but not by *dispatcher.aspx*. See the full text at the beginning of this chapter.

2.1.2 SEARCH and FLD/VAL pairs in one search query

The three possible syntaxes of a search query can be mixed up. So you can use `FLD/VAL` pairs and a `SEARCH=` in one and the same query, separated normally by & characters.

To combine both parts, the `SEARCHBOOL` operator can be used. You can assign the values `OR` or `AND NOT` to this operator. `AND` is the default if both syntaxes appear, so then you don't have to provide a `SEARCHBOOL`.

When interpreting a query, Internet Server cannot distinguish the order in which the `SEARCH` part and `FLD/VAL` pairs are placed, if they have to be combined with `AND NOT`. Therefore, by default the assumption is made that the `SEARCH` part is located left in the equation. So that makes an `FLD/VAL` query `AND NOT` a `SEARCH` query impossible.

2.1.3 Pointer files

You can also open pointer files through wwwopac. Pointer files are useful to present the visitor of your web site with a selection of records, in the form of a gallery, e.g. you can use a pointer file to regularly save a new selection (from within your own Adlib-application). When you save the file under the same pointer file number, you do not even have to modify the hard-coded search query in the HTML or ASPX-page to show the new gallery.

For information about how to create pointer files, see the Adlib User Guide.

In the search query for wwwopac simply use:

```
SEARCH=pointer%20#
```

For # enter the number of the desired pointer file (entering its name is not possible). %20 replaces the space in CGI-strings. Wwwopac searches for the pointer file in a subfolder reserved for pointer files (with the name of the database) in the *data* folder of your application. You can save the pointer files for wwwopac somewhere else, but then you have to tell wwwopac where the pointer file can be found. Do this with the variable `ADLIB_PTR`. Assign to this variable the path name to the pointer files, in the search query or in the web configuration file.

When, upon opening a pointer file, you encounter error code 135 then the pointer file cannot be opened. This may mean that the pointer file is currently in use, or that it does not exist.

2.1.4 isutf8=1

If you insert the assignment `isutf8=1` as an argument in the CGI-string, then all text in that string will be encoded as UTF-8; without that assignment all text will be interpreted as WinLatin1. For *wwwopac.exe/adlibweb.dll* it is necessary that the CGI-string is presented in one particular encoding (UTF-8 or ASCII). But MS Internet Explorer is not quite consistent in encoding CGI-strings for making a GET or POST request, when characters from a different character set (e.g. Japanese) are inserted: the whole string will be encoded in UTF8, except for characters with accent-marks from the current character set – in western Europe this is WinLatin1 (ISO-8859-1) for example. (And technically these accentuated characters unfortunately cannot be distinguished from UTF-8 encoded characters.)

If you think that users will only enter western characters, then you can leave `isutf8=1` out, but reality can of course turn out differently and give problems.

If you want all possible characters to be accepted in the search string, then use `isutf8=1` to interpret the CGI string as UTF-8. In the Internet Server 3 model web application, this setting is inserted by default, and you cannot change that.

To use this conversion in an older Internet Server application, you'll have to adjust it as follows:

In the HTML page after the `<html>` tag, insert the following code:

```
<meta http-equiv="Content-Type" content="text/html;
charset=us-ascii">
```

and in the search form put:

```
<input type="hidden" name="isutf8" value="1">
```

(If `us-ascii` results in problems, you might replace it with `ISO-8859-1`.)

For web applications written in C#, you will *have* to use `isutf8=1`, because C# functions always apply UTF-8 encoding.

2.2 Retrieving images

2.2.1 Creating thumbnails dynamically

You may want to show images of your collection items as thumbnails in the brief display. For this, it is not necessary to keep separate miniature images in your database. The wwwopac can dynamically decrease the size of the normal images in your database.

For this, you can instruct wwwopac by adding a specific command to the src attribute of an HTML image-tag. The Internet Server web application derives the name of the image which needs to be retrieved from the current search result. (In static HTML pages the src attribute is usually followed by the URL to an image.)
In the same command, you can also specify the dimensions of the thumbnail, and the output type of the miniature (this does not have to be the same type as the original image). The benefit is that the decreased image is sent to the client, and because it takes up less memory, a smaller amount of data is transferred via the internet and web pages are loaded faster.
An HTML command for retrieving an image with a known name and location, may look like this:

```
<Img src="http://localhost/mywebapp/wwwopac.exe?
thumbnail=../../webimages/test.gif&outputtype=
image/jpeg&xsize=10&ysize=20&dontkeepaspectratio=1">
```

The parameters for use in a thumbnail query are:

`XSIZE`	`XSIZE` specifies the horizontal size of the resulting image. `YSIZE` is determined automatically, in which the x/y aspect ratio is maintained.
`YSIZE`	`YSIZE` specifies the vertical size of the resulting image. `XSIZE` is determined

	automatically, in which the x/y aspect ratio is maintained.
`OUTPUTTYPE`	Specify the mime type of the resulting image. This does not have to be the same as the original. Valid types include: • image/jpeg (factory setting) • image/gif • image/bmp
`DONTKEEPASPECTRATIO`	When you use this parameter and set it to 1, the original x/y ratio is not maintained. Only use this parameter when you specify both `XSIZE` and `YSIZE`. The resulting thumbnail image will always become `YSIZE` high and `XSIZE` wide, probably stretched in one of both dimensions. If you only specify one dimension, the original size will be used for the other dimension.

Maximum size for thumbnails

From wwwopac 6.1.0 there is the possibility to specify both the maximum width and the maximum height for these thumbnail images and also maintain the original aspect ratio of the image. Simply provide both an `XSIZE` and `YSIZE` value for the thumbnail that you want to retrieve, and do not specify `DONTKEEPASPECTRATIO`, or set `DONTKEEPASPECTRATIO` to zero. Then the aspect ratio will remain intact and the thumbnail image will always be limited in size by either the provided maximum width or the provided maximum height, or by both if `YSIZE`/`XSIZE` matches the height of the original image divided by the original width.

Setting a background colour - If you set a maximum size for image thumbnails, and the original image has a different aspect ratio, then next to, or above and beneath the thumbnail, empty space will become visible. This is the background for the thumbnail, and it is black by default.

When dynamically generating a thumbnail image, you can give it a different background colour though. You do this via the

`THUMBNAILBACKCOLOR` parameter. Behind it you provide a colour as a hexadecimal colour code (like in HTML), in the RRGGBB format (R=red, G=green, B=blue). In this colour model, each colour component can have a decimal value of 0 up to and including 255 (so there are 256 shades of red, 256 shades of green, etc.) The combination of three shades in these colours gives the resulting colour. The decimal numbers have to be converted to hexadecimal, per number, and concatenated. A decimal 00 stays 00 in hex, and 255 becomes FF. Examples: FF0000 is bright red, 00FF00 is bright green, 888888 is grey. (If necessary, use the Windows Calculator to convert decimal values to hexadecimal.)

So, in an Internet Server request this may become, e.g.:
`…&thumbnailbackcolor=222222…`

In the Internet Server 3 web application, to be generated thumbnails are specified in the XSLT stylesheets for the different brief and detailed displays of museum objects, for example:

```
<xsl:template match="Reproduction/reproduction.identifier_URL">

 <xsl:if test="position() = 1">

  <!--  Only show the first image   -->

  <img class="detailImage" border="0" src="{$dataUrl}?thumbnail=
  {@weburl}&outputtype=image/jpeg&xsize=80&dontkeepaspectratio=0&
  fullimage=1" alt="{../object_name}" />

 </xsl:if>

</xsl:template>
```

You may adjust the `thumbnail` parameters to your desire, for instance to generate bigger or smaller thumbnails.

2.2.2 Retrieving images in high resolution

You can send (reduced) images of items in your collection from the server to the client via the `thumbnail` command. But with this command, larger images are sometimes retrieved in a non-optimal resolution. That's why extra parameters are available for retrieving larger images in a good resolution (read: quality). With these parameters you may clip an image, if needed, and then reduce or enlarge it, before sending the result to the client. So the parameters are used with `thumbnail=` and are the following:

`FULLIMAGE`	`FULLIMAGE=1` uses the full image for clipping and reducing or enlarging, and gives the best resolution.

	`FULLIMAGE=0` (default) uses either the full image or thumbnails that have been created automatically by Windows for display in Explorer. Use this setting when you wish to generate thumbnail images.
`CROPX`, `CROPY`	These are the coordinates, measured from the left upper corner of the image (x is horizontal), that indicate the left upper corner of the section to be clipped. The number to provide as a coordinate, may be in pixels or a percentage of the concerning dimension of the full image.
`CROPWIDTH`, `CROPHEIGHT`	These are the dimensions that the image to be clipped must get, before reducing or enlarging via `XSIZE` and `YSIZE`. The number you provide as the width or height may be in pixels or a percentage of the relevant dimension of the full image.
`CROPABSOLUTE`	With `CROPABSOLUTE=1` you indicate that the numbers you provided for the other `CROP` parameters are absolute (that means: in pixels). `CROPABSOLUTE=0` (default) indicates that the concerning numbers are percentages.
`XSIZE`, `YSIZE`	The horizontal and vertical dimensions (in pixels) of the resulting image. If you provide only one of both parameters, the other will be calculated automatically for a constant aspect ratio.

An HTML command for retrieving an image of which the name and location are known, may look as follows:

```
<Img src=”http://localhost/mywebapp/wwwopac.exe?
thumbnail=../../webimages/myimage.gif&outputtype=
image/jpeg&xsize=200&dontkeepaspectratio=0&fullimage=1&crop
x=10&cropy=10&cropwidth=380&cropheight=280&cropabsolute=1”>
```

Suppose *myimage.gif* in this example is 400x300 pixels in size. It is a photo of a painting that also shows the frame. What this query does is “clipping” a border of 10 pixels wide from the image, until this has the dimensions 380x280 (and the frame is no longer visible); then the image is reduced to 200x147 pixels (the y-coordinate will be calculated automatically, because the aspect ratio must remain constant in this example). This result will then be sent to the client. Of course the original image file *myimage.gif* will remain unchanged.

2.2.3 Default thumbnail image

If, for display of a search result, an image to be retrieved cannot be found, this error is normally indicated with a standard Windows icon displayed where the image should've been placed. But from wwwopac 5.0.3 you can specify a default thumbnail image in the *adlibweb.xml* web configuration file, to display that instead of a missing image. This way you can make a suitable image yourself, for instance one that doesn't make the impression of an error so much, or one that fits the design of your site better.
In *adlibweb.xml* you specify a default image through the setting: `DEFAULTTHUMBNAIL=<image.ext>`, so for example:
`<DEFAULTTHUMBNAIL>museumlogo.jpg</DEFAULTTHUMBNAIL>`.

2.2.4 Watermark all displayed images automatically

If images in the (detailed) presentation of records are displayed so large that it may become tempting for visitors of your website to copy the images and possibly reproduce them without taking the copyrights into account, it may be desirable to apply a watermark or copyright notice to all those images when they are displayed. Since you probably do not want to change the images themselves, from wwwopac 5.0.3 there is the possibility to have your images automatically watermarked on display in the web browser, by overlaying each image from a record with an image of a watermark or copyright notice, made more or less transparent and with a custom size and position relative to the underlying image from the record. The settings described below offer many possibilities. (The image of the watermark or the copyright notice is called an "overlay" here.)
Note that these settings can only be made in *adlibweb.xml* and not in CGI strings (to prevent watermarking from being switched off through the command-line in the web browser).

IMAGEOVERLAYFILE=<file name>

Provide (the path and) the file name of the image that you want to use as the overlay. If you want to use a relative path, take the relative path with respect to your *\data* folder.
Example:
`<imageoverlayfile>..\imgs\copyright.jpg</imageoverlayfile>`

IMAGEOVERLAYBLEND=<1-100>

Enter de extent of opacity of the overlay: with a percentage of 100 the overlay is completely opaque, while a low percentage makes the overlay almost completely transparent.

IMAGEOVERLAYPOSITION=<0-4>

With the five possible values you determine the position of the overlay with respect to the underlying image (because the overlay might be smaller): 0= top left, 1 = top right, 2 = bottom right, 3 = bottom left, and 4 = centered.

IMAGEOVERLAYPERCENTAGE=0 or <1-100>

With this option you determine the relative size of the overlay with respect to the underlying image. Choose 0 (the default value) if the overlay size must never be adjusted to the size of the underlying image. Choose a value (percentage) between 1 and 100 to have the overlay resized with respect to the size of the image to be displayed underneath. With 100 the overlay will get the same size as the image; with every lower value the overlay will be made proportionally smaller than that image.

IMAGEOVERLAYMINPIXELSIZE=<number of pixels>

If you only want to use the overlay for large images, not for small images or thumbnails, then through this setting you can provide the minimum size (in the total number of pixels) that an underlying image to be displayed must have, before an overlay will be applied. The original size of the image to be retrieved is not important, only the size in which that image will be displayed in the detailed or brief presentation. Provide the total number of pixels through an integer number. If you enter 0 (default value), the overlay will always be applied. Suppose you only want to use your overlay for images that are larger than the thumbnails of them in the brief presentations. If your thumbnails are 10x10 pixels, you'll have to provide a value of 101 for the currently described setting. To displayed images of for instance 5x75 or 250x200 pixels, the overlay will then be applied, but not to images of for example 8x9 or 10x10 pixels.

2.3 Imagemetadata

With the command `imagemetadata=<file_name_of_picture>` you can request metadata of images. This means information such as the date on which the picture was stored and the file size, amongst others. For the file name it's best to enter a relative pathname (relative with respect to the main folder of the Internet Server). The path needn't be enclosed by quotes, and in the path you can use / as well as \, for example:

```
…wwwopac.exe?imagemetadata=images/test.gif
```

or for multiple images at the same time:

```
…wwwopac.exe?imagemetadata=images/*.gif
```

The result for one found image will look something like the following:

```
<?xml version="1.0" encoding="UTF-8" ?>
<imageMetaDataSet>
  <imageMetaData>
    <FILE>test.gif</FILE>
    <SIZE>2150</SIZE>
    <CREATIONDATE>27/03/2005</CREATIONDATE>
    <CREATIONTIME>16:11:39</CREATIONTIME>
    <WRITEDATE>24/01/2006</WRITEDATE>
    <WRITETIME>11:45:38</WRITETIME>
    <ACCESSDATE>07/06/2006</ACCESSDATE>
    <ACCESSTIME>15:44:10</ACCESSTIME>
    <PATH>D:\demo.adlibsoft.com\IS3\images\test.gif</PATH>
  </imageMetaData>
</imageMetaDataSet>
```

For more information about image metadata, see the Adlib Designer Help.

3 OAI (Open Archives Initiative)

The OAI-protocol, based on HTTP and XML, is a metadata harvesting protocol, intended to make the normally invisible contents of internet databases accessible and searchable by search engines through a submitted term.

The Adlib *wwwopac.exe* and *aoiserver.exe* support OAI-protocol 2.0. Through an OAI protocol request to a repository, a server on which e.g. wwwopac is configured for the OAI-protocol, (meta) data can be extracted from the database (Adlib databases as well), and be made available on the internet where it can be indexed by service providers such as search engines. A full description of the OAI-protocol can be found on http://www.openarchives.org

---

The difference between data and metadata in an Adlib database is a matter of agreement. Normally, we regard the information in the database as data, but the description of an object or book, can just as easily be seen as metadata because the contents of the book itself are not contained in the database. Adlib databases store information about other data or objects: this is metadata.

The metadata supplied by an OAI search query to an Adlib database, can comply to a number of specified standards. The Dublin Core metadata standard is the standard that is provided with the Adlib implementation of OAI. See http://www.openarchives.org/OAI/openarchivesprotocol.htm#dublincore for the XML.-schema, or http://dublincore.org/document/1999/07/02/dces/ for a more reader-friendly specification of Dublin Core.
The standard consists of 15 elements (comparable to Adlib fields) in which the metadata is passed on. Since Adlib databases contain many more fields than Dublin Core has elements, a limited quantity of information is selected from a retrieved Adlib record. But you can define more than 15 elements. Dublin Core is really a narrow base that can be supported by anyone; this makes it easier to exchange information between differently structured data.

In principle, wwwopac or oaiserver returns all fields from an Adlib record (in Adlib XML format). But specially for OAI, wwwopac/oaiserver uses an XSLT stylesheet (with an Adlib field to Dublin Core element mapping) set in the web configuration file to transform that search result to the proper metadata format (also in XML), before sending it to the harvester. With this, the metadata search result is transformed to a so-called OAI record (one OAI-record per Adlib-record). Such a record primarily consists of a *header* and the *metadata*. The header consists of a unique identifier for the retrieved record, and of a date stamp that indicates when the record was last modified. The metadata is of course the retrieved data after transformation.

So the output format of an OAI search result is determined by an XSLT stylesheet on the server. Our standard Dublin Core stylesheet for this purpose is: *oai_dc.xsl*. Possible other output formats can be based on this stylesheet.

3.1 Settings for the adlibweb.xml file

The use of OAI must be configured in *adlibweb.xml* – you can no longer use the www file for this configuration. By default, there is no OAI configuration in *adlibweb.xm*l for *wwwopac.exe* with Internet Server 3; in *adlibweb.xml* for *oaiserver.exe* however, you'll only find an OAI configuration. So for the Internet Server web application you'll have to add this configuration. Adjust the example below (of a complete OAI configuration) to your own situation and add it to *adlibweb.xml*.

```
<OAIConfiguration>
<databasepath>..\..\data</databasepath>
<database>collect</database>
<OAI_REPOSITORY_NAME>My Museum</OAI_REPOSITORY_NAME>
<OAI_ADMIN_EMAIL>oai@mymuseum.uk</OAI_ADMIN_EMAIL>
<OAI_ADMIN_EMAIL>admin@mymuseum.uk</OAI_ADMIN_EMAIL>
<OAI_REPOSITORY_IDENTIFIER>myMuseum</OAI_REPOSITORY_IDENTIFIER>
<OAI_DELIMITER>:</OAI_DELIMITER>
<OAI_MAX_RECORDS>5</OAI_MAX_RECORDS>
<OAI_MAX_IDENTIFIERS>10</OAI_MAX_IDENTIFIERS>
<OAI_DIR>..\oai</OAI_DIR>
<OAI_SETS_DIR>..\oai\oai-sets</OAI_SETS_DIR>
<DM>dm</DM>
<OAI_ID_DESCR>id-description.xml</OAI_ID_DESCR>
<OAI_ID_DESCR>eprins.xml</OAI_ID_DESCR>
<metadataFormats>
  <metadataFormat>
   <metadataPrefix>oai_dc</metadataPrefix>
    <schema>http://www.openarchives.org/OAI/2.0/oai_dc.xsd</schema>
    <metadataNamespace>http://www.openarchives.org/OAI/2.0/oai_dc/
                                            </metadataNamespace>
  </metadataFormat>
  <metadataFormat>
    <metadataPrefix>dc_culture</metadataPrefix>
    <schema>http://urltoschema</schema>
    <metadataNamespace>URI:DC.Culture/XMLSchema/1.0
                                            </metadataNamespace>
  </metadataFormat>
```

```
</metadataFormats>
</OAIConfiguration>
```

Explanation:

Below, everything in between the XML tags, is a description of what you should fill in. New terminology will be explained later in this chapter.

`<database>``the Adlib database in which one can search`**`</database>`**

`<databasepath>``disk station:\my Adlib folder\data`**`</databasepath>`**

`<DM>``date tag`**`</DM>`**

> In a lot of databases, in a date-of-modification field the date on which a record was last edited, is stored. When you make your database available on the internet by means of the Open Archive Initiative, you can use the `DM` variable in the *adlibweb.xml* file to provide the tag of said field, so that search engines or other clients can only retrieve the records (or index them) that have been changed after a certain date, for instance after the previous visit to your OAI server. An OAI request with a date selection for a database in which no `DM` is specified, returns *all* records.

`<OAI_ADMIN_EMAIL>``administrator e-mail address`**`</OAI_ADMIN_EMAIL>`**

`<OAI_DELIMITER>``separator character`**`</OAI_DELIMITER>`**

> Choose a separator (for instance a semicolon) to separate the URI values behind the identifier variable in an OAI request. If you do not include an `<OAI_DELIMITER>`, wwwopac/oaiserver assumes you want to use a colon.

`<OAI_DIR>``path to a directory where the OAI resumption tokens and style sheets must be stored`**`</OAI_DIR>`**

> When you enter a relative path, this path must be relative to the path you set with `<databasepath>`.

<OAIFIELDS>`fields or tags`**</OAIFIELDS>**

> Normally, at first all data dictionary fields are retrieved in the XML result of an OAI search. Only after retrieval, a selection from these might be made with a stylesheet. Through the optional `<OAIFIELDS>` though, you can indicate just which fields should be retrieved; this may result in faster performance.
> Specify the desired tags or field names, each between `<field>` tags (also see paragraph 1.2.3).

<OAI_ID_DESCR>`XML file with id-description`**</OAI_ID_DESCR>**

> The optional `OAI_ID_DESCR` variables in the *adlibweb.xml* file refer to XML files of which the contents is shown by the Identify request as description entities (extra metadata about the repository itself, that may be important especially within a certain community). The concerning files will be pasted underneath the XML search result, in between *<description>* tags.
> Since these files won't be checked for accuracy during the inserting in the response, it's important to always check whether they are correct before they are inserted. An incorrect file will cause an Identify request to fail.
> The XML files must be located in the data directory, in the directory in which wwwopac/oaiserver resides, or in the OAI_DIR folder. The program first searches the data folder, then the OAI_DIR folder, and last the wwwopac/oaiserver folder.
> There may be zero or more `OAI_ID_DESCR`s.

<OAI_MAX_IDENTIFIERS>`maximum number of identifiers`**</OAI_MAX_IDENTIFIERS>**

<OAI_MAX_RECORDS>`maximum number of retrievable records`**</OAI_MAX_RECORDS>**

<OAI_REPOSITORY_NAME>`name`**</OAI_REPOSITORY_NAME>**

<OAI_REPOSITORY_IDENTIFIER>`repository id`**</OAI_REPOSITORY_IDENTIFIER>**

<OAI_SETS_DIR>`path to sets directory`**</OAI_SETS_DIR>**

> The `OAI_SETS_DIR` variable contains the directory in which Adlib pointer files are stored in subfolders with the relevant database name, which will be used as sets by the OAI protocol.
> When you enter a relative path, this path must be relative to the path you set with `<databasepath>`.

If you do not specify a sets directory, the default pointer files folder will be used instead.

<XMLTYPE>`STRUCTURED or UNSTRUCTURED`**</XMLTYPE>**

With the optional `<XMLTYPE>` you may specify the XML structure of the search result. By default that is `STRUCTURED`, but you may also specify the result to be `UNSTRUCTURED`. For e.g. Dublin Core this won't make much of a difference, but it does for hierarchical data structures. (See page 12 for more information about these two XML structures.) You should use a different stylesheet for each XMLTYPE.

In the example you can also see that within the first `<metadataFormat>` tags the standard stylesheet is specified, of which the `<metadataPrefix>` indicates the name of the file. In the second section a fictitious (custom made) stylesheet named `dc_culture` is declared. You can provide as many stylesheets as you want. In the OAI search query it is specified which of these stylesheets must be used.

The following variables have to be entered in the *adlibweb.xml* file for use by the OAI-protocol:

- `OAI_REPOSITORY_NAME`
- `OAI_ADMIN_EMAIL`
- `OAI_REPOSITORY_IDENTIFIER`
- `OAI_DELIMITER`
- `OAI_DIR`

The rest is optional.

3.1.1 AdlibXML

By default, the wwwopac/oaiserver yields XML in the *adlibXML* format. If you wish to produce OAI search results in this format, and not have them transformed to "metadata" by a stylesheet, then you'll have to remove the `<metadataFormat>` section(s) and instead specify the *adlibXML* format to be used explicitly, in your OAI configuration with the `<metadataFormats>`. To apply this option, include the code below literally in your OAI configuration:

```
<metadataFormat>

   <!--  the built-in type 'adlibXML'
   -->
   <metadataPrefix>adlibXML</metadataPrefix>
   <schema>http://www.adlibsoft.com/adlibXML.xsd</schema>
   <metadataNamespace>http://www.adlibsoft.com/adlibXML
   </metadataNamespace>
</metadataFormat>
```

You may specify the contents and structure of this XML format with `<OAIFIELDS>` and `<XMLTYPE>`.

Possible applications of this raw XML result are for synchronisation with XML databases, or for development purposes.

3.2 Use

■ function call

When the keyword `verb` is entered in the function call of *wwwopac.exe* or *oaiserver.exe*, the server will process the search query as an OAI-request.

In principle, the search result of an OAI-request is yielded as an XML-file, and so the client has to convert it to an HTML-page first in order to show the result as a web page. Nevertheless, current browsers can also present an XML-file in code, which is sufficient for testing purposes. The syntax of a search query and the ways you can enter them are the same as for a normal search query for the wwwopac, be it that behind the question mark in the CGI-string a number of special variables has to be entered due to the OAI protocol.

To make for instance an Identify-request , use the following syntax:

```
…wwwopac.exe?verb=Identify
```

The `Identify` value provides information about the relevant repository which has been made available with the OAI-protocol, such as the repository-name (e.g. the name of your collection), the BASE-URL to address search queries to, the protocol version that is supported, the e-mail address of the repository system administrator, and any additional information.

Note that in this and the following examples `wwwopac.exe` can be replaced by `oaiserver.exe`, depending on which server you use.

A standard OAI-protocol request has at least one name/value pair to specify the request by the client. The above Identify-request is an example of this, but instead of Identify, each of the standard OAI-protocol requests can be used. The number and the nature of extra name/value pairs depend on the arguments for the specific protocol request, e.g.:

```
…wwwopac.exe?verb=GetRecord&identifier=oai:MyMuseumORG:3&me
tadataPrefix=oai_dc
```

for a GetRecord-call of the record with identifier `oai:MyMuseumORG:3`. But in a URI (the value-part in an OAI name/value pair) special characters have to be encoded as an escape-string, a percentage character with a hexadecimal value. So the valid request is:

```
…wwwopac.exe?verb=GetRecord&identifier=oai%3AMyMuseumORG%3A
3&metadataPrefix=oai_dc
```

The identifier value is preceded by `oai`, the "schema" to be used. Next is the repository-identifier (case sensitive), and finally a local identifier that indicates the record number of an Adlib-record (the priref). These three parts are separated by the encoded delimiter (here %3A for a colon). The metadataPrefix `oai_dc` specifies the use of the Dublin Core output format, as specified in the *adlibweb.xml* file.

Also when you submit a `ListIdentifiers` or `ListRecords` request (see further in this chapter), then with the `metadataPrefix` parameter you specify (indirectly) which stylesheet should be used .
On the cd-rom of Adlib 5.# or higher you'll find the standard stylesheet for this purpose: *oai_dc.xsl*. Place it in your own *InternetServer\OAI* directory (that you specify with `<OAI_DIR>`). (Create this subfolder if it doesn't exist yet.) On the base of this stylesheet you can make you own stylesheet with another name, if you desire. Save it in the same directory; in here you also store the resumption tokens (see further on), and in a subfolder for OAI sets you store the pointer files that you want to make available as sets.

Reserved characters for use in URIs are:

Character	**URI-role**	**Escape-string**
/	path-component separator	%2F
?	query-component separator	%3F
#	fragment identifier	%23
=	name/value separator	%3D
&	argument separator in query component	%26

:	Host-port separator	`%3A`
;	authority namespace separator	`%3B`
	space	`%20`
%	escape indicator	`%25`
+	escaped space	`%2B`

If the use of these characters does *not* correspond with their predestined URI-role, then they have to be represented by their escape string.

3.3 Protocol requests

A client can submit various requests to a repository. Every request is the value of a `verb`, namely: `verb=<request>` and follows directly behind the question mark in a search query.

The most important requests are summed up underneath. Detailed information about their syntax and the responses they trigger can be found on
http://www.openarchives.org/OAI/openarchivesprotocol.html.
(Requests are case-sensitive.)

`GetRecord`

This verb is used to extract an individual metadata-record from a repository record. Mandatory arguments specify the identifier of the requested record, and the output format of the metadata (e.g. `oai_dc`).

`Identify`

Identify gathers information about the repository. No arguments.

`ListIdentifiers`

This verb is used to gather the local identifiers (prirefs) of all records that are available in the repository. Optional arguments enable selective gathering, for instance based on the date they were last modified, or as a part of a set.

`ListRecords`

This verb is used to gather all* the records available in the repository. Optional arguments enable selective gathering, for

instance based on the date they were last modified, or as a part of a set.
* The *adlibweb.xml* `OAI_MAX_RECORDS` settings specifies the maximum number of records which can be retrieved per request. Subsequent partial lists of the same search result may be retrieved via so-called resumptiontokens (see paragraph 3.5).

3.4 OAI Sets in Adlib

An OAI-search query can also access a set of records. Although the OAI set has more options, such sets are applied in Adlib by means of pointer files. One pointer file represents one set. In order to differentiate between pointer files for use in Adlib and those meant for OAI-requests, the *adlibweb.xml* file can be used to indicate where the sets are located. So the pointer files can be created with *adlwin.exe* and then copied to the correct directory.

In the directory that holds the OAI-sets, there should be the same sub-division (per used database) as in the data-directory. So when the pointer file comes from the *..\data\collect* directory and the OAI sets directory is called *\oai-sets*, then the pointer file must be placed in the *..\oai-sets\collect* directory.
If you do not make this subdivision, then wwwopac/oaiserver cannot find the pointer file (and the set will not be retrieved).

With the `ListSets` verb, a list of all pointer files can be retrieved, for example:

http://demo.adlibsoft.com/wwwopac/oaiserver.exe?verb=ListSets

The set structure of the repository will be returned. In the case of Adlib pointer files, this means that the first `setSpec` node contains the name of the database, to which the `setSpecs` listed below it (the numbers of the relevant pointer files) apply. In our example this yields pointer files 4 and 5 for the collect database.

The contents of a pointer file itself can be retrieved via other verbs, via `ListRecords` for instance. You'll have to provide the extra `set` argument in the request. You refer to the pointer file via the following format: `set=pointer_file_number`, for example:

http://demo.adlibsoft.com/wwwopac/oaiserver.exe?verb=ListRecords&set=5&metadataPrefix=oai_dc

3.5 ResumptionTokens

When the search result is a very long list, a limited portion of it will be created and accompanied by a resumptionToken (at the bottom of the XML result). Such a resumptionToken is necessary to open the next part of the list, as an argument in an OAI-request. (But the argument must first be encoded as an escape-string, should that be necessary.) Note that if in a request you use a resumptionToken, you must not provide a metadataPrefix, nor any other argument: the last used arguments are automatically applied again.

Suppose you've entered a first ListRecords request, which resulted in a list with for instance ten records, and a resumptionToken "97EODx6A3phYxIY":

```
../oaiserver.exe?verb=ListRecords&metadataPrefix=oai_dc
```

The next 10 records of the search result are now retrieved with:

```
../oaiserver.exe?verb=ListRecords&resumptionToken=97EODx6A3phYxIY
```

In the new list you'll find another resumptionToken if more records are available.

The resumptionTokens that are made by an OAI-response from wwwopac/oaiserver, are also stored together in one XML file on the server: *OAIResumptionTokens.xml.* Below, you can see an example:

```
<?xml version="1.0" ?>
<resumptionTokens>
 <resumptionToken>
  <name>z6LLHpGifhyW63KiUIOI</name>
  <fromDate />
  <untilDate />
  <metaDataPrefix>oai_dc</metaDataPrefix>
  <verb>ListIdentifiers</verb>
  <limit>25</limit>
  <hits>2066</hits>
  <startFrom>26</startFrom>
  <expires>1238657612</expires>
 </resumptionToken>

 <resumptionToken>
  <name>4NJeSGjY9XWXbL5RwK13</name>
  <fromDate />
  <untilDate />
  <metaDataPrefix>oai_dc</metaDataPrefix>
  <verb>ListIdentifiers</verb>
  <limit>25</limit>
  <hits>2066</hits>
  <startFrom>26</startFrom>
  <expires>1238658375</expires>
 </resumptionToken>

 <resumptionToken>
  <name>49DVgIjnJeKbzYWeJwND</name>
  <fromDate />
  <untilDate />
  <metaDataPrefix>oai_dc</metaDataPrefix>
  <verb>ListIdentifiers</verb>
  <limit>25</limit>
  <hits>2066</hits>
  <startFrom>26</startFrom>
  <expires>1238658417</expires>
 </resumptionToken>
</resumptionTokens>
```

With this information, the subsequent search query can be executed correctly. A token expires after 24 hours; this period is not configurable. However, there is a work-around to establish a longer period incidentally: date and time of expiry are saved per resumption-Token as the number of seconds from some historical date up to the expiry date and time. If you know up front that you want to keep certain resumptionTokens longer than 24 hours, then adjust the desired `<expires>` values in the XML file, by adding enough seconds to them (86400 per day of delay).

When a resumptionToken is made, and the file doesn't exist yet, it will be created. If it does exist, the resumptionToken is simply added. After the resumptionToken has been used, it'll be removed from the file. An empty *OAIResumptionTokens.xml* file will not be removed and needn't be.

3.6 Examples

For instance, go to http://re.cs.uct.ac.za/ to test OAI in practice. Enter a URL for an Adlib repository to an Adlib wwwopac/oaiserver that supports the OAI-protocol – explicitly name `oaiserver.exe` or `wwwopac.exe` in the URL – or choose one of the archives from the list. Enter parameters in the appropriate fields, and choose a verb to extract metadata.
See also http://oai.dlib.vt.edu/cgi-bin/Explorer/oai2.0/addarchive for an automatic test of a new repository. When an error occurs during testing, and it is unclear what caused it, the user can enter the URL to `oaiserver.exe` or `wwwopac.exe` him- or herself, followed by `?verb=Identify`. The output will then indicate what is wrong. Note

that full validation is only possible with *oaiserver.exe* (version 6.4.0.579 or higher), not with *wwwopac.exe*, because the error handling is somewhat different.

3.7 OAI repositories and search engines

Whether and how search engines like Yahoo and MSN search OAI repositories, differs per search engine, and usually the manner and extend of indexing is classified information. Any information on this topic must come from research. In 2000, an independent study was done (http://library.lanl.gov/cgi-bin/getfile?LA-UR-05-9158.pdf) into the search engine coverage of the OAI corpus. It turned out that Yahoo had indexed 65%, Google 44% and MSN 7% of that corpus. 21% had not been indexed by any of these three search engines. (Note that things may have changed significantly since 2000.)

To increase the chances of having your OAI repository indexed as much as possible, there are a couple of things you have to do:

1. If you own a web site on which users can view in detail records coming from the same database(s) as the ones you open up through OAI, you must first ask yourself if you want internet search engines to be able to index those detail records. The advantage could be a complete indexing of your database records; the disadvantage is a higher web server load. Whether you open up your probably dynamically generated web pages to "crawling" by "spiders", is often set by a *robots.txt* file on your web server. And most search engines respect a so-called robots exclusion so that your web site won't be searched/indexed.

2. For a useful indexing of your OAI repository it is not necessary per se that your web site is indexed too. It *is* relevant that all URIs to the full resources, e.g. URLs for requesting detailed views of your database records, will be indexed. So you have to make sure that the URI to a resource is part of the relevant OAI record. For this purpose use the Dublin Core `identifier` element. In the Adlib example stylesheets with the field mapping to Dublin Core, the `identifier` tag is still filled with a record number or object number, but you can easily put in there the URL to the page on your web site where the detailed display of the full record can be viewed.
   To this end, replace the *priref* template in *oai_dc.xsl* by something like the following:

```
<xsl:template match="priref">
  <xsl:element name="dc:identifier">
   <xsl:text>http://our_website/dispatcher.aspx?action=search
```

```
&amp;database=ChoiceFullCatalogue&amp;search=priref=</xsl:text>
   <xsl:value-of select="."/>
  </xsl:element>
</xsl:template>
```

All this does is paste a URL in front of the record number. A comparable URL can be used in Adlib Internet Server 3 to retrieve detail records. Note that in a stylesheet, ampersand characters appearing in the URL must be escaped, meaning: replaced by `&amp;`, as can be seen in the example.
This way you achieve two goals at once: your web site does not need to be indexed because OAI records will already be indexed, and since all URLs to detailed views will be indexed too, users of internet search engines will still have access to the best presentation of your database records.

3. Submit your OAI repository to OAI registers like that of open-archives: http://www.openarchives.org/data/registerasprovider.html . The base URL that you need to submit here, is your repository URL, in principle up to and including *oaiserver.exe* or *wwwopac.exe* (although when using an HTTP handler, the executable name possibly no longer occurs in the URL).
4. On: http://www.oaister.org/dataproviders.html you may then submit your repository to OAIster, which starts harvesting your OAI data immediately. OAIster is a search engine created especially for OAI repositories, and with it you can even search individual Dublin Core elements for data: so, this search engine implements OAI-PMH (Open Archives Initiative Protocol for Metadata Harvesting) itself, contrary to the "normal" search engines which harvest indirectly. Yahoo for example, does this via OAIster. (As of May 2008, Google has unfortunately ended support of OAI-PMH feeds as sitemap.)

3.8 Excluding certain records from OAI search results

With the Adlib OAI server you open up a database for OAI searches from the internet. However, it may be that not all records in that database should be publicly accessible. Luckily it's possible to exclude specific records from OAI search results.
Adlib *oaiserver.exe* runs under a certain account in IIS (Internet Information Services) – the name of this account is `ISUR_<server name>` by default, to indicate the anonymous internet user – and this account name can be entered as user name in a field to be added to your application for this purpose, which has to be set as *Authorisation user field* with *Exclude* as *Authorisation type* in the database. Every record in which subsequently said account name is stored, is excluded from results of any OAI search from then on.
See the *Use the authorisation functionality* paragraph in the *User authentication and access rights* topic under the *General topics* chapter in the Designer Help for an explanation about setting up this type of access restriction.

4 SRW/SRU

The successor of Z39.50 is SRW (Search/Retrieve Web service). SRW is an XML oriented protocol to perform search operations in a remote database with metadata and objects, and other information requests over the internet. It offers a standardized way to execute this functionality, contrary to the Adlib Internet Server web application which implements comparable functionality in an Adlib-specific (non-standardized) way (with input via the Adlib expert search language and the result as AdlibXML output).

With the SRW web technology it becomes easier for information suppliers to make their databases accessible through a standard web service. Because of the standardized character of this web service, third parties can independently build their own web applications using standard technologies, which are able to search said databases and present the data in a custom interface to visitors from the internet. Database owners only need to configure Adlib *wwwopac.exe* for the SRW protocol via the *adlibweb.xml* file; they don't need their own Adlib Internet Server web application for this purpose. Note that SRW makes the databases set up for this goal, available (read-only of course) to the entire internet, and that everyone can use the information from the databases as they please. Normally, public databases will be most suited. If you want to sell data or wish to publicize your data only within its context via your own web application (maybe also because of copyright on certain data), then SRW is not suitable.

SRU (SearchRetrieve by URL) is an companion service for SRW: one of at least two ways to transport the SRW protocol, in this case as name/value parameters (not in XML format) in a URL (in principle an http GET request) – the other way is via SOAP, but that is not available in wwwopac. (By the way, the term "SRW" is gradually being superseded by the phrase "SRU over SOAP", indicating the same thing.) SRW and SRU requests and results are comparable, the difference being the way search queries and results are encapsulated. Moreover, SRU is complementary to OAI, the Open Archives Initiative, which is supported by Adlib as well. Whilst the main purpose of OAI is to allow harvesting of bulk (meta)data, SRU is well suited to retrieve only the records you are looking for by means of specific search queries. The main similarity between both protocols is that the XML format in which the search result is retrieved, can be chosen by the user (the client), up to a certain extent.

CQL (Common Query Language) is the query language that is being used in SRW/SRU.

Adlib supports version 1.1 of the SRU search protocol and the query language CQL. *Wwwopac.exe* version 6.3 or higher is recommended.

4.1 How Adlib implements SRU

Adlib wwwopac implements SRU/CQL version 1.1 and conforms to the so-called SRU Base Profile (see http://www.loc.gov/standards/sru/base-profile.html) and the Adlib Base Profile (see appendix 2). The Adlib Base Profile can be supplied to third parties.
Base profiles are textual specifications, necessary in order to ensure interoperability between SRU servers and clients and are considered required for a client or server to be described as supporting the SRU protocol. The features described in such documents are considered to be the baseline implementation specifications, and if not implemented, the client or server will be unlikely to interoperate at all.
The Adlib Base Profile extends the SRU Base Profile.

SRW/U and CQL are broad protocols, with a lot of optional and server-dependent options. For instance, the CQL context set describes a lot of features which are just possibilities. Some of these possibilities have been implemented differently in wwwopac. Also, not all of the optional features have been implemented, for instance: SRW (SRU through SOAP) is not implemented, nor is the *scan* operation. All of this is documented in the Adlib Base Profile.

The Adlib SRU server is implemented inside the existing Adlib wwwopac executable, but is not configured for actual use after installation, since that would be undesirable for most customers. Just like for OAI (the Open Archive Initiative), you must configure this subserver yourself in the *adlibweb.xml* web configuration file before it can be used.

Configuration is split up into three parts:

- adding settings to *adlibweb.xml*;
- creating or adjusting an XSLT stylesheet for the required record formats in the search result;
- creating an XSD schema to define the resulting record formats.

Also, probably a customized profile document needs to be created. If you want Adlib to make this configuration and documentation for you, please contact our sales or helpdesk departments for an offer.

4.1.1 Creating your own SRU Profile

For SRU clients to be able to use your database, they need to know what indexes have been defined, what record formats are supported and how searches behave. This is defined in one or more so-called "context sets" and a "profile".
A context set is a list of indexes, relations, relation modifiers and their behaviour. The Adlib Base Profile (see appendix 2) already describes a/o the relations, relation modifiers and their behaviour, which cannot be changed by the configuration. So, the context set that you define yourself will only contain index descriptions.
A profile describes the context sets which are used, the record formats, and it specifies URIs.
It is encouraged to use pre-existing context sets if possible. For example, the Bath profile defines its own context set but also draws from the Dublin Core, CQL and Record Metadata context sets. Doing this promotes transparency and standardization. The same holds for usage of record schemas. The Bath profile for instance, uses the pre-existing Dublin Core and MarcXML formats.

First configure wwwopac (see the next paragraph) to support the indexes and the record format that you would like to export using SRU. Then describe these indexes and record formats in your own profile document. Explicitly state for each index whether it is a word, term, date (either ISO or European/US) or numeric index. This is very important since clients have to use different CQL relations for word and term indexes. Also state the *sortpath* that clients should supply for sorting on the index.

Your profile document extends upon the Adlib Base Profile, so refer to that document in your own profile.

Create an XSD XML schema for each new record format you define, or re-use existing formats (like Dublin Core or MarcXML). This way, the client knows how the retrieved records are structured. This is relevant for the further processing of the records (a transformation to HTML via a custom XSLT stylesheet, for instance).

Assign URIs* and (if applicable) short ids for:

- each record format;
- each context set, i.e. each set of indexes;
- your profile.

Add these URIs and ids to *adlibweb.xml* as well as your profile document. Examples of profile documents can be found on http://zthes.z3950.org/srw/ and http://zing.z3950.org/srw/bath/2.0/.

* A URI is an identifier that looks like a URL, but it does not need to point to a web address. For example, the URI for the SRU Base Profile is "*info:srw/profile/1/base-profile-v1.1*". By adding, e.g. the web address of your organization, a name and a version number, you can make sure this is a globally unique identifier, now and in the future.

Now you can publish the following information:

- the URL to your web server (up to and including '*wwwopac.exe*');
- your profile document;
- all XSDs that you defined.

This completes your SRU server description, so that anybody who wants to use it can do so.

It is possible to implement multiple profiles at once.

4.1.2 Configuring the Adlib SRU server (wwwopac)

The *adlibweb.xml* file must be extended with a separate section, named `<SRUConfiguration>`. See below for an example of such a section. Inside this XML element you can find the following subsections:

1. database settings;
2. implemented custom profiles;
3. record retrieval settings;
4. textual information;
5. record schemas;
6. context sets.

1. Database settings

Refer to the Adlib database which you want to make available on the internet via SRU, with:

- `<databasepath>`, `<database>`: the path to the data folder and the name of the Adlib database, for example:
  `<databasepath>D:\demo.adlibsoft.com\IS3\data</databasepath>`
  `<database>document</database>`
  Note that the SRU protocol does not allow for the possibility to indicate in the CGI string the database to be searched: only the one database set in the *adlibweb.xml* file can be searched. if you wish to make more than one database available, you’ll have to make a copy of the main folder (and thus of all its subfolders) of your Adlib Internet Server module – the data folder must not be

copied – for each database you want make available, and place that copy in a different folder. Then create a different virtual folder for every copy, and in the *adlibweb.xml* file of each copy, change the name of the database to be searched (and check whether the path to the database is still correct). Other settings in every copy of this web configuration file probably have to be adjusted as well, because the other database has different fields. So, if you would like three searchable databases for example, you'll need two copies and the original of the Adlib Internet Server folders, and in it different *adlibweb.xml* files. So the base URL of each wwwopac/database, for use by the SRU client, will be different.

- `<sruFields>` and `<XMLType>`: the fields (or tags) and xmltype (structured or unstructured) that are used to generate intermediate XML records. Using XSLT stylesheets, these records can subsequently be transformed to one of the record formats which your profile allows for.
  In the *adlibweb.xml* file you use these settings for example as follows:

```
<XMLType>structured</XMLType>
<sruFields><allDataDictionaryFields/></sruFields>
```

2. Implemented custom profiles

Enter the URIs of your own profiles (which must be based on the SRU Base Profile and the Adlib Base Profile), via `<sruProfile>`. All URIs to custom profiles indicated here, plus the URIs to the two default profiles as hardcoded in wwwopac, will be present in the SRU *explain* record. This way, clients know which profiles you implement. In *adlibweb.xml* you could for instance write:

```
<sruProfile>extraProfile1-URI</sruProfile>
```

In an *explain* record this comes back as:

```
<zrx:supports type="profile">info:srw/profile/1/base-profile-
v1.1</zrx:supports>
<zrx:supports type="profile">info:srw/profile/6/1.0</zrx:supports>
<zrx:supports type="profile">extraProfile1-URI</zrx:supports>
```

Note that although the default URIs still have "srw" in the identification, they refer to profiles which are now denoted as SRU profile.

3. Record retrieval settings

The <`limit>` and <`maxlimit>` define respectively the default maximum number of records returned when the client submits no `maximumRecords` argument, and the maximum number of records that can be returned regardless of the client's request. Both are optional. For example:

```
<limit>10</limit>
<maxlimit>100</maxlimit>
```

4. Textual information

This relates to information for SRU clients to be shown to the user. The `<sruTitle>` is mandatory and the other settings are optional. If `<sruLangUsage>` is supplied, then `<sruLangUsageCodes>` must contain a space-separated list of RFC-1766 country codes (see http://www.faqs.org/rfcs/rfc1766.html), e.g. 'en nl'. The meaning of all of these fields is explained in the ZeeRex documentation on: http://explain.z3950.org/.
An example of the relevant *adlibweb.xml* section:

```
<sruTitle>Adlib document database</sruTitle>
<!--  optional   -->
<sruContact>E-mail: info@adlibsoft.com Address: Postbox 1436
Maarssen the Netherlands</sruContact>
<sruDescription>Our SRU database</sruDescription>
<sruExtent>Extent</sruExtent>
<sruDataAuthor>Adlib</sruDataAuthor>
<sruHistory>history</sruHistory>
<sruLangUsage>Database contains records in Dutch and
English</sruLangUsage>
<sruLangUsageCodes>nl en</sruLangUsageCodes>
<sruRestrictions>There are no restrictions on the use of this
data</sruRestrictions>
```

5. Record schemas

At least one schema needs to be defined, and one of those must be the default. The following information is required:

- `id`: this is the short identifier to denote the schema in an SRU request; it is case-insensitive. Must contain no . < > & " ' characters.
- `description`: human readable description of the set.
- `schemaXSD`: full URL path to the XSD of the record schema.
- `schemaURI`: URI of the record schema. Must contain no . < > & " ' characters.

Optional is:

- `stylesheet`: the relative path to the location of a stylesheet which transforms the AdlibXML record to the desired record format. The location is relative to the location of *adlibweb.xml*. If no stylesheet is supplied, the AdlibXML record will be published. This then still needs an XSD describing the specific format of these records!

An example of record schema settings in *adlibweb.xml*:

```
<sruSchemas>

 <sruSchema default="true">
  <id>dc</id>
  <description>Dublin Core</description>
  <schemaXSD>http://www.loc.gov/standards/sru/dc-schema.xsd
   </schemaXSD>
  <schemaURI>info:srw/schema/1/dc-v1.1</schemaURI>
  <stylesheet>..\sru\dc.xsl</stylesheet>
 </sruSchema>

</sruSchemas>
```

6. Context sets

At least one context set needs to be defined, via `<sruSet>`, and one of those must be the default. Each context set needs to contain at least one index. One or none of the indexes can be the default; if no default index is specified, *cql.anywhere* will be the default.

For each context set, the following data is required:

- `id`: the short identifier for this set (associated with a record schema), is case insensitive. Must contain no . < > & " ' characters.
  `'adlib'` (without the quotes) is a special context set. In this case no `contextURI` or indexes may be specified. If this special id is specified, then just all indexes and linked fields will be added to the built-in `'adlib'` context set.
- `contextURI`: URI of the context set. Must contain no . < > & " ' characters.
- a list of indexes which you wish to make available. For each index, the following data is required:
  - `field`: Adlib field name or index tag (case-sensitive). It is strongly recommended to only use indexed fields! This XML element can be repeated if necessary, to search multiple Adlib indexes “simultaneously”; the split-up “parts” (for the separate indexes) of the search query – Adlib only searches one index at a time – will be combined using the Boolean OR.
  - `name`: name for this index for use in CQL (the search query in the SRU request), is case-insensitive. Must contain no . < > & " ' characters.

- `description`: human readable fieldname (in case the `name` is not clear enough yet).

Optional are:

- `sortXPath`: case-sensitive XML path to the contents of the indexed field in the retrieved XML search result, which can be used to sort on this index.
- `sortAllOccurrences`: set this to `true` if sorting has to be done on all occurrences of this field, instead of just on the first occurrence. `False` is the default. If it is `true` then records might be returned more than once if they contain more than one occurrence of the current field.
- `sortField`: if more than one field is defined in the current `sruIndex`, then by default the first field is used for sorting – sorting can't be executed on more than one index. To use another tag or field for sorting, specify it in this `sortField` element.

Below, you can see an example of an `sruSet` specification:

```
<sruSets>
  <sruSet default="true">
    <id>dc</id>
    <contextURI>info:srw/cql-context-set/1/dc-v1.1</contextURI>
    <sruIndexes>
      <sruIndex default="true">
         <field>ti</field>
         <name>title</name>
         <description>Title</description>
         <sortXPath>/record/title</sortXPath>
         <sortAllOccurrences>false</sortAllOccurrences>
      </sruIndex>
      <sruIndex>
         <field>au</field>
         <name>author</name>
         <description>Author</description>
         <sortXPath>/record/author</sortXPath>
         <sortAllOccurrences>false</sortAllOccurrences>
      </sruIndex>
    </sruIndexes>
  </sruSet>
</sruSets>
```

A complete SRU configuration

The following is an example of a complete SRU configuration in the *adlibweb.xml* file:

```
<SRUConfiguration>
<!--  Adlib db settings   -->
<databasepath>D:\demo.adlibsoft.com\IS3\data</databasepath>
<database>document</database>

<XMLType>structured</XMLType>
<sruFields>
  <allDataDictionaryFields />
</sruFields>

<!--  optional additional SRU profiles. Defaults: SRU Base Profile
& Adlib Base Profile  -->
<sruProfile>extraProfile1-URI</sruProfile>

<!--  SRU settings   -->
<limit>10</limit>
<!--  defaultNumRecs   -->
<maxlimit>100</maxlimit>
<!--  maxRecs   -->

<!--  textual information   -->
<sruTitle>Adlib document database</sruTitle>
<!--  optional   -->
<sruContact>Email: info@adlibsoft.com Address: Postbus 1436
Maarssen</sruContact>
<sruDescription>My SRU database</sruDescription>
<sruExtent>Extent</sruExtent>
<sruDataAuthor>Adlib</sruDataAuthor>
<sruHistory>history</sruHistory>
<sruLangUsage>Database contains records in Dutch and English
 </sruLangUsage>
<!--  if sruLangUsage is supplied: must contain space-separated
list of RFC-1766 country codes   -->
<sruLangUsageCodes>nl en</sruLangUsageCodes>
<sruRestrictions>There are no restrictions on the use of this
data</sruRestrictions>

<!--  record schemas   -->
<sruSchemas>
<!--  at least one schema required   -->
  <sruSchema default="true">
    <!--  must be filled. id and URI must be CQL-able and XML-able:
     no . < > & " ' characters! -->
    <id>dc</id>
    <description>Dublin Core</description>
    <schemaXSD>http://www.loc.gov/standards/sru/dc-schema.xsd
     </schemaXSD>
    <!--  should be full URL path  -->
    <schemaURI>info:srw/schema/1/dc-v1.1</schemaURI>
    <!--  optional  -->
    <stylesheet>..\sru\dc.xsl</stylesheet>
    <!--  relative to localdir. if not available: adlibXML  -->
  </sruSchema>
</sruSchemas>

<!--  context sets  -->
<sruSets>
  <!--  at least one set required  -->
  <sruSet default="true">
    <!--  only one set can be default -->
    <!--  must be filled. id, URI and sruIndex/name must be
     CQL-able and XML-able: no . < > & " ' chars! -->
    <id>dc</id>
    <contextURI>info:srw/cql-context-set/1/dc-v1.1</contextURI>
    <sruIndexes>
      <!--  at least one index per set required -->
      <sruIndex default="true">
        <!-- note: default-attr is only useful if set is default-->
        <!-- must be filled -->
        <field>ti</field>
        <!--  Adlib field or index tag. Case sensitive -->
        <name>title</name>
        <!--  CQL index. Case insensitive. Must contain no
         . < > & " ' characters! -->
        <description>Title</description>
        <!--  human readable field name -->
        <!--  optional -->
        <sortXPath>/record/title</sortXPath>
        <!--  case sensitive -->
        <sortAllOccurrences>false</sortAllOccurrences>
        <!--  false is the default -->
      </sruIndex>
      <sruIndex>
        <!-- default-attr is only useful if set is default -->
        <!--  must be filled -->
        <field>au</field>
        <!--  Adlib field or index tag. Case sensitive -->
        <name>author</name>
        <!--  CQL index. Case insensitive. Must contain no
         . < > & " ' characters! -->
        <description>Author</description>
        <!--  human readable fieldname -->
        <!--  optional -->
        <sortXPath>/record/author</sortXPath>
        <!--  case sensitive  -->
        <sortAllOccurrences>false</sortAllOccurrences>
        <!--  false is the default -->
      </sruIndex>
    </sruIndexes>
  </sruSet>
</sruSets>
</SRUConfiguration>
```

4.2 Some characteristics of SRW/SRU requests

Both protocols, SRW and SRU, define the following three operations:

- **searchRetrieve** – Use a `searchRetrieve` operation to submit a search query to the database via CQL. These may be complex queries. Do note that the server doesn't have to implement every aspect of CQL. The client can have the result (records) produced in one of the formats specified by the database owner. This may be Dublin Core for instance, or MarcXML and such.
  You can try this out at our online demo of wwwopac, for instance with a truncated search for "na" in titles:
  `http://demo.adlibsoft.com/internetserver3/wwwopac/wwwopac.exe?version=1.1&operation=searchRetrieve&query=dc.title='na'*&recordSchema=dc&startRecord=2&maximumRecords=12`
- **scan** – A scan retrieves a list of index terms that complies with the request. (However, this operation has not been implemented in wwwopac.)
- **explain** – An explain request retrieves information about the database, such as (at least) its location, what the database contains, and which features of the protocol are being supported by the server. The result is in the form of a ZeeRex record.
  You can try this out at our online demo of wwwopac:
  `http://demo.adlibsoft.com/internetserver3/wwwopac/wwwopac.exe?version=1.1&operation=explain`

In any case, the `searchRetrieve` request must have the following parameters:

- *query* – This parameter contains a CQL string as the search statement, and is necessary in a search request.
- *version* – This is the other mandatory parameter in a request. It specifies the highest version of the SRW protocol that can be understood by the client.

Other (optional) parameters offer more control over the search result, among which: *maximumRecords, startRecord, recordSchema,* and *sortKeys*. For a complete list of parameters, and the specification of the syntax of an SRU request, see: http://www.loc.gov/standards/sru/sru-spec.html#parameters.

See http://www.loc.gov/standards/sru/ for more information about SRW/SRU and CQL.

4.3 Testing the SRU Server

Rob Sanderson (azaroth@liv.ac.uk) has implemented a set of XSLT stylesheets that convert the XML output of any SRU server to browsable HTML. These stylesheets are released under GPL (General Public License) and have been modified slightly by Adlib to be a bit more usable. They are a very easy way of testing your SRU configuration. Please contact our helpdesk if you want to receive these stylesheets and information on how to use them. You will need a web browser capable of client-side XSL transformation; almost every modern browser has this capability.

5 Writing to a database, from the web

You can allow users of your web site to write to one or more of your databases. Therefore you should place the assignment `<write_allowed>true</write_allowed>` in the web configuration file. If you do this at the top of that file, in the `<globalConfiguration>`, the assignment applies to all databases. If you just want to make specific databases writable, then enter the assignment just under the relevant headers, in a `<databaseConfiguration>` or in a `<groupConfiguration>`, not at the top.
To write data, you'll have to send a POST request from the website with `DATABASE=name_of_targetdatabase&DATA=A_CGI_escaped_UTF-8_adlibXML_document&isutf8=1`.

> Since a GET request (URL+data) can only have a limited length – assume 1024 characters, although that is not set down exactly in the HTTP protocol – you can only use this when you are sure that very little data is being written. So it's safer to always use POST.

Note that this functionality has not been applied in the Adlib Internet Server 3 model web application.

5.1 CGI escaping

An adlibXML document which you want to write to the database, first has to be encoded in UTF-8, in which letters with accents are represented in two bytes, and after that, escaped for CGI. The latter means that of each character the hexadecimal ASCII value should be taken, with a %-character in front of it, but scripting languages have functions for this. If there are characters in the data itself that can be confused with XML characters, such as <, > en &, then those should be escaped according to the XML rules, before CGI escaping, namely to: `&lt;`, `&gt;` and `&amp;` respectively.

The following (Dutch) record for example:

```
<record>
  <priref>10</priref>
  <author>tweeën</author>
</record>
```

will have to be send as:

```
data=%3crecord%3e%3cpriref%3e10%3c%2fpriref%3cauthor%3etwee
%c3%abn%3c%2fauthor%3e%3c%2frecord%3e
```

Note that this is only a very brief example; in a real adlibXML document this record should be in between:

```
<adlibXML><recordList>..</recordList></adlibXML>
```

5.2 Writing, updating or deleting records

The adlibXML document that you send to the database, can contain one or more records. Each record is treated separately and differently, dependent on the value of the priref in it:

- If the priref is greater than 0, the record will be created if the record number did not yet exist, and filled with the provided data. If the record number does already exists, then that record will only be updated with the provided fields in the new data; all fields that are not mentioned in the new data remain as they are in the existing record. For instance, regard the following existing record in the database:

```
<record>
   <priref>10</priref>
   <author>Bert</author>
   <author>Hedzer</author>
   <title>A book</title>
  </record>
```

  and the record you want to write is as follows:

```
<record>
   <priref>10</priref>
   <author>Hedzer</author>
   <price>100</price>
</record>
```

  then in the database the following record results:

```
<record>
   <priref>10</priref>
   <author>Hedzer</author>
   <title>A book</title>
   <price>100</price>
</record>
```

- If the priref is 0 or missing, then a new record is a created; the priref of the resulting database record will automatically be determined.
- If the priref is smaller than 0, then the minus is removed and the record with the resulting priref will be removed from the database.

5.3 After writing a record

After writing a record, wwwopac sends an adlibXML document back as a reply. Per processed record, the record as it has become in the database, will be returned (except when a record has been deleted, then the deleted record is returned), plus possibly an error code (the `stat` attribute), for example:

```
<record stat="34">
   <priref>10</priref>
   <author>Hedzer</author>
   <title>A book</title>
   <price>100</price>
</record>
```

In the following table you'll find frequently occurring errors:

stat=	**Cause**	**How to handle this**
0	No errors; everything's okay.	
1	A tag was not found.	Check the assignment `DATABASE=…`
11	Database is not open.	Check the permissions and the assignment `DATABASE=…&DATABASEPATH=…`.
19	Dongle error.	Put *adlib.lic* in the proper folder.
31	Priref has a value that exceeds the limits of the dataset.	Choose a different priref or adjust the limits of the dataset.
34	The record is (already) locked.	Use the record lock manager in Adlib Designer to remove the lock, if you are sure that at the moment no-one is editing the record.
145	Useblock write error.	Check whether IIS has permission to write to the database.
147	A fieldname has not been found.	Check the assignment `DATABASE=…`

157	A duplicate term was submitted.	Check the thesaurus terms.
196	Time-out; processing the write action took too long.	Use a faster computer, or make a bug report when it concerns faulty syntax in the adlibXML document.
301	Parsing-error; there is a syntax error in the adlibXML document.	Make a bug report because it concerns faulty syntax in the adlibXML document.
303	MSXML error.	Check the installation of MSXML4.0 and the XML syntax.

If a record contains an error, the next record will be processed normally. So all record actions that yield *stat="0"* are successful. Apart from this status code per record, there can of course also occur a global error, which can be found in the diagnostics part of the adlibXML document (XPath: `/adlibXML/diagnostics/error/code`).

6 Related issues

6.1 Status

Use the `Status` command to generate general information about the server; the result is an XML file.

In CGI the call is:

```
http://mylocalhost.com/webdata/wwwopac.exe?Status=1
```

This results in something similar to the following:

```
<?xml version="1.0" encoding="UTF-8" ?>
<status>
  <dllCreateDateTime>May 31 2005 20:59:13</dllCreateDateTime>
  <build>Unicode</build>
  <moduleName>D:\demo.adlibsoft.com\InternetServer3\demo\wwwopac\
   wwwopac.exe</moduleName>
  <moduleBuildVersion>Version 6.1.0.100</moduleBuildVersion>
  <buildVersion>Unified Version 6.1.0, build 100</buildVersion>
  <operatingSystem>Windows 2003 server 5.2 (Build 3790) Service
   Pack 1</operatingSystem>
  <clientName>ADLIB Internal use only</clientName>
  <processID>2412</processID>
  <threadID>3444</threadID>
  <numberOfConnections>1</numberOfConnections>
  <numberOfActiveConnections>1</numberOfActiveConnections>
  <computerName>EUROPA</computerName>
  <userName>IUSR_EUROPA</userName>
  <protocolVersion>601</protocolVersion>
  <runningTime>0.0</runningTime>
  <totalRequestTime>0.0</totalRequestTime>
  <load>0.000</load>
  <availableDLLs>
  <dll name="adlibimg.dll">0x00e20000</dll>
  <dll name="adapl.dll">0x00000000</dll>
  <dll name="adaplU.dll">0x00000000</dll>
  <dll name="gdiplus.dll">0x4dd60000</dll>
  <dll name="unicows.dll">0x7f2d0000</dll>
  <dll name="rtf2xml.dll">0x01710000</dll>
  </availableDLLs>
  <localWebDirectory>D:\demo.adlibsoft.com\InternetServer3\demo\
   wwwopac</localWebDirectory>
  <currentDirectory>D:\demo.adlibsoft.com\InternetServer3\demo
   </currentDirectory>
  <loadedConfig>
  <localDirectory>D:\demo.adlibsoft.com\InternetServer3\demo\
   wwwopac</localDirectory>
  <configFile timestamp="1150404402">D:\demo.adlibsoft.com\
   InternetServer3\demo\wwwopac\adlibweb.xml</configFile>
  </loadedConfig>
  <connections>
  <connection>
  <handle>0x00d14008</handle>
  <remoteAddress>10.1.1.155</remoteAddress>
  <queryString>http://demo.adlibsoft.com:80/internetserver3/
  wwwopac/wwwopac.exe?Status=1</queryString>
  <numberOfRequests>0</numberOfRequests>
  <numberOfErrors>0</numberOfErrors>
  <running>yes</running>
  <myself>yes</myself>
  <global>yes</global>
  <lockInfo>
  <locks />
  <lockStatistics>
  <maxWait>0</maxWait>
  <totalWaits>0</totalWaits>
  <numberOfLockCalls>0</numberOfLockCalls>
  <numberOfUnlockCalls>0</numberOfUnlockCalls>
  <maxNumberOfTicksLocked>0</maxNumberOfTicksLocked>
  <numberOfLengthErrors>0</numberOfLengthErrors>
  <numberOfUnlockErrors>0</numberOfUnlockErrors>
  </lockStatistics>
  </lockInfo>
  </connection>
  </connections>
  <memory>
  <system>
  <commitTotal>271056</commitTotal>
  <commitLimit>3057116</commitLimit>
  <commitPeak>307660</commitPeak>
  <physicalTotal>1048008</physicalTotal>
  <physicalAvailable>584360</physicalAvailable>
  <cache>796300</cache>
  <kernelTotal>125340</kernelTotal>
  <kernelPaged>97192</kernelPaged>
  <kernelNonPaged>28148</kernelNonPaged>
  <handleCount>9796</handleCount>
  <processCount>32</processCount>
  <threadCount>469</threadCount>
  </system>
  <process>
  <work>7675904</work>
  <pageFaultCount>2111</pageFaultCount>
  <peakWork>7712768</peakWork>
  <quotaPeakPagedPoolUsage>37500</quotaPeakPagedPoolUsage>
  <quotaPagedPoolUsage>37400</quotaPagedPoolUsage>
  <quotaPeakNonPagedPoolUsage>7040</quotaPeakNonPagedPoolUsage>
  <quotaNonPagedPoolUsage>5832</quotaNonPagedPoolUsage>
  <pageFileUsage>4255744</pageFileUsage>
  <peakPageFileUsage>4362240</peakPageFileUsage>
  </process>
  </memory>
</status>
```

6.2 Field names

Do not use reserved words (from the expert search language) as field names in your Adlib databases, to prevent errors. These words are: adapl, adlibce, all, nd, as, ascending, contains, date, descending, diagnostic, end, equals, flat, from, generic, greater, key, limit, nand, narrower, not, numeric, occurrences, or, pointer, print, profile, raw, record, related, resolved, set, smaller, sort, startfrom, structured, style, text, to, transformed, tree, unstructured, when, xml.
In field names, never use spaces or the & character. And preferably do not use the following characters either: **" ` @ = [ ] { } < > / ^**
Preferably, only use letters and digits, and possibly the following three characters: **- _ .**

6.3 Multi-lingual fields

Fields can be multi-lingual in different ways. Dependent on the language in which a visitor/user opens the web application, data is always automatically searched in the right language and found records will be displayed in the same language; for multi-lingual fields which do not contain a value in the relevant language, but do in the so-called invariant language, the value *can** be displayed in the latter language in the record. Inside such an application, language preference is usually set by clicking a flag icon.

In a wwwopac search query, you can use the `LANGUAGE=<x>` parameter in two different ways:

- You can replace `<x>` by an Adlib language number. This way you only specify the presentation language of retrieved records, which is required to be able to display the proper translation of an enumerative field for example. This way you do not set the search language: all languages of each multi-lingual field will be searched.
- You can replace `<x>` by a standard language code, for instance `[en-GB]` for English or `[nl-NL]` for Dutch, to provide the language in which all multi-lingual fields must be searched. For Adlib, the first two letters of such a code specify the presentation language. This only works in an Adlib SQL or Oracle database.

* Only if you include `INVARIANT=true` in the search query, then in the search result the empty field (in the active language) will be replaced by the contents of that field in the invariant language.
If you do not specify a search language parameter in the search query, then all languages of each multi-lingual field will be searched,

yet of every multi-lingual field in found records only the value in the invariant language can be displayed if you include `INVARIANT=true` in the search query; if you do not specify a search language parameter nor include `INVARIANT=true`, then multi-lingual fields will be retrieved empty and of found records you will only see the contents of non-multi-lingual fields. (The invariant language of a multi-lingual field in a record is nothing more than the language in which the first value in that field in that record was entered.)

In a wwwopac search on an Adlib SQL or Oracle database, you may also provide a language attribute per multi-lingual field, in the format: `tag[language code]=value&…`. A language attribute has priority over the `LANGUAGE` parameter. With a language attribute, the relevant field will only be searched in the indicated language. For every multi-lingual field in a search you may provide a (different) language attribute.

6.4 Dependencies

With a normal installation of the Adlib Internet Server all necessary files will become available. But if by accident certain files are removed from your system, then the Internet Server/wwwopac can't function properly anymore. By analysing the occurring errors, you may be able to point to the missing file:

- For the processing of `imagemetadata` and `thumbnail`, Internet Server uses the Adlib file *adlibimg.dll* and the Windows extension *gdiplus.dll*. Without (one of) these files only said processing doesn't work.
- For the processing of SDI mail, wwwopac uses the familiar SMTP dll's, but it also works without them.
- When MSXML4.0 (or higher) is not installed, the Internet Server doesn't work because then it can't process XML operations.

7 Appendix 1: your own record format

Search results are produced in an XML file that complies with the so called *adlibXML.xsd* (Extensible Schema Definition: a validation schema for our own XML-code) for Spectrum-XML.

From wwwopac 6.1 it is possible to define your own AdlibXML record format. This may be desirable if the standard `STRUCTURED` and `UNSTRUCTURED` record formats are not sufficient for your application.

You define such an XML schema in the *adlibweb.xml* configuration file for your wwwopac. (This functionality is not supported for *.www* configuration files, the predecessor of *adlibweb.xml*.) A handy way to define such a schema is by basing it on an existing XML schema.

By submitting a query to wwwopac directly as a CGI string in your browser, the search result will be presented in XML. Of course, you can indicate whether you want that result to be in `STRUCTURED` or `UNSTRUCTURED` format. The specification of the format itself can be requested at the same time, by submitting one of the options `schema=template` or `schema=text` in the CGI string as well:

- **`schema=template`** displays the specification of the used schema in the *<recordTemplate>* node, above the record list. Metadata is transported through attributes, as usual.
- **`schema=text`** shows the specification of the used schema (fields, links, types etc) in comment text in XML, above the record list.

Via the XML record schema displayed here, you'll also find out what the names of the current tag nodes are, and which tags might be retrieved in the search result. Select and copy the complete *<recordTemplate>* and paste it in your *adlibweb.xml* file as a child of a *<groupConfiguration>* node or *<databaseConfiguration>* node. (Do not forget to escape characters like "&", when doing so. And if you copy and paste from within the browser, then also remove the "-" signs which are there to fold nodes in and out.) Now you can edit the schema to your desire.

- The top node is always *<recordTemplate>*.
- The subnodes consist of a *<node>* with different possible attributes, namely:
    - o `tag` – the database tag to be retrieved;
    - o `type` – the field type, namely one of the following:

    - ▪ `normalField` – this is the default type;
    - ▪ `group` – for fields to be grouped;
    - ▪ `richTextField` – converts line-breaks in this kind of fields to *<br/>* tags;
    - ▪ `RTF` – converts the contents of RichText fields to XML, via *rtf2xml.dll*;
    - ▪ `referenceLink` – for linked fields;
    - ▪ `enumeratedField` – for enumerative fields;
    - ▪ `reservedTag` – for including options;
    - ▪ `termLink`;
    - ▪ `attribute` – for including attributes;
    - ▪ `occurrence`;
    - ▪ `text` – to include fixed text, no element;
    - ▪ `attributeText`;
    - ▪ `expandedField` – for internal links, reads in the linked record and displays it recursively. Subnodes (like in the example below) will be ignored! All types of internal links can be expanded this way. However, there is no automatic check for circular links.
  - o `name` – the XML element to be used. If the `name` is not provided, the tag itself will be used; then, any percent sign in the tag will be escaped as "`perc`". The `name` has to be XML-compliant, so spaces cannot be used in it;
  - o `linkdb` – only necessary for `referenceLink` and `termLink`;
  - o `linktag` – only necessary for `referenceLink` and `termLink`;
  - o `lreftag` – only necessary for `referenceLink`;
  - o `text` – the text which must be used for the node types `text` and `attributeText`.
- Nodes of the group `referenceLink` and `termLink` type may have subnodes.

- When a node attribute has subnodes, those will have to be included as the first subnodes. Attribute subnodes after the first “normal” subnode won’t be displayed!

Below, you can see a fictive example, for test purposes:

```
<groupConfiguration group="template">
  <recordTemplate>
   <node tag="QQ">
    <node type="attributeText" name="staticAttr" text="xxx"/>
    <node name="occ" type="occurrence"/>
    <node type="text" text="drieën &lt; 10"/>
    <!-- this text is not shown, see last remark in the syntax
     description -->
    <node type="attributeText" name="notshown" text="yyy"/>
    <node tag="QQ" name="QQattr" type="attribute"/>
    <node tag="WW" name="WWattr" type="attribute"/>
    <node type="text" text="end"/>
   </node>
   <node type="expandedField" tag="bt" name="part_of_reference"
    termtag="IN">
   <node type="group" name="mygroup">
    <node type="richTextField" tag="WW" />
    <node tag="GG" />
   </node>
   <node tag="%0" name="priref" />
   <node type="referenceLink" tag="AA" name="thename" lreftag="l1"
    linktag="BB" linkdb="d:\mydbA+yourdb">
    <node tag="AA" name="Oliver" />
    <node tag="D1" name="mergedfield" />
   </node>
   <node tag="AI" name="autoinc" />
   <node tag="DD" name="date" />
   <node type="element" name="extra">
    <node tag="DD" name="date" />
    <node type="text" name="fixed text">
   </node>
   <node type="enumeratedField" tag="EE" />
   <node tag="FF" />
   <node type="group" name="yourgroup">
    <node tag="H1" name="Heylang1" />
    <node tag="H2" name="Heylang2" />
    <node tag="H3" />
   </node>
   <node tag="ID" name="ISO_date" />
   <node tag="II" name="Integer" />
   <node tag="IS" />
   <node tag="LL" />
   <node type="termLink" tag="TI" linktag="ti"
    linkdb="d:\mydbA+yourdb">
    <node tag="ss" name="tt" />
   </node>
   <node tag="TT" />
   <node tag="l1" name="linkref_info" />
  </recordTemplate>
```

```
</groupConfiguration>

<databaseConfiguration database="mydb" groups="template">
 <DATABASEpath>c:\something</DATABASEpath>
 <DATABASE>mydb</DATABASE>
</databaseConfiguration>
```

Save the changed *adlibweb.xml* file in Unicode UTF-8 encoding.

In principle, you can also submit your own schema through the CGI string, if you do not want to use a web configuration file. You do this behind an `xmltemplate=…`, in which the entire `<recordTemplate>` node (URL-encoded) must be inserted. Do check if the entire node has been pasted to the URL, since the length of a CGI string is limited.

Do note that if you have defined you own record schema, this new schema will always be used. You then no longer have the choice to sometimes use the standard `STRUCTURED` and `UNSTRUCTURED` formats too. It's also impossible to specify a difference between fields to be retrieved for a brief display and a detailed display. However, this feature may be implemented in the future.

8 Appendix 2: the Adlib Base Profile

Adlib wwwopac implements SRU/CQL version 1.1 and conforms to the so-called SRU Base Profile (see http://www.loc.gov/standards/sru/base-profile.html) and the following Adlib Base Profile:

Adlib Base Profile (version 1.0)

Beta-4

Initiated: Monday December 13th, 2004

Last updated: June 14th, 2006

Hedzer Westra, Erik Lange

Adlib Information Systems

http://www.adlibsoft.com

Contents

  - Relations
  - Terms
  - Modifiers
  - Sorting
- The Adlib Context set
- CQL to Adlib bridging

Introduction

This document describes the Adlib Base Profile 1.0 of the SRU server that is implemented by Adlib Information Systems in its wwwopac software. It is only a partial profile: it only describes the software capabilities. For each installation and configuration of this software a full profile document should be defined (by the client).

URIs

The URI for the Adlib Base Profile is "info:srw/profile/6/1.0".

The Adlib context set is identified by the URI "info:srw/cql-context-set/6/1.0". The preferred identifier for this URI is 'adlib'.

These URIs are defined using the SRW 'info URI' convention, using the Adlib authority string "6", which was assigned by the ZING working group.

SRW, SRU and CQL

The SRW and SRU protocols, and the CQL language, CQL context set and SRU Base Profile are all described on http://www.loc.gov/standards/sru/ . Please refer there for information about this search & retrieval protocol.

SRW Requirements

The Adlib wwwopac.exe implements all features required by the SRU Base Profile, refer to http://www.loc.gov/standards/sru/base-profile.html.

Optional and server-dependent features

SRW/U and CQL are quite broad protocols which allow for many optional and server-dependent features.

This document defines which optional and server-dependent features are implemented, and how these function.

Configuration

Adlib supplies a number of applications to its customers, who can make changes and additions to their applications. Therefore there is no single 'Adlib' context set or profile. Each customer can configure their wwwopac software (through settings in the *adlibweb.xml* file) to produce the SRU context sets and profiles that are required. For any such configuration a custom context set and profile document should be created - or existing ones should be adapted. This document describes the Adlib Base Profile and the Adlib Context Set, which must form the basis of such a customer-defined profile. Therefore, only meta-indexes are defined here. Metadata formats (like Dublin Core or MarcXML) are not defined here.

Implemented SRW/CQL features

- protocol version 1.1
- the SRU protocol, i.e., HTTP GET/POST CGI requests. This is a slight extension of SRU, since the current documentation does not mention SRU POST.
- the explain operation
- the searchRetrieve operation
- CQL 1.1 parsing
- CQL 1.1 handling
- CQL context set as far as the Base Profile requests it, plus some extras:
  - and, or, not Booleans
  - =, >, <, <=, >=, <> relations
  - exact, all, any, scr relations
  - the within relation
  - cql.anywhere, adlib.allIndexes and adlib.record meta-indexes
  - cql.serverChoice surrogate index
- sorting
- surrogate & non-surrogate diagnostics generation
- recordSchemas
- request echoing, xSortKeys and XCQL

Unimplemented optional features

- recordXPath handling
- result sets
- full proximity searches, i.e. the 'prox' boolean
- word anchoring (^)
- matching on a single character using '?'

- scan operation
- SRW (SOAP) as communication layer

Adlib server-specific implementations

Indexes

- The adlib.allIndexes meta-index searches all indexes defined in the Adlib database at once. This is different from cql.anywhere, which searches all indexes in all context sets.

- The adlib.record meta-index searches the whole record, so all of the fields in each record. This includes data that is not indexed (and possibly not even displayed in any record schema) and therefore not searchable using CQL indexes. The relation must be '='.

Note: In future versions, CQL might support a cql.record meta-index with the same semantics.

Relations

- cql.scr is always handled as '='.

- The 'within' relation is implemented using range searching. Exactly two words must be supplied, separated by a single space. The range search type can be selected by using an adlib.range modifier, with the following values:

leftexclusive

rightexclusive

exclusive

inclusive (default)

Example: `date within/adlib.range=leftexclusive "2000 2004"`

Note: the CQL context set always uses inclusive range searching; there is no range modifier.

Terms

- Empty term searches are not supported.

- * for pattern matching is only usable at the beginning and/or end of a search term.

- A maximum of 250 characters per term is supported.

Booleans

- A maximum depth of 250 nested expressions is supported.

Modifiers

- The 'and' boolean accepts one of the modifiers 'adlib.when' and 'adlib.whennot'. See below for more information.

- Thesaurus-enabled searches can be executed by issuing an adlib.thesaurus modifier with one of the following values:

  generic

  broader

  narrower

  related

  topterm

  parents

These only work correctly on indexes with thesaurus links defined. Otherwise, they fall back on normal searching. The modifier is supported only for the 'exact' and '=' relations.

- There are two types of CQL context set modifiers: data type modifiers and pattern modifiers. The accepted data type modifiers are:

  cql.string

  cql.word

  cql.isoDate

  cql.number

Note: cql.uri is invalid in the Adlib Base Profile.

The accepted pattern modifiers are:

  cql.masked

cql.unmasked (not yet defined in CQL context set)

Of each modifier type only one can be active for each search clause. The combinations /cql.masked/cql.number and /cql.masked/cql.isoDate are invalid.

- Adlib interprets terms in the following manner:

If modifiers are sent in the query, those are used. If they are not, modifiers are implied according to the following rules:

1. pattern modifer: cql.masked is always assumed, unless cql.number or cql.isoDate are supplied or implied.

2. data type modifier, implied using the relation and sometimes the Adlib index type:

- 'exact': implied modifier is cql.string.
- '=': implied modifier is cql.word, cql.number or cql.isoDate, depending on the index type.
- 'any' and 'all': implied modifier is cql.word. Words are combined using OR (for 'any') or AND (for 'all').
- 'within': there is no implied data type modifier; there must be two words separated by a single space

Modifier cql.unmasked means: the CQL pattern match character * has no special meaning; pattern matching is not possible using this modifier. The characters ^, ? and * do not have to be escaped.

Modifier cql.masked means: pattern matching on * is possible. The characters ^, ?, * and \ must be escaped with \.

Modifier cql.word means: words are split and then re-combined using the Adlib separators and concatenators rule. Word adjacency is not used when searching. An error will be returned when searching with cql.word on a string index.
Separator characters are: [];,!@()|{}<>? carriagereturn newline space tab
Concatenator characters are: `-=\./~#$%^&_+:"'*
Note: the CQL context set says nothing about how words are to be split but instead leaves that up to implementations to be specified.

Modifier cql.string means: terms are not inspected for separators or concatenators. An error will be returned when searching with cql.string on a word index.

Modifier cql.isoDate means: term is interpreted as a single ISO 8601 date. If the Adlib index type is European or US, the term will be translated before the actual search takes place.

Modifier cql.number means: term is interpreted as a single 32-bit signed integer.

The difference(s) between the CQL context set is/are the following:

- word adjacency, which should be used with the cql.word data type modifier, is a feature which the Adlib database engine currently does not implement.

Sorting

- sorting is only supported for hard-wired (case-sensitive) paths, not for full XPaths. The customer can define a path for each CQL index.

The Adlib Context set (version 1.0)

The Adlib Context Set version 1.0 defines:

(meta-)indexes:

- adlib.record (whether this will be added to the CQL context set is still unclear, so the prefix is adlib, not cql)

- adlib.allIndexes

modifiers:

- adlib.thesaurus for the '=' and exact/cql.word relation with its six accepted values (generic, broader, etc.)

- adlib.range for the 'within' relation

- adlib.when and adlib.whennot for the 'and' boolean

- cql.unmasked (it is assumed this modifier will eventually be added to the CQL context set, hence the cql prefix)

CQL to Adlib bridging considerations

The ‘and’ boolean modifiers:

The and-relation modifiers are implemented using the WHEN and WHEN NOT booleans in Adlib. In Adlib, the WHEN (NOT) booleans on two distinct indexes will first search records using the left operand index. For each record the right operand operator and value will be checked in the same occurrence. The Adlib-specific property of occurrences comes down to the following: each field can have 0 or more values (unless explicitly stated as non-repeatable), whereas conventional RDBMS's can only hold 0 or 1 values in a field. Indexes can be specified to index either the first or all occurrences. The WHEN operator explicitly checks matching occurrences. See the Adlib User Guide (available from http://www.adlibsoft.com/) for more information.

‘within’:

This relation is implemented using range searching, i.e. the Adlib WHEN boolean used on two identical indexes. Note that this is a very different operation from the WHEN boolean used on two distinct indexes.

Performance:

- The adlib.allIndexes and cql.anywhere meta-indexes might have slow search responses if there are a lot of indexes.

- The adlib.record meta-index cannot use any index and will always be slow.

- Searching on *...* is done using the Adlib 'contains' operator, which is slow since no index can be used.

9 Index