



Installation of the Adlib Internet Server web application 5 or the OAI Server

Axiell ALM Netherlands BV

Copyright © 1992-2016 Axiell ALM Netherlands BV® All rights reserved. Adlib® is a product of Axiell ALM Netherlands BV®

The information in this document is subject to change without notice and should not be construed as a commitment by Axiell ALM Netherlands BV. Axiell assumes no responsibility for any errors that may appear in this document. The software described in this document is furnished under a licence and may be used or copied only in accordance with the terms of such a licence. While making every effort to ensure the accuracy of this document, products are continually being improved.

As a result of continuous improvements, later versions of the products may vary from those described here. Under no circumstances may this document be regarded as a part of any contractual obligation to supply software, or as a definitive product description.

Contents

1 Introduction	1
2 Requirements	3
2.1 wwwopac.ashx or oai.ashx on the server(s)	3
2.2 On a work station (client)	5
3 Security recommendations	7
3.1 Server setup	8
4 The installation procedure	13
4.1 Step 1: create separate folders	13
4.2 Step 2: IIS 7 setup under Windows Server 2008	14
4.3 Step 3: install Adserver	21
4.4 Step 4: modify adlibweb.xml	26
4.5 Step 5: modify globalsettings.xml	26
4.5.1 <dataservers>	27
4.5.2 <emailservers>	28
4.5.3 <languages>	29
4.5.4 <documents>	30
4.5.5 <session>	32
4.5.6 <session><reserve>	40
4.5.7 <session><sdiProfiles>	42
4.5.8 <session><comments> and <tagging>	44
4.5.9 <displaylist>	44
4.6 Some possibilities of formsettings.xml	45
4.7 Functionality for comments and tags	49
4.7.1 Setup of parts	50
4.7.2 Functionality in the Internet Server interface	57
4.7.3 Comments and tags database management	59
4.8 Customization examples	60
4.8.1 Adding a field to the searched fields	60
4.8.2 Adding a field to the detailed display	65
4.9 Visiting the Internet Server website locally	72
5 User authentication	75
5.1 Setting up SQL authentication	76
5.2 Windows authentication with Active Directory	84
5.3 Excluding internet users from specific records	91
5.3.1 Setup in IIS 7 (Windows Server 2008)	91
5.3.2 Further setup in SQL Server Management Studio	95
5.3.3 Excluding records via the Adlib application	96
6 File types	97

7 Adlib OAI Server	101
7.1 Installing oai.ashx: global setup	102
7.2 MetadataFormats	108
7.3 Creating an XSLT stylesheet for Adlib OAI-PMH	109
7.4 Testing and validating your OAI repository	114
7.4.1 <i>What if every OAI request generates a login error?</i>	115
7.5 OAI repositories and search engines	116
7.6 How to harvest an OAI repository	116
7.6.1 <i>Basic principle</i>	117
7.6.2 <i>Protocol requests</i>	117
7.6.3 <i>Harvesting in batches</i>	119

1 Introduction

Adlib Internet Server 5 (AIS 5) is a complete package to make library catalogues, museum collections and archives accessible through the Internet. Via this application, your website visitors can search for records in a collection, get detailed information and possibly print it. And much more.

AIS 5 comes to replace version 3.1.2 and was rebuilt from scratch using a new framework (ASP.NET MVC).

The Adlib webopac (*wwwopac.ashx* aka the Adlib API) is a .NET HTTP handler which implements the Adlib functionality for web applications based on an Adlib SQL database, and is installed as supplementary program to an existing web server (IIS 6.0 or higher for Windows Server 2003 or 2008).

But the program has no graphical interface, so therefore a (web) application is needed to address the *wwwopac*. Such an application can be a website on which visitors can search your catalogue via the internet, and maybe place reservations, or take care of other business. For this, the standard Adlib Internet Server web application (which can largely be adjusted to your preferences) is available.

An Adlib Internet Server web application may be part of the package you are installing, but with this HTTP handler chances are you are building your own (web) application and you probably want to install the *wwwopac.ashx* server separately.

See the [wwwopac installation guide](#) for information limited to what you need to know to just install *wwwopac.ashx* (or the older *wwwopac.exe*) on a server. In the Adlib Internet Server installation guide before you, you'll find information on how to install a complete Internet Server package.

OAI functionality is available as separate software, in the shape of Adlib *oai.ashx* (for Adlib SQL databases only) or the older *oaiserver.exe*. The OAI Server doesn't need a web application (from your side). See chapter 7 in this manual for all information about the *oai.ashx* implementation and setup of the Adlib OAI Server, or see chapter 3 in the *WWWOPAC reference guide* for general information about *oaiserver.exe*. The OAI Server can possibly be downloaded from the Adlib website or requested from the Adlib Helpdesk.

2 Requirements

You will need the following software on the server to be able to use the Adlib webopac 6.6.0 or higher in combination with the Adlib Internet Server web application 5, or Adlib OAI Server:

2.1 *wwwopac.ashx* or *oai.ashx* on the server(s)

- minimally Windows Server 2003 or Windows Server 2008 or higher. The Adlib Internet Server 5 does not function under Windows Server 2000, Windows NT4, 95, 98, or ME.
- an Adlib application (with a subfolder *\data*); although an application is not strictly necessary, just the *\data* folder is sufficient. A requirement is that the *\data* folder is actually accessible. If that folder resides locally on a server, this accessibility is in principle not a problem. However, should the folder be located on a different server, then you have to check whether the access rights to the share and the ntfs rights to the relevant folder have been set up properly. For access to the relevant share, by default the account is used under which the application pool is running. For OAI Server, for the *\data* subfolder at least read-only access rights must be set for web users.
- an Adlib SQL Server or Adlib Oracle database.
- the *wwwopac.ashx* (for Internet Server) or *oai.ashx* (for OAI Server), plus accompanying files delivered with the package. Either file package is suited for both 32-bit as well as 64-bit operating systems.
(Internet Server 5 doesn't work with the older *wwwopac.exe*.)
- an *adlib.lic* file, containing your product license.
- http server software must be installed on the servers on which *wwwopac.ashx* and the web application will be placed, such as IIS 6.0 for Windows Server 2003 or IIS 7.0 for Windows Server 2008. This makes it possible for workstations (client side) to access pages from the server. For the required Windows versions, these services are available: already installed, or on the Windows installation cd's.

All of our web applications use IIS and Active Server Pages. Other web servers with scripting support (e.g. Apache and PHP) can also be used, but in these cases customers themselves will have to provide the middleware scripts that are used in the multi-layer

structure of Adlib Internet Server applications. The Microsoft XML-parser (MSXML4) is being used for processing XML. This software can be downloaded free from the Microsoft website.

- MSXML4. Version 4.0 SP2 of the MSXML parser from Microsoft should be installed. You can check this in your registry (*Start > Execute*, type `regedit` and click *OK*). In it, search for the text `MSXML`. If the parser is not present, then you can download this software from the Microsoft web site (<http://www.microsoft.com>); from their homepage search for `MSXML4`. Open the relevant download page, choose the `msxml.msi` file to download and store it on your hard disk. Now install the software by double-clicking the file.
- The Microsoft .NET Framework 4.0 (the *Extended* or *Full* version) must be installed on the server, yet only when IIS has already been installed (with the latest security updates), otherwise some important features of ASP.NET will be missing from the installation. So on a new server, always install IIS before you install the .NET Framework. For information about .NET 4.0, see: <http://msdn.microsoft.com/en-us/vstudio/aa496123>. (If .NET 4.0 still has to be installed, then please take into account that the web server might need rebooting after this installation.) On IIS 7, ASP.NET must operate in integrated mode (which is the default configuration). The application pool which we will create for the `wwwopac.ashx` or `oai.ashx` server later on in this manual, must run in this mode.
It is also a requirement that physical or virtual folders above the `wwwopac.ashx` or `oai.ashx` folder do not run in earlier versions of the .NET Framework, so .NET 2.0 application pool must not contain applications or folders using .NET 4.0.
- ASP .NET MVC Framework 4 must be installed on the server as well. That Microsoft software can be downloaded for free, from: <http://www.asp.net/mvc/mvc4>.
- If Active Directory authentication will be used for access to the database, instead of SQL authentication, then the application pool must be configured to use an account which has access to the SQL Server.
- Use servers with at least a dual-core Intel Pentium processor, and 2 GB RAM or more.
- `Adserver.exe` (not for the OAI server): the installation procedure has been designed for a workstation or server on which the Adlib program `adserver.exe` has not been installed yet. (Adserver is used for the Loans module and for making online reservations.)

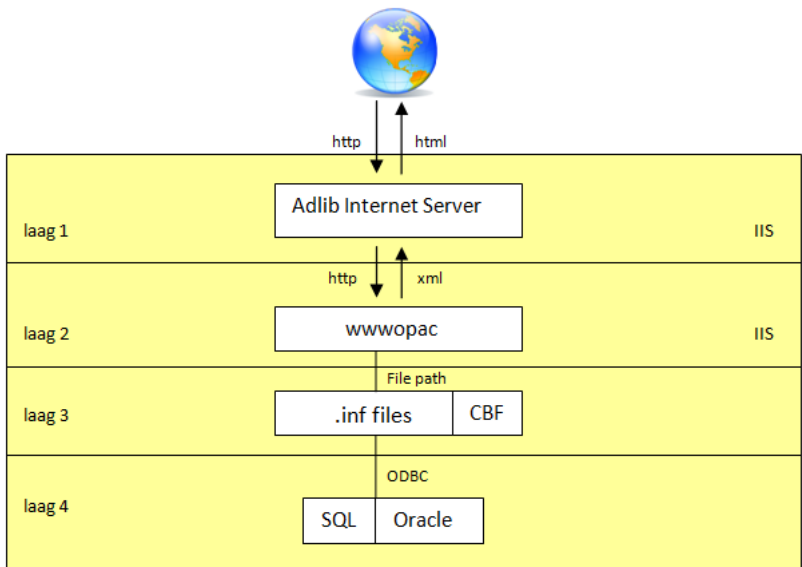
2.2 On a work station (client)

- a web browser such as Microsoft Internet Explorer or Mozilla Firefox. This lets you view web pages.
- a network connection to the server on which the web application is located.

3 Security recommendations

An Adlib Internet Server system consists of four software layers, namely:

1. the Internet Server web application, which embodies the (graphical) interface between computer users and the webopac (the Adlib core software);
2. the webopac, namely *wwwopac.ashx* or the obsolete *wwwopac.exe*, which processes commands over http (coming from the user-friendly Internet Server web application, an application of your own or entered directly into the address field of a browser), retrieves the proper data from the database, and exports it as XML;
3. the Adlib *.inf* files (database structure files) which must be accessible to *wwwopac* via the file system, and possibly the Adlib CBF databases (the latter in case you are not using an Adlib SQL Server or Adlib Oracle database). The *wwwopac.ashx* doesn't work with CBF.
4. the Adlib SQL Server or Adlib Oracle database which contains all records (in case you are not using Adlib CBF databases).



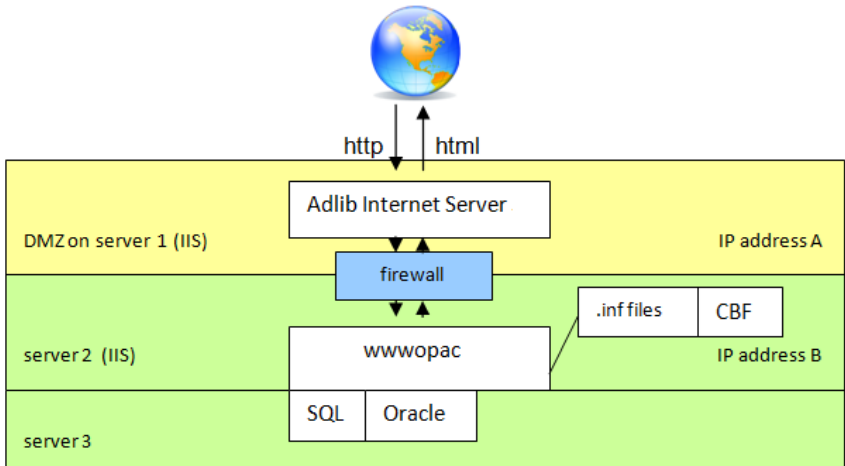
The advantage of this structure is that each layer may be installed on a different server, to optimize the security of your network and databases. (For the functioning of the Adlib system, it doesn't matter whether you distribute the four layers over one, two, three or four servers.)

3.1 Server setup

Although the databases can be protected against illegal access in different ways, it is recommendable to set up your Adlib Internet Server environment in such a way that security is handled on network level as well, by using the defences of servers. This is possible because the four software layers can be distributed over as many physical (or virtual) servers, and because the communication between those servers can be secured by means of hidden IP addresses and firewalls. The only* Adlib software layer which needs to be accessible to internet users directly (over http), is the Internet Server web application layer. So, you can place the Adlib Internet Server on a different server (let's call it server 1 for now) from `wwwopac**` (on server 2), the `.inf` files (on server 3), and the databases (on server 3 or 4). The communication between server 1 and 2 will then proceed through IP addresses of both servers, hidden from internet users. A firewall can secure that communication even more, making it impossible for internet users to approach `wwwopac` or the databases directly.

Even if your Adlib Internet Server system is only available to co-workers on a (secured) local network, it may be useful to install the databases on another server, as a form of extra protection against external or internal hacking attempts.

Place the Adlib Internet Server in the DMZ (demilitarized zone or data management zone) of the server which makes web services available to the internet. A DMZ is a subnetwork which adds an extra security layer to your local network: external web users can only access the DMZ, not the entire network. Also, no direct communication should take place between Adlib Internet Server in the DMZ and `wwwopac` on the other server; that communication should be shielded by a firewall. Normally, port 80 is used for http, the protocol with which Adlib Internet Server communicates with `wwwopac`; therefore, port 80 should be open in the firewall.



* If you make an OAI repository available as well, then “users” (usually consisting of special search engines) must have access to the oa-server too, since OAI queries are not handled by an Adlib web application. For the security of your databases it is nonetheless wise to place the databases on a different server, and safeguard the data traffic between both servers through a firewall. The same applies if you offer an SRW/SRU web service to make your database(s) accessible on the internet.

** If users of your web application need to be able to retrieve images (of which your application indicates the address as URLs) stored on server 2, then it’s best to use an IIS reverse proxy to access the webopac and images. A reverse proxy is used to forward outside http requests from the DMZ through a firewall on port 80 to another server within a domain:

1. You will first have to install *URL Rewrite* using the IIS Web Platform Installer from <http://www.iis.net/download/urlrewrite>.
2. After installation of the IIS Web Platform Installer, it will be launched automatically. Type the words *URL Rewrite* in the search box in the top right and press **Enter**.
3. *URL Rewrite* will turn up in the search result. Click the *Add* button.
4. Also search for *Application Request Routing* and *Add* it.
5. Click the *Install* button and accept the license agreement. Finish and exit the Web Platform Installer after successful installation.
6. Start your IIS Manager (*Control panel > Administrative tools > IIS Manager*) and create a new *Site* for which you want to activate a

reverse proxy: in Windows Server 2008, right-click the *Sites* node and choose *Add web site* in the pop-up menu.

This site will not have any data or application files. For the *Site name*, enter a name which will only be used internally as the first part of your new host header, for example `wwwopac` or another unique name. As the *Application pool*, preferably choose one exclusively to this new site: create one first if it doesn't exist yet. You can leave that application pool to its default settings since nothing really happens in this site, except for the URL rewriting. *Physical path* can be left empty. *Binding type*, *IP address* and the *Port* can be left to their default settings. Your *Host name* will become something like `wwwopac.ourinstitution.org`. Start the web site immediately and click *OK*.

7. Select the new web site in the IIS Manager and in the right window pane, start *URL Rewrite*.

8. In the *URL Rewrite* window, click the *Add rule(s)* option in the column on the right.
9. Select *Reverse proxy* in the *Add Rule(s)* window, click *OK* and accept the activation for application request routing.
10. In the *Add reverse proxy rules* dialog, fill in the destination server name (where the webopac resides – this can be a server inside the domain, behind a firewall) in the *Inbound rules*, for example `adlibappserver.ourinstitution.local`.

Add Reverse Proxy Rules [?] [X]

Inbound Rules

Enter the server name or the IP address where HTTP requests will be forwarded:

Example: contentserver1

Enable SSL Offloading
Selecting this option will forward all HTTPS requests over HTTP.

Outbound Rules

Rewrite the domain names of the links in HTTP responses

Responses that are generated by applications that are behind a reverse proxy can have HTTP links that use internal domain names. These links must be updated to use external domain names.

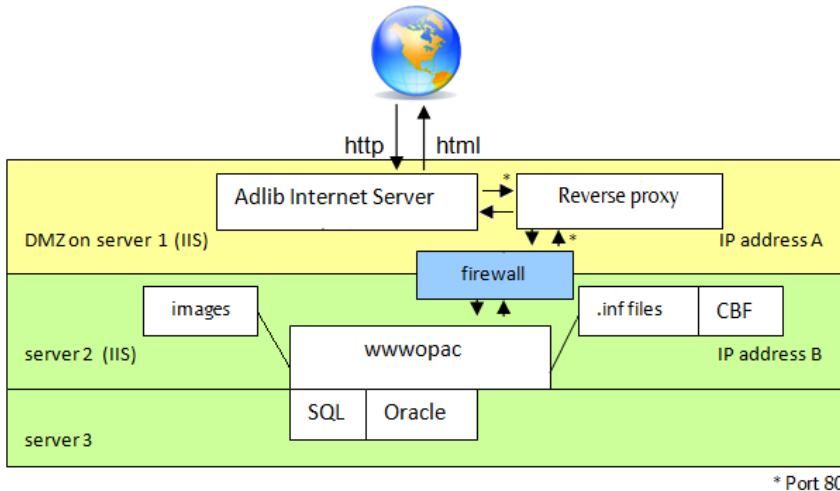
From:
Example: contentserver1

To:
Example: www.contoso.com

OK Cancel

Keep the *Enable SSL Offloading* option marked, and in the *Outbound rules* mark the *Rewrite the domain names of the links in HTTP responses* option. In the *From* field, also enter the destination server name, and in *To*, enter the DMZ server name (the host name of the reverse proxy site). Click *OK* to store these rules. The default settings for the added rules are fine for use with the Adlib webopac, so the setup of the reverse proxy itself is now done.

11. Now your Adlib Internet Server web application has to refer to the host name of the reverse proxy, in the *globalsettings* file, so that the reverse proxy can forward http requests to the webopac in the internal domain. This way you only need to allow http traffic through port 80.



Note that you don't need an Adlib Internet Server web application per se to make good use of a reverse proxy: direct http traffic to and from *wwwopac.ashx* (the Adlib API), without an Adlib Internet Server web application, can also be protected this way.

As an alternative to the option above you could also place a copy of *wwwopac* and accompanying files (amongst which a second, sufficiently configured *adlibweb.xml* file) and the images themselves on server 1. This will allow the web application to retrieve images from server 1, and database records from server 2. However, this alternative is not preferred because the images have to reside redundantly on two servers at the same time.

4 The installation procedure

As mentioned, MSXML4, IIS, .NET 4.0 and MVC 4.0 have to be installed first.

4.1 Step 1: create separate folders

When you spread the different components of an Adlib Internet Server system over different servers, as recommended in the previous chapter, this automatically creates the desired situation in which those components are located in different folders.

If you are placing your Adlib Internet Server web application, including the webopac, together on one server anyway, then it is recommended to have two different folders for the Internet Server web application and the webopac with its accompanying files, and have these folders at the same level in the directory structure, not one folder nested inside the other. By default, the folder for the Internet Server web application is named `\internetserver5` and the folder for the webopac `\wwwopacx`.

In all cases, the different folders should have their own virtual directory/application and their own separate application pool, so that a disrupted process in one place doesn't disrupt processes elsewhere.

When you copy folders from a cd to your computer, you will still have to remove the *Read only* marking: in Windows Explorer, right-click the folder to which you have just copied all files, choose *Properties* in the pop-up menu and unmark the *Read-only* option. Click *Apply* and choose the *Apply changes to this folder, subfolders and files* option, in the window that appears, before clicking *OK*.

Also note that under Windows Server 2008 the subfolder which holds the executables, must not be named `\bin`.

4.2 Step 2: IIS 7 setup under Windows Server 2008

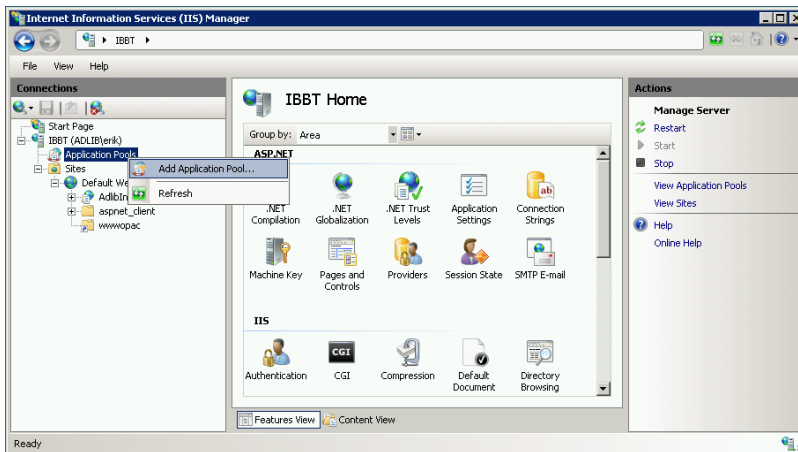
For *wwwopac.ashx*, the Internet Server and/or *oai.ashx* each, in IIS a so-called *application* must be created, to secure your server and to create an internet address. We recommend to run each created application under its own application pool to keep the processes of the web services separated at all times.

■ Application pools

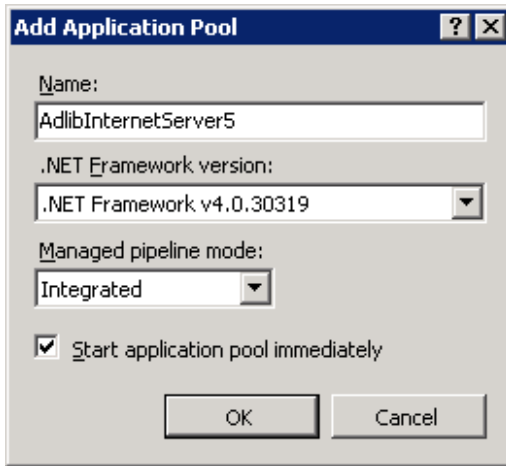
We start by configuring new *application pools*. You should create at least one new pool.

1. Start the *Internet Information Services (IIS) Manager* by clicking *Start > All programs > Administrative tools > Internet Information Services (IIS) manager*.

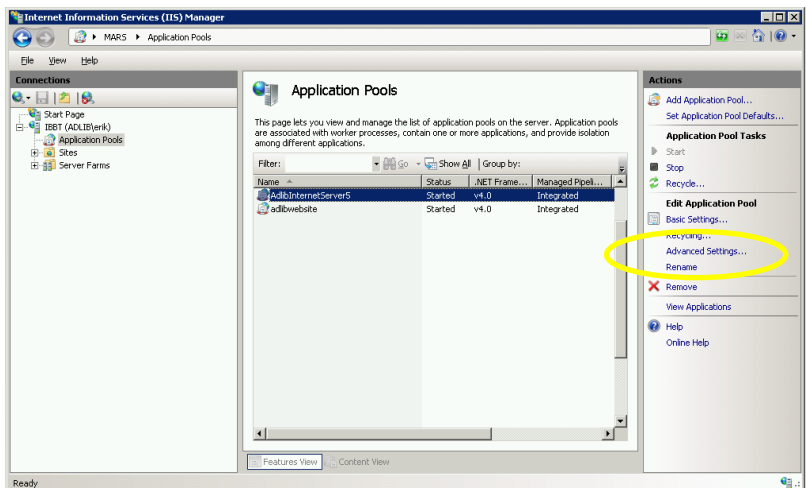
In the left window pane of the *Internet Information Services (IIS) Manager*, right-click *Application pools*, and choose *Add Application Pool* in the pop-up menu which opens.



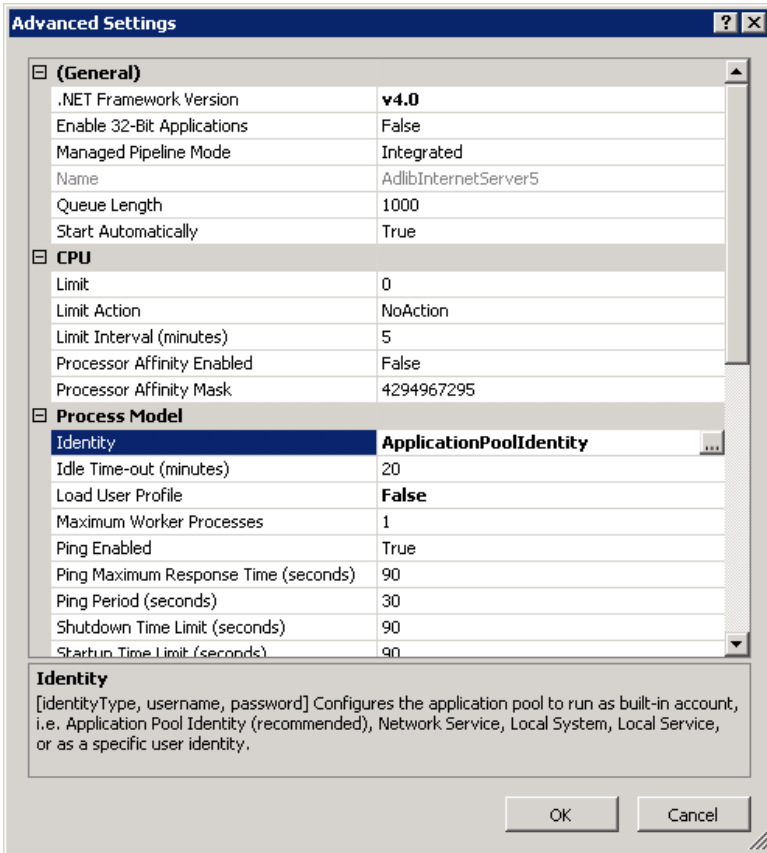
2. The *Add Application Pool* window opens. Fill it in like below. You may choose a different name for the application pool though, if you want (note that no spaces are allowed in the name). When you create several application pools, you can name them according to the application you will be running in them, like `internetserver5pool`, `wwwopacpool`, `adserverpool` or `oaiserverpool`. (Note that an application pool for *adserver.exe* doesn't require a .NET version: for *adserver.exe* you can select *No Managed Code* instead.) Click *OK*.



3. In the IIS manager, select the *Application Pools* node, if that is not the case yet. In the *Application Pools* list in the middle window pane, you'll see your new application pool(s). Select the new application pool and click the *Advanced settings* option underneath *Actions* in the right window pane.



4. The *Advanced Settings* window opens. Click *Identity* underneath *Process Model* and then click the ... button in the entry field next to it which currently reads *ApplicationPoolIdentity*.



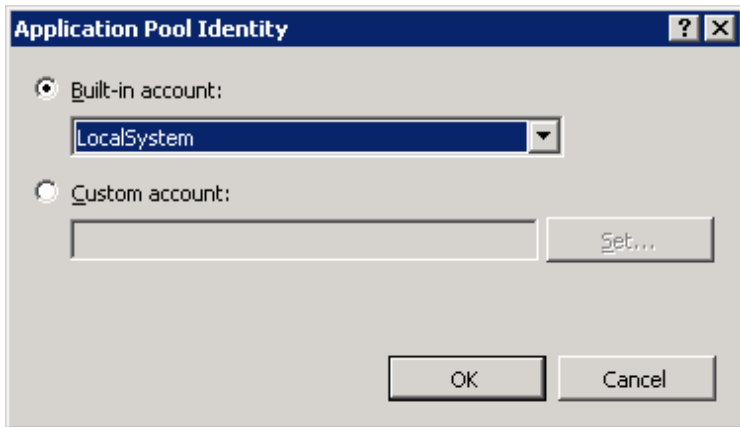
5. The *Application Pool Identity* window has opened. Set the account to use.

In most cases, your Internet Server, *wwwopac.ashx* web service, *\data* folder and SQL Server won't be located on the same physical server. You'll then have to set the *Application Pool Identity* to a *Custom account*. This should be the Active Directory account under which anonymous internet users will be able to access web services in this application pool. It is very well possible that such an account already exists within your network. If not, you'll first have to create one before you get this application pool in order. Subsequently use the *Set* button to enter the name (preceded by your domain and a backslash) and the password of the account to apply.

Further, this account must have at least read-only access rights on the database in SQL Server and possibly write access as well if users must be able to save tags and comments in the database or if they must be able to make reservations. (See chapter 5.3 for more information about using such an account.)

Only if your Internet Server, *wwwopac.ashx* web service, *\data* folder and SQL Server are located on one and the same physical server, set the *Built-in account* to *LocalSystem*.

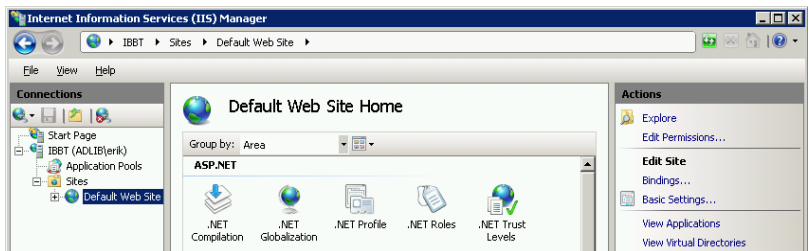
Click *OK*. Do this for each of your new application pools.



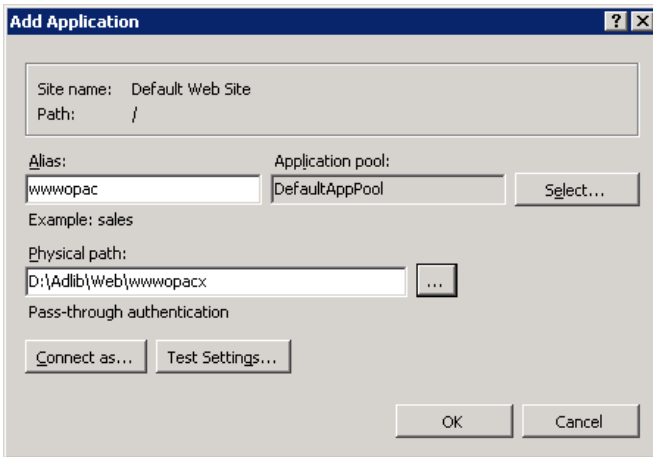
■ Adding the web service as an IIS application

The *wwwopac.ashx* or *oai.ashx* web service setup proceeds as follows:

6. In the *Internet Information Services (IIS) Manager*, open the *Sites* node and right-click the site in which you want to accommodate your *wwwopac.ashx* or *oai.ashx* web service, the *Default Web Site* for instance.



7. In the pop-up menu that opens, choose *Add Application*, after which the *Add Application* window opens. Then, first enter the desired *Alias* for the application, for example `AdlibWebService` or `Adlibwwwopac`; choose a clear, descriptive name. Then select the path to the physical folder on your system in which `wwwopac.ashx` (or `oai.ashx`) and its accompanying sub-folders and files can be found. The syntax of the URL for calling the Adlib web service becomes:
`http://<webserver>/<applic_alias>/wwwopac.ashx`
 or `http://<webserver>/<applic_alias>/oai.ashx`

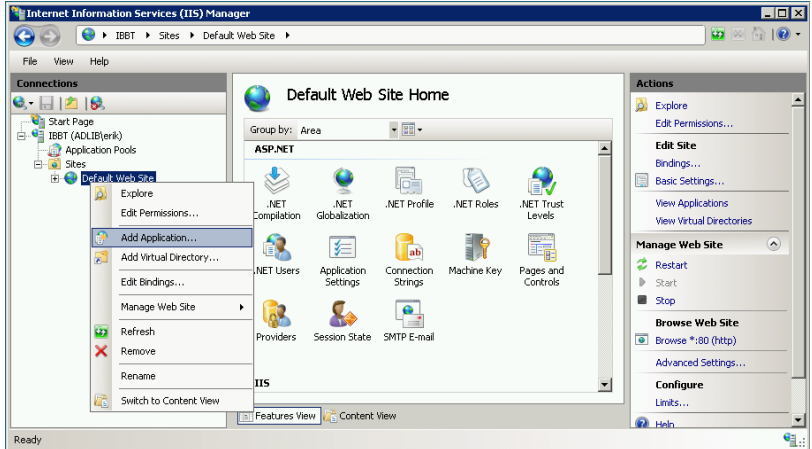


8. Click the *Select* button to set the application pool for this application. Select the `.NET 4.0` application pool that you created for this web service earlier. Click *OK*.
9. Also click *OK* in the *Add application* window to create the application. In the *Internet Information Services (IIS) Manager*, you'll then observe that the new application has been added to your website. Your `wwwopac.ashx` or `oai.ashx` setup in IIS is finished now.

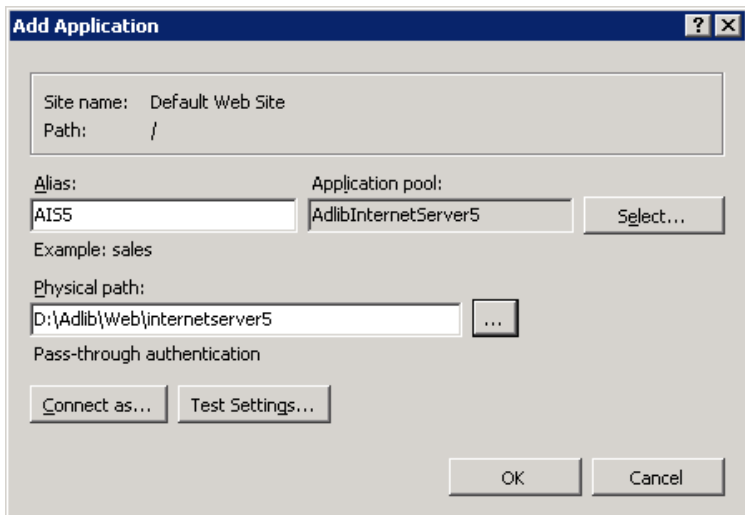
■ Adding the Internet Server as an IIS application

And finally, we have to define the Internet Server web application itself as an IIS application too:

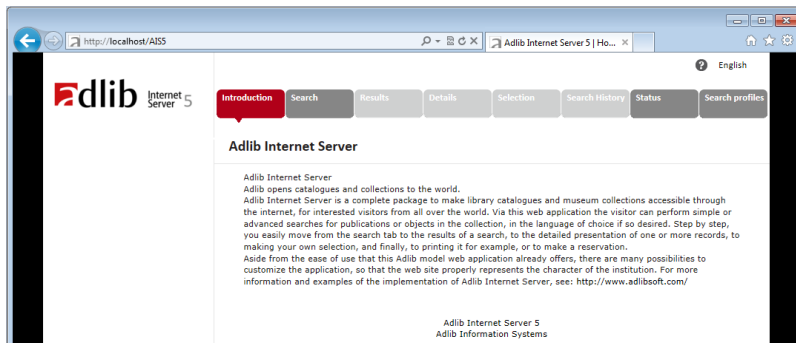
10. Right-click the website to which you want to add the Internet Server, in the example below the *Default Web Site*, and choose *Add Application* in the pop-up menu which opens.



11. Enter an *Alias* for the application, like in the figure below (do not include spaces in the name). The alias will be part of the URL to the website.
 Click the *Select* button, set the *Application pool* for this application to the pool that you created for it earlier and click *OK*.
 In *Physical path* you should now enter the path to the folder which holds your Internet Server folders and files. Click the ... button next to this entry field to be able to look up the relevant folder on your system.



12. The Internet Server folder is now visible as an application (AIS5 in our example) underneath the chosen website in the left window pane. Right-click that application and choose *Manage Application* > *Browse* in the pop-up menu to see if the new application will start up. Your Internet Server webapplication now opens in your default internet browser; you cannot work with it yet, because the application hasn't been configured yet. In our example, the result is as follows:



500 – Internal server error: asperrorpath

If you decided to make the *wwwopac.ashx* IIS application a sub application of the Internet Server IIS application, maybe because it helps you keep track of which *wwwopac.ashx* version is associated with which Internet Server version, you may get an internal server error when you try to access the *wwwopac.ashx* application in your browser: "There is a problem with the resource you are looking for, and it cannot be displayed." A part of the URL in the browser's URL box reads *.../error?asperrorpath=...*, pointing to the *wwwopacx* folder. You can solve this problem by enclosing the `<system.web>`, `<system.net>`, `<appSettings>` and `<system.webServer>` settings in the *Web.config* file of your Adlib Internet Server in a `<location path="." inheritInChildApplications="false"></location>` node.


```

<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <configSections>
    <sectionGroup name="elmah">
      <section name="security" requirePermission="false" t
      <section name="errorLog" requirePermission="false" t
      <section name="errorMail" requirePermission="false"
      <section name="errorFilter" requirePermission="false"
    </sectionGroup>
  </configSections>

  <location path="." inheritInChildApplications="false">
    <system.web>...</system.web>
    <system.net>...</system.net>
    <appSettings>...</appSettings>
    <system.webServer>...</system.webServer>
  </location>

```

4.3 Step 3: install Adserver

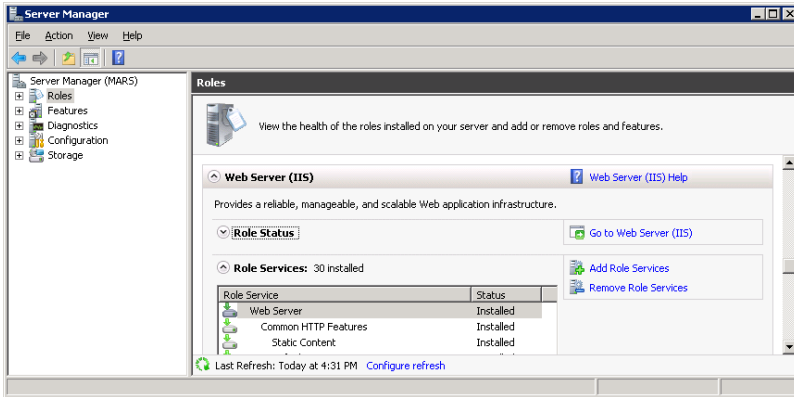
Adserver must be installed only if the issuing and reserving of (library) materials via your Adlib Internet Server must be made possible. If that is not the case or if you've only installed the OAI server, then Adserver is not available to you, so you can't install it either: then skip this chapter.

The installation proceeds as follows:

1. Only if *adserver.exe* has not been installed yet: in Windows Explorer search the `\wwwopacx` folder you installed earlier (or the folder with the name you chose yourself), and open it. Now copy the files *adserver.exe*, *adserver.dll*, *adserverInstall.bat*, *adlib.lic* and *adliblic.dll* to a new folder next to the `\wwwopacx` folder; name the new folder `\adserver` or `\wwwopac`, for example. Strictly speaking there's no requirement for the separate folder, but it keeps things organized.
2. In the new folder, right-click the *adserverInstall.bat* file and choose *Edit*. Behind *workdir*, change the path to the `\library loans management` or `\wadcirc` directory on your hard disk. Save the file and close it.
3. Then install *adserver.exe* by double-clicking the *adserverInstall.bat* file. (The path to *adcirc.pbk* in the `\library loans management` folder will be added to the Window registry.)

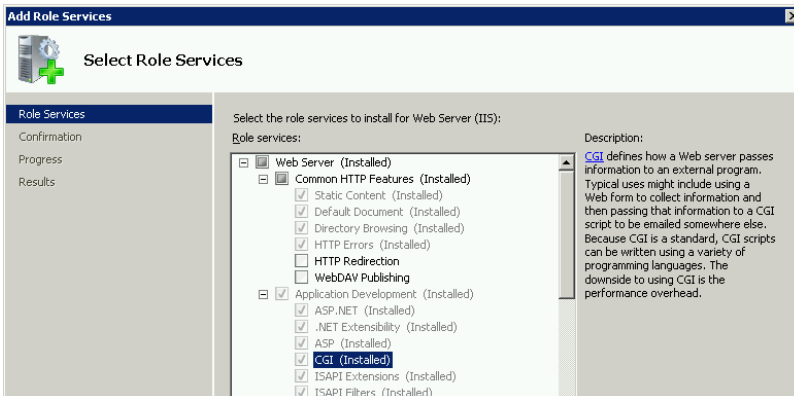
Subsequently check whether the CGI role service for the web server has already been installed:

- Open the *Server manager* first (not IIS), by clicking *Start > All Programs > Administrative Tools > Server Manager*. In the left window pane, click the *Roles* node and then click the *Add role Services* option right next to the *Role services* header in the **Web Server (IIS)** section in the right window pane (fold in sections or scroll down if you do not see the section right away).



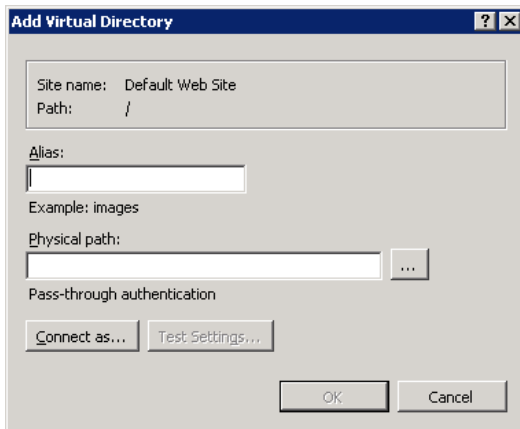
The *Add Role Services* window opens. In it, select the *Role services* page and mark the *CGI* checkbox if it isn't selected already*. Click *Next* and then click *Install*.

* If CGI has been installed already, the *CGI* checkbox will be displayed checked and greyed out. Then close the *Server Manager* via *Cancel*.



Finally, a virtual folder and a handler mapping have to be created for *adserver.exe*. (Earlier you may have created a separate application pool already.)

5. Start the *Internet Information Services (IIS) Manager* by clicking *Start > All programs > Administrative tools > Internet Information Services (IIS) Manager* and open the *Sites* node. Then right-click the site in which you want to accommodate your Adserver web service, the *Default Web Site* for instance. In the pop-up menu that opens, choose *Add virtual directory*, after which the *Add Virtual Directory* window opens.

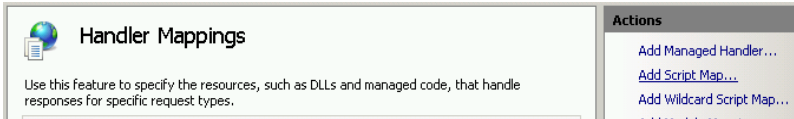


6. First, type a clear and descriptive name for the virtual folder in the *Alias* field, *Adserver* for example. Then select the path to the physical folder on your system in which *adserver.exe* can be found. Then click *OK* to create the virtual directory. The syntax of the URL for calling the Adserver web service becomes:
`http://<webserver>/<applic_alias>/adserver.exe`
7. Click the *Connect as* button and select *Application user* in the dialog which opens. Click *OK* in both windows to create the virtual directory.
8. Underneath the *Default Web Site*, your new virtual folder now appears. Right-click it and choose *Convert to application* in the pop-up menu. The *Add application* window opens, with virtual directory details already filled in. Click the *Select* button to set the base application pool. Preferably select your earlier created application pool for the Adserver. Click *OK* in both windows to create the application.

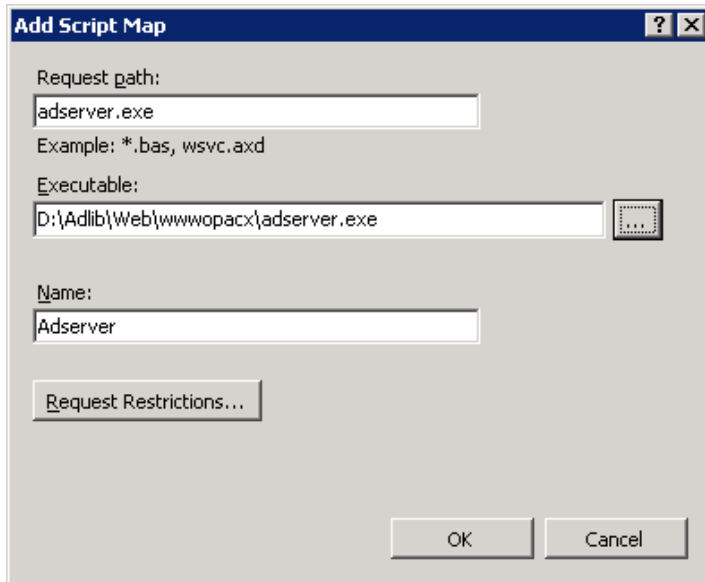
9. To make sure that the *adserver.exe* will be used as a CGI handler and won't be downloaded as a file, you'll have to create a so-called handler mapping in IIS. Select your virtual directory and double-click the *Handler Mappings* icon in the middle window pane.



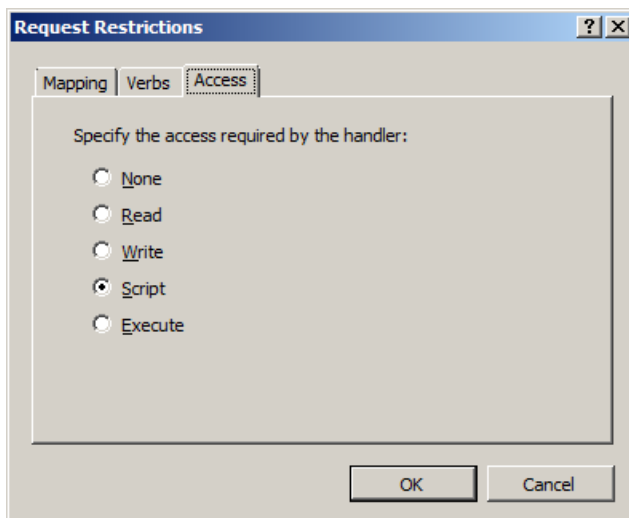
When the *Handler Mappings* page is visible, click the *Add Script Map* option underneath *Actions* in the right window pane.



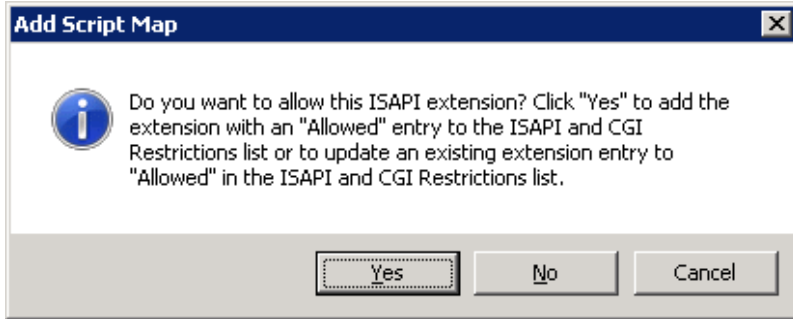
10. The *Add Script Map* window opens. In the first entry field, enter *adserver.exe* and in the second field enter the path to it, for example like in the figure below. In the last entry field, enter a name of your choice for this executable reference; this name identifies the handler mapping in the mapping list in the IIS Manager main window. Make sure that the name of the executable in the first two entry fields both have been entered in lower case, like below, or both in capitals (otherwise you'll get an error message).



Click the *Request restrictions* button to check if the proper restrictions have been set: this is usually the case, by default. No settings have to be marked on the first two tabs, and on the third at least *Script* access must have been selected. Click *OK* in both windows.



In the *Add Script Map* message which then appears, click Yes.



11. The Script Map for *adserver.exe* has now been created and is visible in the *Handler Mappings* list.

4.4 Step 4: modify *adlibweb.xml*

The *adlibweb.xml* file serves to initialize the *wwwopac* or *OAI* server. In here you must at least enter a `<databasepath>` and a `<database>` for *wwwopac*.

- See <http://api.adlibsoft.com/site/documentation> for more information about configuring *wwwopac.ashx* through *adlibweb.xml*.
- The *oai.ashx* can use its own *adlibweb.xml* file, completely separate from the *adlibweb.xml* for *wwwopac*. See chapter 7 for information about setting up *adlibweb.xml* for *oai.ashx*.

4.5 Step 5: modify *globalsettings.xml*

Contrary to *adlibweb.xml*, in which you provide *wwwopac* settings necessary for searching and for the search results to generate, in an XML document named *globalsettings.xml* in the `\Config` subfolder of your new *internetserver5*-directory you make general settings for the Internet Server web application (not for *OAI* Server).

The Internet Server web application has to be told first which information comes from which virtual folders. That is why, amongst others, the path to *wwwopac.ashx* must be set in *globalsettings.xml*.

4.5.1 <dataservers>

The path to the virtual folder in *globalsettings.xml* is specified in a code fragment that should look something like this:

```
<dataservers>
  <server database="*">
    <url>
      http://www.ourmuseum.com/wwwopacx/wwwopac.ashx
    </url>
    <imageHandlerUrl>
      http://www.ourmuseum.com/wwwopacx/wwwopac.ashx
    </imageHandlerUrl>
  </server>
  <server database="test">
    <adlibDatabase>collecttest</adlibDatabase>
    <url>
      http://www.ourmuseum.com/test/wwwopacx/wwwopac.ashx
    </url>
    <imageHandlerUrl>
      http://www.ourmuseum.com/test/wwwopacx/wwwopac.ashx
    </imageHandlerUrl>
  </server>
</dataservers>
```

- If all databases must be approached through the same server, then you only need to specify the `<server database="*">` node.
- If the Internet Server web application must also be able to access one or more databases which must be approached through another server, then specify that server separately, as has been done in the example above.
- If the web application tries to access a database which hasn't been listed here, then the server specified in the `<server database="*">` node will be used by default.
- The `<adlibDatabase>` node is not mandatory, but points to the alias of an Adlib database definition (an *.inf* file), as specified in the *adlibweb.xml* file of the relevant server; in that case, the value to be entered behind the `database` attribute of the `server` node can be any unique identifier, although it's often best to give it the same name as the database alias.
If you do not use the `<adlibDatabase>` node, the web application will assume that the value entered behind the `database` attribute of the `server` node is actually a reference to such an alias.
In the `<server database="*">` node you must never enter an `<adlibDatabase>`.

- Behind `<url>`, enter the (http) URL to the webopac in the virtual folder on your system. `<imageHandlerUrl>` must contain the URL to your `wwwopac.ashx` image server.

4.5.2 `<emailservers>`

Next, you must specify an SMTP e-mail server. The `id` attribute of the `server` node can be given the value `"*"`. This server will be used by the web application to send e-mails concerning the editorial approval of comments and tags added by users, should that functionality be active in your web application. The relevant e-mail address must be specified in the section for that functionality. In the standard Internet Server you can only specify a single server and you don't need to provide any general contact details for e-mailing because the web application doesn't offer contact links.

```
<emailservers>
  <server id="*">
    <smtp>smtp.ourmuseum.com</smtp>
  </server>
</emailservers>
```

Non-standard e-mail options

Although the current standard Internet Server only requires a single SMTP e-mail server and doesn't support e-mail contacts, you can specify one or more of your SMTP e-mail servers underneath `<emailservers>`, in principle. Often there will be only one, which means that the `id` attribute of the `server` node can be given the value `"*"`. If your customized web application must display contact links, you can list them underneath `<contacts>`. Then, you don't need to refer to that e-mail server explicitly underneath `<contacts>`: the server identified with `id="*"` will automatically be used. If you do want to use multiple SMTP e-mail servers, then that is possible as well: assign a unique id to each server and use it to refer to the relevant server per e-mail contact that you define next.

```
<emailservers>
  <server id="*">
    <smtp>smtp.ourmuseum.com</smtp>
  </server>
</emailservers>
```



```

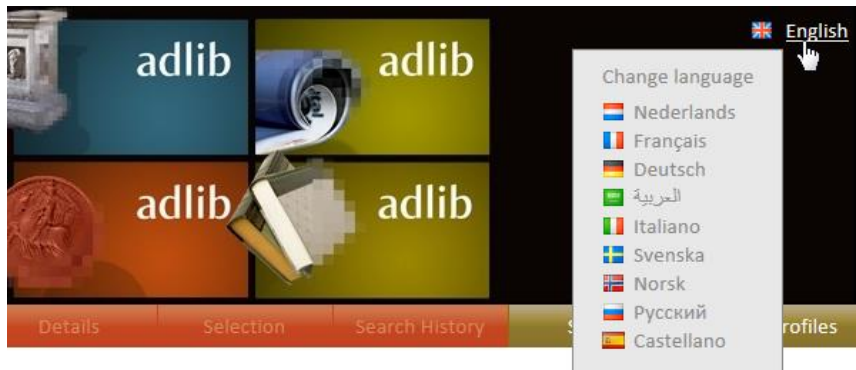
<contacts>
  <contact id="Admin">
    <name>Administrator</name>
    <email>
      <address>john@ourmuseum.com</address>
      <serverId>*</serverId>
    </email>
  </contact>
  <contact id="NoReply">
    <name>Adlib Internet Server</name>
    <email>
      <address>noreply@ourmuseum.com</address>
    </email>
  </contact>
</contacts>

```

Most other settings in *globalsettings.xml* do not have to be changed if you installed your Adlib system by the defaults. Depending on the deviations in your installation, you do have to adjust the relevant settings. Below, you'll find some more examples. Note that in *globalsettings.xml* itself most settings are clarified briefly, helping you to make correct adjustments.

4.5.3 <languages>

This list specifies in which languages the interface of the Internet Server web application will be available: of course, the relevant translations of labels and such, must actually be present in *userlanguage.xml* and *userHelp.xml* (underneath *\internetserver5\ Config*).



Per language you provide a standard language code and the image of a flag representing the language. The *\flags* subfolder can be found

underneath `\internetserver5\Content\Images`. Whether the images will actually be displayed in the user interface depends on the active theme that controls the design. The theme can be set via the `<defaultTheme>`-option (see further down in this chapter).

```
<languages>
  <language culture="en-GB" iconImage="flags/flag_great_britain.png"
    default="true" dirCssClass="ais-ltr" />
  <language culture="nl-NL" iconImage="flags/flag_netherlands.png"
    dirCssClass="ais-ltr" />
</languages>
```

One of the languages can be set as the start language, via the `default="true"` attribute.

4.5.4 <documents>

Let's view the `<documents>` group, for example:

```
<documents>
  <file type="Reproduction/reproduction.reference">
    <item>
      <source>../images</source>
      <destination></destination>
    </item>
  </file>
</documents>
```

Here, you can indicate per database field which strings entered in that field must be replaced, for the Internet Server to be able to retrieve images from the correct location if not all images reside in one folder together. The "translated" reference will be added to the field as the `weburl` attribute, in the XML of the search result. This happens on-the-fly when retrieving the relevant image file; no changes are made to the original data in the database. However, the attribute is not being used in the standard Internet Server 5 – this means that AIS 5 expects all images to be located in the same folder indeed – but it is possible to adjust the image retrieving stylesheets in such a way that not the field content itself but the `weburl` attribute content of the field will be used.

If in `globalsettings.xml` a `<documents>` section has been defined like in the example above, then all `reproduction.reference` field nodes (in the `Reproduction` field group) will get the `weburl` attribute. For the example above this means that if in the field contents a partial string `../images` appears, the `weburl` will become equal to the field data in which the replacement has taken place. If the replacement is not ap-

pllicable, because the searched string is not present, the `weburl` will become equal to the field data.

An example of the `reproduction.reference` node in the resulting XML after the replacement, is the following:

```
<Reproduction>
  <reproduction.reference weburl="/2012/M53.jpg">
    ../images/2012/M53.jpg
  </reproduction.reference>
</Reproduction>
```

This example would be realistic if you had stored your images in sub-folders per year underneath the `\images` folder. In such case you must not only adjust the `<documents>` section but several stylesheets as well because they currently use the field contents by default, instead of the `weburl`. Moreover, you need to take care not to call the `"ais-strip-image-path"` template for the `weburl`, because that template removes any path preceding a file name. Such an adjustment can be done as follows:

In the stylesheets for the list (*brief...*) and detailed display (*detail...*) underneath `..\internetserver5\Views\Results`, you will currently find templates comparable to the following:

```
<xsl:template match="Reproduction/reproduction.reference">
  <xsl:variable name="strippedImage">
    <xsl:call-template name="ais-strip-image-path">
      <xsl:with-param name="image" select="."/ >
    </xsl:call-template>
  </xsl:variable>
  
</xsl:template>
```

So instead of the field contents we would like to use the contents of the `weburl` attribute, without removing any path in front of the file name. The solution is to simply adjust this template to the following:

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

If your images are located in a single folder (whatever the name of the folder), the `<documents>` section in *globalsettings.xml* can be removed altogether: the `weburl` attribute then won't be added to the

field. Or you can leave the section as it is. In principle you don't need to adjust the `<documents>` section, even if currently an undesirable path replacement is specified, because the standard stylesheets remove any path in front of a file name anyway. The current structure of the Internet Server is aimed to use only the file name, in combination with the path to the images folder set in the `<imageServerConfiguration>` section in *adlibweb.xml*.

As far as images are concerned, the default settings in *globalsettings.xml* only work if the image reference in records is either just a file name (as is the case in model applications 4.2) or a file name plus a preceding path and all images are located in the same folder: in the first case the missing path in the reference won't cause problems whilst in the second case any path in the reference will be stripped from the search result. In the default case, a single image server will do.

4.5.5 `<session>`

The `<session>` group contains settings which are activated again per user session. In the example below you can see a selection of elements from this group. The elements themselves are sometimes subdivided into groups, so that it's easy to see which settings belong together:

```
<session>
  <xmlType>grouped</xmlType>
  <display mode="brief" navMode = "ReadOnly" />
  <display mode="detail" navMode = "ReadOnly" />
  <debug>>false</debug>
  <limit>20</limit>
  <highlight>>true</highlight>
  <scanLimit>20</scanLimit>
  <autocompleteLimit>10</autocompleteLimit>
  <selectionLimit>100</selectionLimit>
  <defaultLogOnRedirectRoute></defaultLogOnRedirectRoute>
  <imageView
    zoomMultiplier="1.25"
    minSize="2"
    defaultSize = "500"
  />
  <defaultTheme>ais5</defaultTheme>
  <indexRedirectsTo></indexRedirectsTo>
  <authentication enabled="true" source="loginserver" />
  <authorization enabled="true">...</authorization>
  <borrowing>...</borrowing>
  <reserve enabled="true">...</reserve>
  <sdiProfiles enabled="true">...</sdiProfiles>
  <comments database="comments" enabled="true"
    writeAllowed="true" approveBeforeDisplay="false"
    addFormExpanded="false">...</comments>
```

```

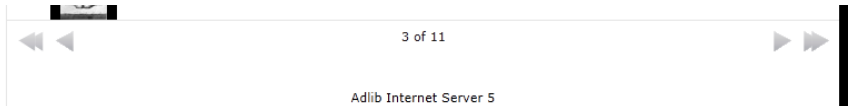
<tagging database="tagging" enabled="true"
  writeAllowed="true" approveBeforeDisplay="false"
  addFormExpanded="false">...</tagging>
<externalDatabases>...</externalDatabases>
</session>

```

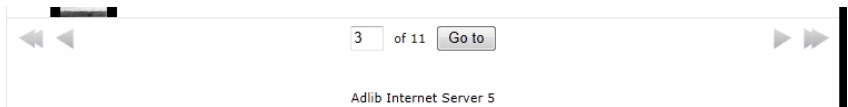
Explication of elements:

xmlType [grouped|structured|unstructured|raw]: the stylesheets of the standard Internet Server 5 expect the *wwwopac.ashx* search result to be delivered as grouped XML, as specified in *adlibweb.xml* by default, in the `<globalConfiguration>`. The other XML formats are available for the sake of completeness, but you cannot use them in the standard Internet Server.

display attributes: **mode**=["brief"|"detail"] **navMode**=["ReadOnly"|"Short"|"Long"] **visiblePages**=["#"]: at the bottom of the *Results* tab and directly underneath the tabs row on the *Details* tab, a so-called navigation bar appears in which you can see which record or results page you are currently viewing and arrows to browse to the previous or next record or page. That navigation bar can be presented in three different ways. The `ReadOnly` variety displays buttons like in the screenshot below and a record/page numbering that won't allow you to enter a different number manually.

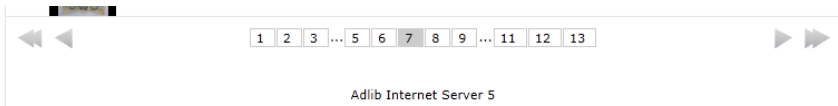


The `Short` variety displays buttons like in the screenshot below and a record/page numbering that does allow you to enter a different number manually after which you can browse to the relevant page directly by clicking the *Go to* button.



The `Long` variety displays buttons like in the screenshot below and a sequence of some available record or page numbers. The sequence is limited by the `visiblePages` attribute. With this attribute you specify the amount of number tiles which must be displayed next to number tile 1, next to the last number and on both sides of the currently selected number tile. `visiblePages="2"` for example, leads to the presentation as shown below. By clicking the desired number tile you

jump to the relevant page immediately. It is mandatory to use `navMode="Long"` together with the `visiblePages` attribute.



debug [`true|false`]: switches debug mode on or off; use this mode during development and testing only. This mode displays extra information on the Results and Details tabs, like the search duration, the used stylesheet, the `wwwopac.ashx` query, the XML produced by `wwwopac.ashx`, etc.

In the standard Internet Server 5 this function has currently (November 2012) not been implemented yet.

limit: this is the maximum number of records that `wwwopac.ashx` retrieves per request and displays on screen.

highlight [`true|false`]: switches the highlight function of the Internet Server on or off. If highlighting is on, the search key will be colour marked in the search results. The example below shows a search on "magic".



The `highlight` setting is present in both `globalsettings.xml` as well as `adlibweb.xml`. There, the `<highlight>true</highlight>` setting always has priority. This means that if you want to switch highlighting off, `<highlight>>false</highlight>` must be set in both files.

The colour in which the search key will be marked can be changed, if you wish. With the `<defaultTheme>` setting (see further down in this `globalsettings.xml` overview), a different design layout can be applied to the Internet Server web application. The relevant files for the selected theme can be found in the subfolder with the same name underneath `\internetserver5\Content\Css\Themes\`. In that subfolder you'll then find a `base.css` stylesheet in which, amongst others, the `highlight` style for this theme has been defined. Here you can adjust the properties of the highlight marking easily. The colour value must be specified in hexadecimal notation; there are websites on the internet where you can choose a colour from a colour picker after which the hex colour value is displayed. To change the standard olive green

marking into e.g. dark yellow, change the colour value from #877404 into #FFE500.

```
.highlight
{
  font-weight: bold;
  color: #FFE500
}
```

scanLimit: this is the maximum number of terms that will be retrieved and displayed if the user clicks a *List* button in the search form for advanced or expert searching.



autocompleteLimit: this is the maximum number of terms that *wwwopac.ashx* retrieves per entered letter and displays in the autocomplete drop-down list that is available for some fields during entry.

When the user enters a value into a field, the autocomplete function will automatically open a drop-down list containing existing values starting with the entered characters. The contents of the drop-down list changes during typing. This way the user might not have to type the whole word, by simply picking the desired value from the list. In *formsettings.xml*, underneath *\internetserver5\Config*, the autocomplete function can be switched on or off (default) per search field on the three search forms (*Simple, Advanced, Expert*).

selectionLimit: use this option to set the maximum number of records that can be selected by the user in the search result by marking the checkboxes in front of them. You can use this setting to limit the amount of record data which can be e-mailed, printed or downloaded all at once.

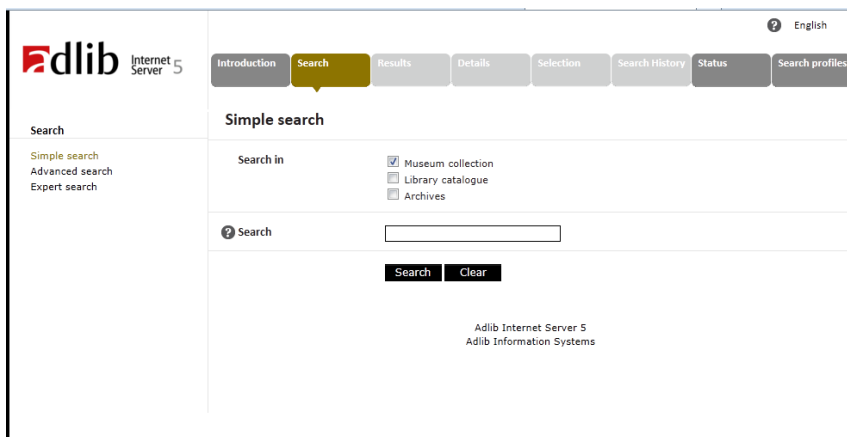
defaultLogOnRedirectRoute: may contain an identifier which refers to a different screen tab of the Internet Server, to have it automatically opened after a user has logged in successfully. Currently there's no accessible list of identifiers available.

imageViewer: when the user clicks a linked image in the detailed display of a record, it will be opened enlarged in a separate window. The initial height of the displayed image can be set with the `imageView defaultSize` attribute, in pixels. The resize factor with which the user can scale the images, must be set with the `zoomMultiplier` attribute. And finally, the smallest image height you want the image viewer to be able to display, can be set with `minSize`, also in pixels.

```
<imageView
  zoomMultiplier="1.25"
  minSize="200"
  defaultSize = "500"
/>
```

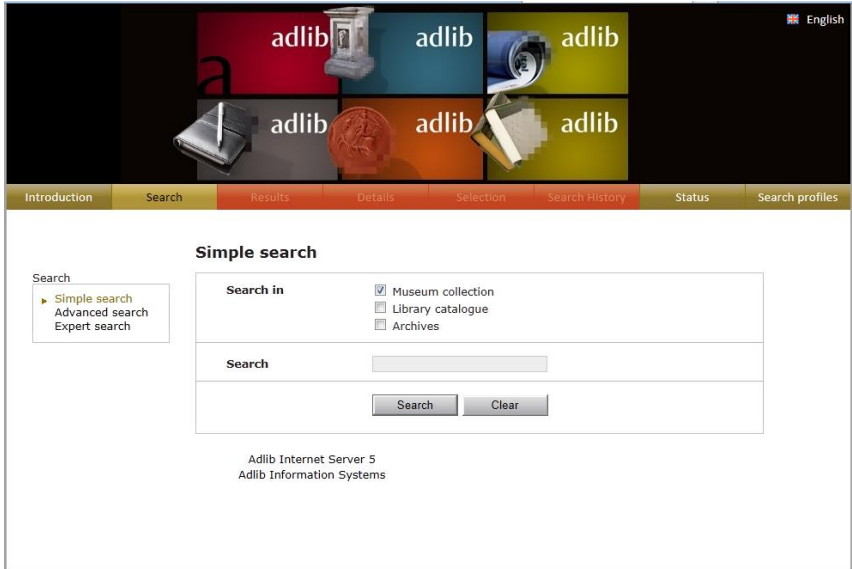
A `maxSize` attribute hasn't been implemented yet. It is important though to prevent images from being downloaded in a high resolution because such downloads have a negative effect on the performance of your website, they take up a lot of disk space and may lead to copyright infringement. So instead of `maxSize`, use the `maxWidth` and `maxHeight` image server options in `adlibweb.xml` (for `wwwopac.ashx` version 3.6.12310 or higher) and/or the possibility of your image server to overlay each image with a watermark, which can also be set in `adlibweb.xml`.

defaultTheme: Internet Server 5 can be presented in four different graphical designs: `AIS5`, `Basic`, `White` or `Mobile` (meant for display on mobile phones).

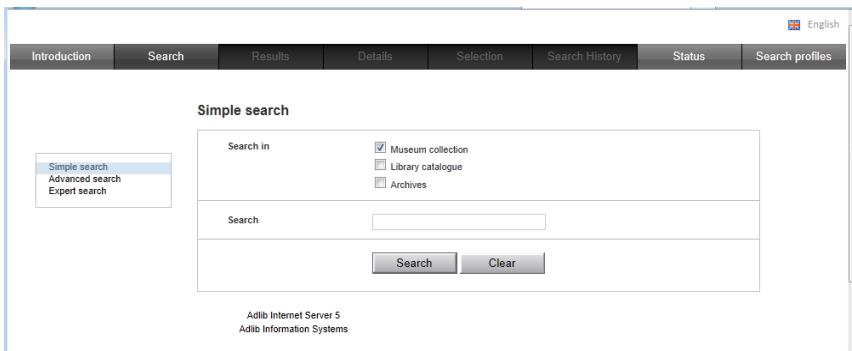


Theme: AIS5

These designs have largely been specified in CSS stylesheets. You can find them underneath `\internetserver5\Content\Css\Themes\`. You are free to use one of these designs as the starting point for a design of your own. Then copy the desired `\Theme` subfolder, assign it a different name and customize the stylesheets in it.



Thema: Basic



Thema: White

So, on all devices Internet Server 5 will be displayed according to the theme you set here. It is not able to automatically switch theme depending on the screen size of the used device. If you do want your

website to be presented in the `Mobile` theme on cell phone screens while one of the other three themes must be used for display on all larger screens, then the only efficient solution is to configure two Internet Server 5 websites, each with their own URL. The two web applications can be identical except for the `defaultTheme` setting and the URL. The link to the website for mobile phones can be made available on the home page of the normal website, for example.

Thema: Mobile

indexRedirectsTo: you can use this setting if you want your website to open with a page different from the default *Introduction* page. Preferably use relative URLs, like `~/search/simple`, `~/search/advanced`, `~/search/expert` (for any of the search forms) or `~/searchprofiles` (for the *Search profiles* tab) for example, etc. You can find the (partial) URL to use, in the *Address* bar of your web browser, while displaying the desired Internet Server page.

authentication: is meant to switch logging in on (`enabled="true"`) or off (`enabled="false"`). If logging in has been switched on, a user must log in with a user name and password before he or she can work with personalized sections of the website: to make reservations, add comments or to view their own borrower details, for example. In the current version of Internet Server 5, `enabled="false"` isn't actually effectuated: you are not allowed to remove the `<authentication enabled="true" source="collect" />` node or comment it out, but the parts of the interface in which the user can enter login

details cannot be switched off via this option.

In the `source` attribute refer to the `wwwopac.ashx` server that you want to use for authentication; use a `database` id as specified in the `<dataservers>` section of the current `globalsettings.xml` file. Which one you choose, is not that important. If you enter a non-existing identifier, the server with the `database="*"` identifier will automatically be used.

The other settings for this authentication functionality must all be set in `adlibweb.xml`, in the `<authenticationConfiguration>` section: see the [online documentation](#) about user authentication for more information.

Example:

```
<authentication enabled="true" source="collect" />
```

authorization: here you can set which Internet Server 5 functionality must be available to which user groups.

```
<authorization enabled="true">
  <allow feature="Comment" />
  <allow feature="Tag" />
  <allow groups="Administrator" feature="CommentApprove" />
  <allow groups="SDIUser, Employee" feature="SDI" />
</authorization>
```

The list of available functionality which is subject to authorization is limited to the following (of which only four types of functionality are currently actually validated):

- `Comment` – the ability to add comments to records;
- `CommentApprove` – approving comments and tags added to records;
- `Tag` – adding tags to records;
- `SDI` – all SDI functionality (having queries executed automatically on a regular basis);
- `Borrow` – the ability to borrow materials (currently – November 2012 – not active in Internet Server 5 yet);
- `Reserve` – the ability to reserve materials (currently – November 2012 – not active in Internet Server 5 yet).

The `<allow />` node must at least contain the `feature` attribute: the `groups` attribute is optional. When no `groups` are specified, all users including anonymous internet users can use that functionality. After logging in, the `groups` (groups or roles) the user is part of will be

returned by *wwwopac.ashx*. If in *adlibweb.xml* the `<authentication-Configuration>` used for this authentication contains a specification for a field from which the group name can be retrieved, and if one or more group names have actually been registered for the user, then those group names will be returned to the web application. Besides any specific group of a user, the (or all) default group name(s) will be returned as well, as specified in the `<defaultGroups>` section underneath the `<authenticationConfiguration>` in *adlibweb.xml*. This means that all logged-in users are automatically part of all default groups: see the [online documentation](#) about user authentication for more information about groups/roles.

So if you'd like to shield off certain functionality, it is important that its authorized users have a group name or role different from the default.

borrowing: is meant for future use, to be able to switch the ability to to borrow materials on or off and to send borrower details via e-mail. The option is not used by Internet Server 5.

```
<borrowing enabled="false">
  <!-- optional settings: firstNameField, lastNameField,
    departmentField, emailField, telephoneField -->
  <emailField>e-mail_address</emailField>
</borrowing>
```

4.5.6 <session><reserve>

With the options underneath `<reserve enabled="true">` you can set the possibilities for reserving materials. Users then need to log in on the *Status* tab of the website in order to be able to make reservations from the *Selection* tab. By the way: the reservation functionality makes use of *adserver.exe*, so that server must have been installed (see the [Complete guide to the Loans module](#)).

If you do not want to offer the possibility to make reservations, you simply set `enabled="false"`. You can leave the other settings as they are: the *Status* tab won't be present on the website anymore.

```
<reserve enabled="true">
  <adServerUrl>http://www.ourmuseum.com/wwwopac/adserver.exe
  </adServerUrl>
  <wwwopacUrl>http://www.ourmuseum.com/wwwopacx/wwwopac.ashx
  </wwwopacUrl>
  <adServerUserId>user1</adServerUserId>
  <adServerPassword>pw1</adServerPassword>
  <adServerLocation>main</adServerLocation>
  <adServerSite></adServerSite>
  <borrowerNumberField>borrower_number</borrowerNumberField>
  <emailField>e-mail_address</emailField>
  <notification>>false</notification>
```

```

<mailAddressFrom>noreply@ourmuseum.com</mailAddressFrom>
<mailAddressTo>sales@ourmuseum.com</mailAddressTo>
<reservationsDatabaseName>reserv</reservationsDatabaseName>
<copiesDatabaseName>copies</copiesDatabaseName>
</reserve>

```

adServerUrl: is the URL to the *adserver.exe* server. The server is used to make new reservations, to cancel existing reservations and to extend (renew) the loan period of already borrowed copies.

wwwopacUrl: is the URL to the *wwwopac.aspx* server that you want to use to have a list of reservations and borrowed copies retrieved for the current borrower, after logging in. (The borrower number will be used for the search.) This may very well be the general data server which you defined at the top of *globalsettings.xml*.

adServerUserid, adServerPassword: must contain the login data (login name and password) of a general user as it has been set up for the Adloan transactions application (in the *.pbk*). This data will be used by *adserver.exe* to log into the database via *adserver.dll*, to register a reservation (and also to be able to store sdi profiles). This is separate from global authentication, so you can't use this login name and password to log on to the website.

adServerLocation: must contain the name of the branch (location) where the reservations will be registered.

adServerSite: only needs to be specified in the special case in which an *adserver.exe* server hosts more than one web site and you have set a different workdir for each application. You can then find those settings in the Windows registry on the server where *adserver.exe* has been installed, below the base *Workdir* (in the path: *HKEY_LOCAL_MACHINE > SOFTWARE > ADLIB Information Systems > Adserver*). Those keys will begin with "Workdir", for example "Workdir2" or "WorkdirMuseum1". The value for the `<adServerSite>` option must then be the second part of the desired extra workdir key, so for instance `<adServerSite>2</adServerSite>` or `<adServer-Site>Museum1</adServerSite>`. If you do not provide an `<adServer-Site>`, the base workdir will be used.

borrowerNamefield: the field name in the user database, in which the full name of the user can be found, for display purposes. This option is no longer used in Internet Server 5.

borrowerNumberField: the field containing the borrower number with which the reservation can be assigned to the proper borrower. This usually pertains to the `borrower_number` field. Because of this, the field must appear in the user profile returned by *wwwopac.aspx*

(in XML format) after logging in. Therefor in *adlibweb.xml*, in the [authenticationConfiguration](#) section for the borrower database, the same field must have been set in the field mapping:

```
<map>
  <source>borrower_number</source>
  <destination>borrower_number</destination>
</map>
```

emailField: the field name in the borrower database, in which the e-mail address of the user can be found.

notification [*true|false*]: can be used to send the borrower a notification by e-mail after the reservation has been registered successfully. Currently (November 2012), this functionality hasn't been implemented in Internet Server 5 yet.

mailAddressFrom: the sender e-mail address for the mailing of the reservation notification, for instance your `noreply` or `info` e-mail address.

mailAddressTo: optionally, an e-mail address inside your institution to which the notification must also be sent, for the purpose of statistics or sales for example.

copyNumberField: is the field in the borrower database that holds copy numbers, for the internal processing of the reservation. This option is no longer used in Internet Server 5.

reservationsDatabaseName: is the name of the reservations database, as specified in *adlibweb.xml*. This database is only used when retrieving existing reservations for the currently logged in user. The existing reservations can be requested on the *Status* screen tab.

copiesDatabaseName: is the name of the title copies database, as specified in *adlibweb.xml*. This database is only used when retrieving details about copies borrowed by the currently logged in user. This information can be requested on the *Status* screen tab.

4.5.7 <session><sdiProfiles>

sdiProfiles [*true|false*]: switches the search profiles service on or off. With this service, the user can store a search as a search profile and have that search executed regularly and automatically. (Registered users do need to have write access rights to the database.) The results will be sent to the user by e-mail.

sdiUrl: is no longer used in Internet Server 5. In the `<dataservers>` section you specify the `wwwopac.ashx` server which handles the sdi functionality of Internet Server (meaning: the saving of updated pointer files). This doesn't need to be a separate server.

mailFormat: specifies whether the e-mail with the results of the search profile will be sent as `html` or `text`. Usually this will be `html`.

deliveryMethod [`email`]: specifies the way in which the results of the search profile must be sent. In Internet Server 5, `email` is the only possible value.

mailBodyFormat: provide the name of an adapl (without extension) or XSLT stylesheet (with extension) which will convert the XML result of the search profile to a readable text (HTML or plain text).

expiryInMonths: indicates the period in months over which a search profile will be executed.

emailField: the field name in the borrower database, in which the e-mail address of the user can be found. The search results will be sent to this e-mail address.

genericSearchProfilesOwner: must be the name of a role which has been defined in the application structure file of your Windows Adlib application (this is the `.pbk` file of e.g. Adlib Museum or Library). (See the [Adlib Designer Help](#) for information about specifying users in a `.pbk`.) In the Windows Adlib application, users with the proper permissions will be able to create pointer files that have this role as their owner. Pointer files with this owner will then be publicly accessible (readable) on the website. This is a good, automated way to offer ready-made and up-to-date lists of recent accessions or thematically sorted records, for example, to which visitors can subscribe.

sortBy: with this option you can have the search result of an sdi profile (in effect a pointer file) sorted on one or more properties of that pointer file. The properties (fields) you can choose from are: `Pfnumber`, `Owner`, `Title`, `Selectionstatement`, `Sortstatement`, `Printstatement`, `HitCount`, `Modification`, `Creation`, `LastRun`, `Expires`, `Caption`, `FacName`, `Frequency`, `Language`, `PruneMode`, `DeliveryMethod`, `MailFormat`, `Suspended`, `Limit`, `PrinterDestination`, `Subject`, `PfComment`, `RandomizeStatement`, `OutputFormat`. These field names are case-insensitive. Multiple field names must be separated by comma's. Behind each field you can further indicate if sorting must be applied ascending or descending.

```
<sortBy>title, pfnumber ascending</sortBy>
```

4.5.8 <session><comments> and <tagging>

These sections are meant to allow visitors of your website to add comments and tags to catalogue records. Although Internet Server 5 is prepared to handle this, the rest of your Adlib system might not be. See chapter 4.7 in this manual for information about comments and tags functionality.

4.5.9 <displaylist>

displayList: is a list of all databases available on the website. In here, the detail and brief presentation XSLT stylesheets to be used (to transform the XML search result to HTML) are selected in the `transformationFile` attribute: refer to the XSLT file without the file extension. The `type` attribute must point to database names as they have been specified in *adlibweb.xml*. The `label` attribute refers to an identifier of translations of options displayed in the website interface. You can find those identifiers and translations in *userLanguage.xml*.

Because multiple presentations are possible, multiple brief and detailed presentations can be included. With the `selected="true"` attribute you can specify which presentation will be offered first. On the website itself, a user may choose one of the other presentations specified here. For any one dataset only one `brief` display can be set to true and one `detail`. The others must be set to `false`. You can change these settings and possibly adjust the stylesheets themselves as well, if you know your way around XSLT. You could adjust a stylesheet to include other record data in the detailed display, for example. (Keep in mind though that fields can only be displayed when they have actually been retrieved by *wwwopac.ashx*: which fields are retrieved, is set in *adlibweb.xml* via the `<brieffields>` and `<detailfields>`.)

To the `display` attribute, any of the values `brief`, `detail` or `rss` can be assigned.

Further it is possible to submit extra information per database:

- With `<addToQuery>` you can specify a part of a search statement that must always be added (implicitly with a Boolean AND) to queries for that database. This way you can shield certain content on field level. `<addToQuery>` is an optional element.
- The `<downloadfields>` are no longer in use. Leave the option unchanged.

4.6 Some possibilities of formsettings.xml

In *formsettings.xml* in the `\Config` subfolder of your new *internetserv-er5*-directory you can specify in detail which databases the interface of Internet Server 5 allows to be searched, which search fields are available and which field data in detailed display can be clicked to continue searching on that value.

```
<module>
  <screens>
    <screen style="simple" isDefault="yes">
      <databasechoice name="all">
        <databasecheck name="collect" selected="yes" />
        ...
      <searchform>simple_database</searchform>
    </databasechoice>
  </screen>

  <screen style="advanced">
    <databasechoice name="fullCatalogue" group="library"
      onlyInternal="false" isDefault="yes">
      <searchform>library_advanced</searchform>
    </databasechoice>
    ...
  </screen>

  <screen style="expert">
    <databasechoice name="collect" group="museum">
      <searchform>museum_expert</searchform>
    </databasechoice>
    ...
  </screen>

  <screen style="detail">
    <databasechoice name="collect" group="museum">
      <searchform>museum_detail</searchform>
    </databasechoice>
    <databasechoice name="fullCatalogue" group="library">
      <searchform>library_detail</searchform>
    </databasechoice>
    ...
  </screen>
```

In the `<module>` node, for the three available search methods (*Simple search*, *Advanced search* and *Expert search*), you specify the method with which the website opens by default, which database are available per search method and which search form (`<searchform>`) must be used. Those search forms are defined in this *formsettings.xml* file too, at the bottom of the document. *Simple search* for example, uses the `simple_database` form definition by default. This definition is specified in `<formdefinition name="simple_database">`. (The `database-`

choice group and onlyInternal attributes no longer have a function.)

So you can offer more or fewer databases per search method. If you want to add an extra database, then that database must have been specified in *adlibweb.xml* and *globalsettings.xml* as well. Moreover, you must then check here in *formsettings.xml*, if the `<formdefinition>` that you want to use does have a `<tag>` section for the extra database: if not, you must still add it.

The `<screen style="detail">` is meant to be able to search on certain key data of a record in detailed display (like the author, creator, material, etc.) by clicking it. Each used `<searchform>` (museum_detail, library_detail and archive_detail) can be found at the bottom of *formsettings.xml* as a `<formdefinition>`. As far as the structure goes, these definitions are similar to the other form definitions, but the fields specified here won't appear in any search form: these are the fields that can be searched implicitly by clicking a value from such a field in the displayed record.

In a `<formdefinition>` you enter a list of fields (within the database for which this form is being used) on which the user must be able to search. Per `<searchrow>` you specify one or more searchable fields. The `<label>` points to the translation of a screen text in *userLanguages.xml* whilst `<helpsubject>` refers to a help text in *userHelp.xml*. Underneath `<tag>` you provide one or more fields which must be searched simultaneously and with the `trunc` attribute you can indicate the type of index defined for the field if you want to have automatically truncated searches: long text fields like *title* (`ti`) and *abstract* (`sa`) are always word indexed ("`word`") (aka free text indexed) whilst all fields with a (single or compound) term are usually term indexed ("`term`"). In the first case, Internet Server will truncate every entered search key whilst in the other case only the last search key will be truncated (if you are searching on multiple words). If you leave the attribute out, the "`term`" type is implied. Whether automatic truncation is actually switched on or not, is set with the optional `<truncation>on</truncation>`. If you leave this element out, truncation is switched on by default.

For *Simple search* you set the `<tag>` attribute `database` to the database in which the listed fields must be searched: you only need to do this for *Simple search* because this search method allows the user to search more than one database simultaneously.

Via the optional `<tag>` attribute `operator` you can specify the comparison operator with which must be searched: in text fields you will usually search with `=`. If you leave the attribute out, the operator `=` is

implied. The allowed operators are determined by *wwwopac.ashx* functionality.

If also fields must be searched that appear in the comments or tags database, you'll have to indicate in which database the field tag can be found. (You can look up the tags of fields in your database structure files, via Adlib Designer, or directly in your running Adlib application, in the properties of an active field.)

```
<tag database="fullCatalogue" operator="=">
  <field trunc="word">ti</field>
  <field trunc="term">au</field>
  <field trunc="term">ca</field>
  <field trunc="word">sa</field>
  <field trunc="term">tr</field>
  <field trunc="term" database="tagging">TA</field>
  <field trunc="term" database="comments">CO</field>
</tag>
```

For *Advanced search* and *Expert search* you can include `<fieldname listbutton= "yes">` to display a list button behind term indexed entry fields, so that the user can request a list of existing terms.

If you want to use an enumerative field as a search field, you must list all possible (language neutral) values here again in the `value` attribute of the `<option>` tags. The *wwwopac.ashx* cannot retrieve those values. The translation of the texts to be displayed in the resulting drop-down list are retrieved from *userLanguage.xml* via the identifier specified in `<option>`.

```
<searchrow>
  <fieldname selected="yes" valuestaticlist="yes">
    <label>Field_Material</label>
    <helpsubject>material</helpsubject>
    <tag>
      <field trunc="term">ms</field>
    </tag>
    <value>
      <option value="" selected="yes">...Select object</option>
      <option value="Blu-ray">Blu-ray</option>
      <option value="Book">Boek</option>
      <option value="CD">CD</option>
      <option value="CD-ROM">CD-ROM</option>
      <option value="DVD">DVD</option>
      <option value="Video">Video</option>
    </value>
  </fieldname>
</searchrow>
```

When the user enters a value into a field, the autocomplete function will automatically open a drop-down list containing existing values starting with the entered characters. The contents of the drop-down

list changes during typing. This way the user might not have to type the whole word, by simply picking the desired value from the list. The autocomplete function can be switched on or off (default) per search field.

```
<fieldname autocomplete="yes" listbutton="yes" selected="no">
  <label>Field_Objectname</label>
  <helpsubject>objectname</helpsubject>
  <tag>OB</tag>
  ...
```

In *globalsettings.xml*, the `<autocompleteLimit>` must have been set to a positive value though.

For Expert search you have some extra possibilities. With the `<fieldname>` attribute `datepicker="yes"`, the interface of Internet Server 5 can display a calendar for a date field, from which the user can pick the desired date immediately.

```
<fieldname selected="yes" listbutton="no" datepicker="yes" type="ISODate">
  <label>Field_InputDate</label>
  <helpsubject>inputDate</helpsubject>
  <tag>di</tag>
  <operatorlist>
    <operator>between</operator>
    <operator>=</operator>
    <operator>&gt;=</operator>
    <operator>&lt;=</operator>
    <operator>&gt;</operator>
    <operator>&lt;</operator>
  </operatorlist>
  <truncation>off</truncation>
</fieldname>
```

The screenshot shows a search interface with a date picker. The search field contains "Input date" and the operator "Between". The date picker is open, showing a calendar for November 2012. The date "22" is highlighted by the mouse cursor. The interface also includes options for "Only records with images" (checked), "Sort by" (Unsorted), and "Sort order" (Ascending).

November 2012						
Mo	Tu	We	Th	Fr	Sa	Su
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		

And by means of an `<operatorlist>` you can specify which comparison operators can be used in a search entry line. The user must then pick the operator from a list (if more than one operator is available).

Expert search

Search in

= Trunc. and

from to Trunc. and

- Between
- Between
- =
- >
- <
- >
- <

- >
- <
- >
- <

Unsorted Title Creator Object name

If `copycurrent="yes"` has been set for a field, it means that whenever the user adds a new search entry line via the **+** button in the search form, the contents of this field will be copied to the new search entry line. This is handy if there's a good chance that the user will want to enter the same value in the new line.

```
<fieldname autocomplete="yes" listbutton="yes" selected="no"
copycurrent="yes">
<label>Field_Creator</label>
<helpsubject>creator</helpsubject>
<tag>
<field trunc="word">VV</field>
</tag>
...
```

You can have users search a range in a single field, between two years for example. Therefore use the `between` operator. The interface will display two entry fields, both associated with the same database field.

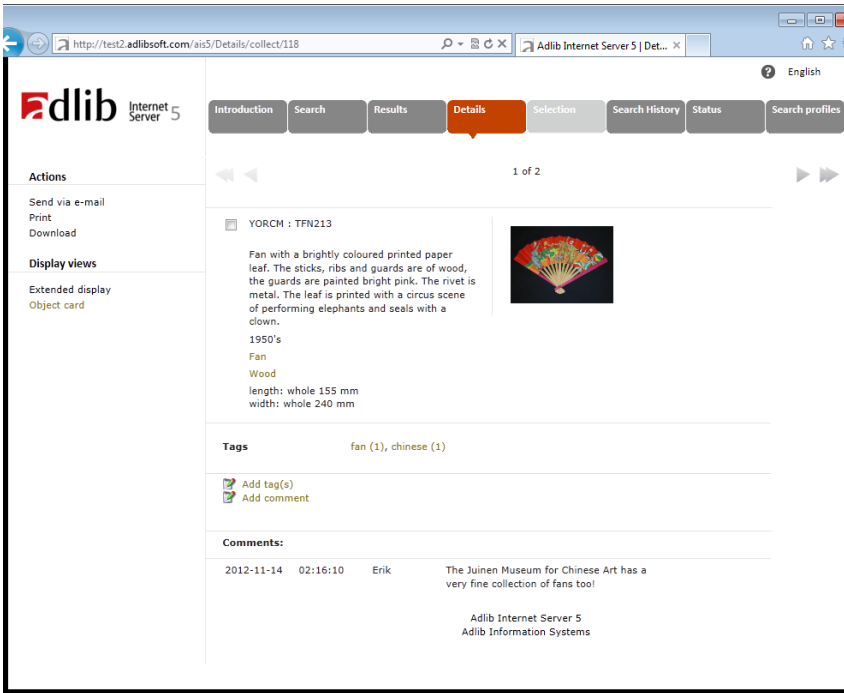
from to

```
<fieldname>
<label>Field_YearFrom</label>
<helpsubject>yearFrom</helpsubject>
<tag>DS</tag>
<operatorlist>
<operator>between</operator>
```

4.7 Functionality for comments and tags

Internet Server 5 has been made ready for the use of comments and tagging functionality. This option allows visitors of the website to leave comments about the currently displayed record: comments which will be seen by other visitors too and tags (keywords) with which one

must be able to find this record as well. The visitor can use this functionality to provide extra information, give a short comment or make the record more easily retrievable.



That Internet Server 5 has been prepared for this functionality, means that you can find the settings for it in *globalsettings.xml*, *formsettings.xml* and *adlibweb.xml*, although they might be commented out or switched off. There is two separate Adlib modules for Windows available, with which this functionality can actually be applied: Adlib Comments and Adlib Tagging. By default though, comments and tags databases are not part of your Adlib system, unless you requested the extra modules to be installed when you purchased Internet Server.

4.7.1 Setup of parts

The installation and setup of the comments and tags databases (SQL), if available in your Adlib system, has probably already been done for you, but it may be useful to know where you can find the different parts and configurations and how you can change them.

- Adlib applications and databases: the *adlib.pbk*'s of comments management and tags management can be found in the *Social indexing\Comments* and *Social indexing\Tagging* subfolders of your Adlib folder, the required screens in the *\screens* subfolder(s), and the database structures *comments.inf* and *tagging.inf* in the *\data* subfolder(s). (If these folders and files are not present in your Internet Server 5 system, then the functionality for comments and tags has not been implemented.)
In *adlibweb.xml* you'll find the configuration for both databases in their own *databaseConfiguration* section, for example:

```

<!-- ===== -->
<!-- Database Comments -->
<!-- ===== -->
<databaseConfiguration database="comments" groups="default">
  <database>comments</database>
  <databasepath>D:\AIS\Social indexing\Comments\data</databasepath>
  <briefFields>
    <field>*</field>
  </briefFields>
  <detailFields>
    <field>*</field>
  </detailFields>
  <writeAllowed>true</writeAllowed>
</databaseConfiguration>
<!-- ===== -->
<!-- Database Tagging -->
<!-- ===== -->
<databaseConfiguration database="tagging" groups="default">
  <database>tagging</database>
  <xmltype>grouped</xmltype>
  <highlight></highlight>
  <databasepath>D:\AIS\Social indexing\Tagging\data</databasepath>
  <briefFields>
    <field>*</field>
  </briefFields>
  <detailFields>
    <field>*</field>
  </detailFields>
  <writeAllowed>true</writeAllowed>
</databaseConfiguration>

```

- In *adlibweb.xml*, also an image server must have been configured for the images which users can upload with their comments, for example:

```

<imageServerConfiguration name="commentsimages">
  <servertype>FileSystem</servertype>
  <path>D:\AIS\internetserver5\uploads</path>
  <cachePath></cachePath>
  <imageOverlayFile></imageOverlayFile>

```

```

<imageOverlayPosition>4</imageOverlayPosition>
<imageOverlayBlend>50</imageOverlayBlend>
<imageOverlayPercentage>100</imageOverlayPercentage>
<imageOverlayMinPixelSize>500</imageOverlayMinPixelSize>
</imageServerConfiguration>

```

- In *globalsettings.xml*, within the `<session>` element, you have the possibility to switch the commentary and tagging functionality off entirely, or to decide whether approval is necessary before a record is published, amongst other settings. You may find the following section:

```

<comments database="comments" enabled="true" writeAllowed="true"
approveBeforeDisplay="false" addFormExpanded="false">
<approvalEmail>true</approvalEmail>
<rejectionEmail>true</rejectionEmail>
<requiredDisclaimer>true</requiredDisclaimer>
<commentsLimit>3</commentsLimit>
<approvedField>approved</approvedField>
<databases>
  <database name="collect" multimediaFileUploadPath=
    "../internetserver5/uploads" imageServer="commentsimages"/>
  <database name="fullCatalogue" multimediaFileUploadPath=
    "../internetserver5/uploads" imageServer="commentsimages"/>
  <database name="books" />
</databases>
<fileSizeLimit>40</fileSizeLimit>
<allowedExtensions>
  <fileType name="image">
    <extension>jpe</extension>
    <extension>jpeg</extension>
    <extension>jpg</extension>
    <extension>bmp</extension>
    <extension>tif</extension>
    <extension>tiff</extension>
    <extension>gif</extension>
  </fileType>
  <fileType name="video">
    <extension>avi</extension>
    <extension>wmv</extension>
    <extension>mp4</extension>
    <extension>mpe</extension>
    <extension>mpeg</extension>
    <extension>mpg</extension>
  </fileType>
  <fileType name="audio">
    <extension>mp3</extension>
    <extension>wav</extension>
  </fileType>
</allowedExtensions>
</comments>

```



```

<tagging database="tagging" enabled="true" writeAllowed="true"
  approveBeforeDisplay="false" addFormExpanded="false">
  <tagValueField>tag</tagValueField>
  <approvedField>approved</approvedField>
  <linkedDatabaseField>linked.database</linkedDatabaseField>
  <linkedPrirefField>linked.priref</linkedPrirefField>
  <databases>
    <database name="collect"/>
    <database name="fullCatalogue"/>
  </databases>
</tagging>

<externalDatabases>
  <database
    name="tagging"
    valueField ="tag"
    linkedDatabaseField="linked.database"
    linkedPrirefField ="linked.priref"
    referenceGroupName ="REFERENCE"
  />
  <database
    name="comments"
    valueField ="comments"
    linkedDatabaseField="referenceDatabase"
    linkedPrirefField ="referencePriref"
  />
</externalDatabases>

```

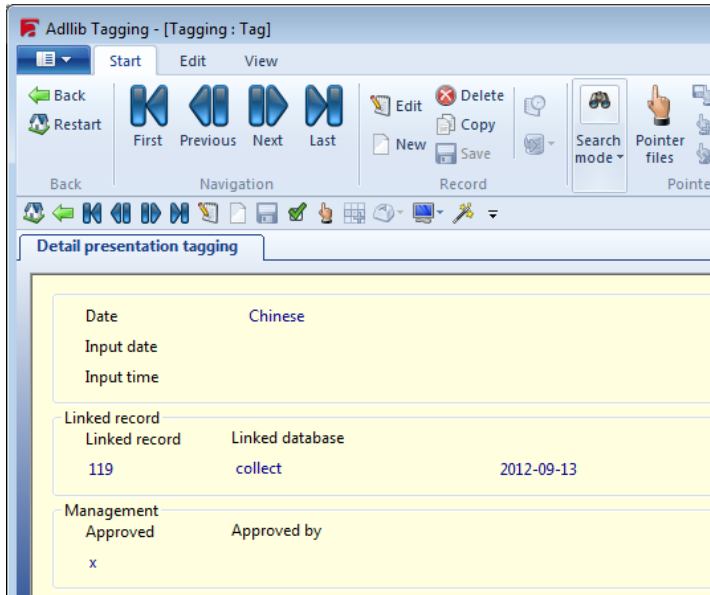
The used elements and attributes have the following meaning:

- *database* (attr.): the name of the comments or tagging database as it has been defined in *adlibweb.xml*.
- *enabled* (attr.): switches the commentary or tagging functionality completely on or off [*true*|*false*].
- *writeAllowed* (attr.): specifies whether new records can be written to the database [*true*|*false*]. This means that visitors cannot add comments or tags if this attribute has been set to *false*.
- *approveBeforeDisplay* (attr.): specifies if the administrator of your website has to approve an added comment or tag in the relevant Adlib module (running on *adlwin.exe*), before it will be displayed on the website [*true*|*false*]. If you set this option to *false*, the contents of the *Approved* checkbox (in the Adlib module) will be ignored.

- *addFormExpanded* (attr.): specifies whether the form (the entry fields) for adding comments or tags on the website, is visible by default or not [`true|false`]. If it is not visible by default, then the visitor will have to click the hyperlink for adding commentary or tags first to open the entry form in a separate window.
- *approvalEmail* (elem.) [`true|false`]: indicates whether the user must receive an e-mail as soon as the submitted comment has been approved, if applicable.
- *rejectionEmail* (elem.) [`true|false`]: indicates whether the user must receive an e-mail if the submitted comment has been rejected, if applicable.
- *requiredDisclaimer* (elem.) [`true|false`]: indicates whether the user is required to mark the *Disclaimer* checkbox before the entered comment can be sent.
- *commentsLimit* (elem.): specifies how many comments will be shown underneath a record in detailed display initially. If more comments are available than this maximum, the link *Read more comments...* appears underneath the displayed comments. As soon as the user clicks the link, all comments for that record will be shown.
- *approvedField* (elem.): must contain the name of the field in the comments or tags database in which the administrator indicates if a comment or tag has been approved. Only if `approveBeforeDisplay="true"`, will the field name be used to retrieve approved comments or tags only.
- *databases* (elem.): a list of databases (as specified in *adlibweb.xml*) for which adding comments or tags by visitors is allowed.
- *multimediaFileUploadPath* (attr.): specifies a physical path in which files uploaded by visitors will be stored. You can specify a path per database. In Windows, set the access rights for internet users to these folder(s) to *read* and *write*.
In IIS you'll then have to create a virtual folder for this physical path, named "uploads". *Uploads* must be a virtual subfolder of the virtual main folder for your web applica-

tion. See elsewhere in this guide for information about creating a virtual folder.

- *imageServer* (attr.): refers to the name of the image server (as specified in *adlibweb.xml*) that you created for the uploading of files.
- *fileSizeLimit* (elem.): is the maximum allowed size (in megabytes) of files to be uploaded with comments. With a small maximum file size you prevent your web server from possibly becoming overloaded, and you reduce the chance that any DoS attacks which could floor you web server, might be succesful. With a large maximum you allow for the possibility to upload large (video) files.
- *allowedExtensions* (elem.): contains a list of file extensions of the file types allowed for uploading with submitted comments. There are three `fileType` sections (`name= [image|video|audio]`).
- *tagValueField* (elem.): must contain the name of the field in the tags database, which contains the submitted word (the tag).
This option is relevant for the display of tags in the detail screen of a record.
- *linkedDatabaseField* (elem.): optional, must be the name of the field which contains the name of the database linked to the tag record: when a tag is registered in the tags database, the name of the database of the catalogue record to which the tag applies, is stored in this field explicitly (see the image below). If the setting is missing, automatically the field name "`linked.database`" will be used, which is the default.
This option is relevant for the display of tags in the detail screen of a record.



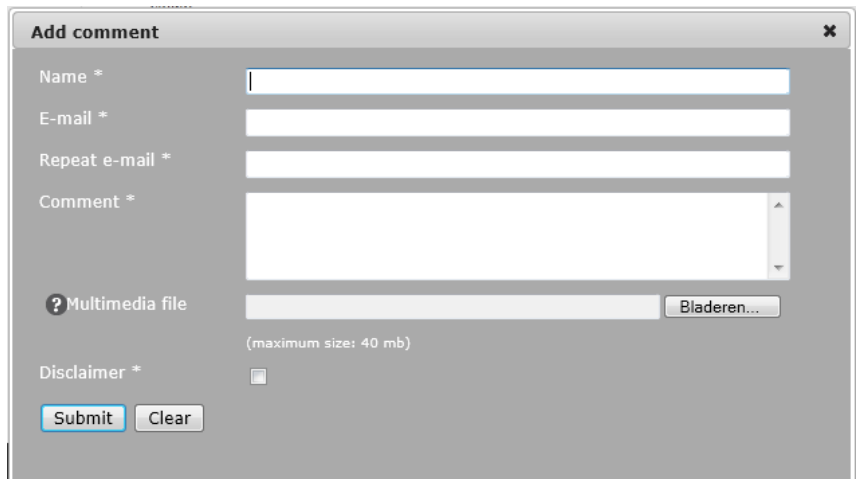
- *linkedPrirefField* (elem.): optional, must be the name of the field which contains the record number of the catalogue record linked to the tag record: when a tag is registered in the tags database, the record number of the catalogue record to which the tag applies, is stored in this field explicitly (also see the image above). If the setting is missing, automatically the field name "**linked.priref**" will be used, which is the default. This option is relevant for the display of tags in the detail screen of a record.
- *externalDatabases* (elem.): to able to use the website to search for records via words appearing in the tags and comments database, you must provide the relevant database and field names here. The **name** of both databases is as specified in *adlibweb.xml*. **valueField** must contain the name of the field in the tags or comments database, which contains the submitted word (the tag) or the comment text. The **linkedDatabaseField** must be the name of the field which contains the name of the database linked to the tag or comments record: when a tag or comment is registered in the database, the name of the database of the catalogue record to which the value applies, is stored

in this field explicitly. The `linkedPrirefField` must be the name of the field which contains the record number of the catalogue record linked to the tag of comments record. For the tags database you must also specify the field group name of the repeated fields that point to the linked records, in `referenceGroupName`. In most cases you can leave these settings to their default values.

4.7.2 Functionality in the Internet Server interface

Once the comments and tagging functionality has been implemented, visitors will find either the hyperlink *Add comment* and/or *Add tag(s)* at the bottom of the detailed display of a record. As soon as the visitor clicks one of the hyperlinks, an entry form opens; see the images below. Entered comments will be saved in the relevant new database. Future visitors opening the relevant record, will automatically get to see the accompanying comments. The name of the visitor and the entry date will be shown as well. The submitted e-mail address won't be visible and will only be used to automatically send a confirmation e-mail to the visitor about the published comments. Added tags can be found directly underneath the detailed display and above the comments.

■ Entering comments



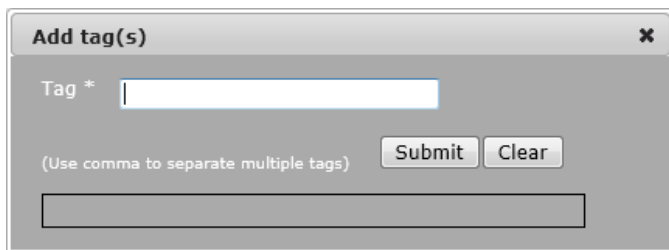
The screenshot shows a web form titled "Add comment" with a close button (X) in the top right corner. The form contains the following fields and controls:

- Name ***: A text input field.
- E-mail ***: A text input field.
- Repeat e-mail ***: A text input field.
- Comment ***: A large text area with a vertical scrollbar on the right side.
- ? Multimedia file**: A file selection field with a "Bladeren..." button next to it. Below this field, it says "(maximum size: 40 mb)".
- Disclaimer ***: A checkbox.
- Submit**: A button.
- Clear**: A button.

- The *Name* entered by the user will appear next to the public comment.
- The e-mail address must be entered twice, to minimize the chance of errors.

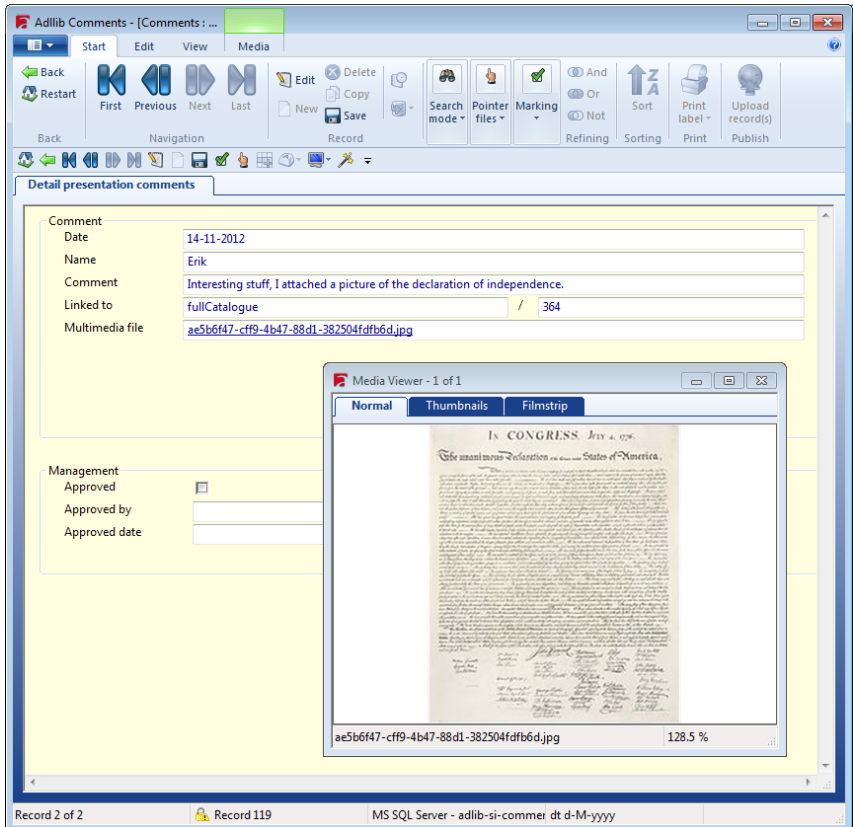
- In the (optionally present) *Multimedia file* entry field, the user may submit an audio, video or image file, which will be available with the comment. The desired file can be selected on the user's own hard drive via the *Browse* button: this means the selected file will be uploaded to the server of the website.
To the right of the displayed comments, an icon will be shown: a thumbnail of a linked image, or an icon representing an audio or video file. Clicking the icon, opens the relevant file in a new browser tab or window and displays the image or plays the file if it concerns audio or video.
- Read the *Disclaimer* by clicking it and mark the checkbox to indicate that you agree.
- Click the *Submit* button to submit the comment. *Clear* empties all fields.

■ Entering tags



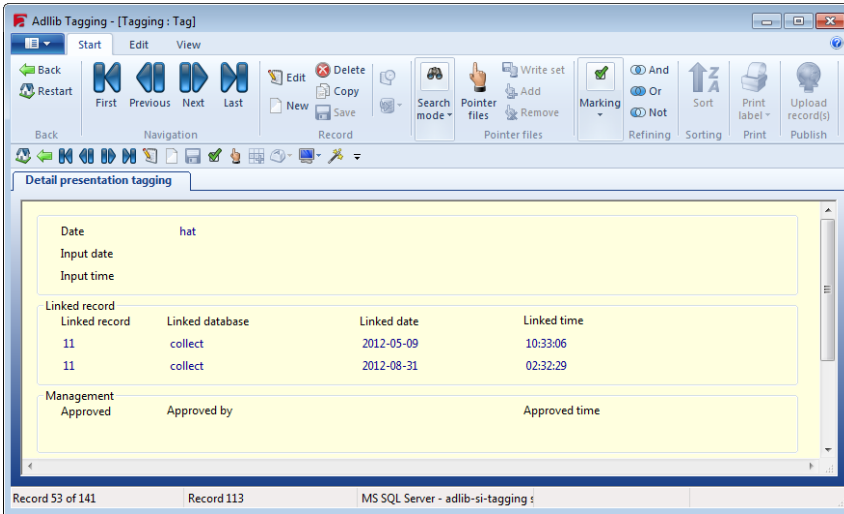
- In the *Tag* entry field, enter a keyword which should be relevant to the current record, so that other users can find this record when they use the keyword in their search.
A user can click the tag displayed in the website, to show all records to which that tag has been assigned. The number between parentheses, displayed next to a tag, indicates how many users have entered the relevant tag for the current record.
- You can enter multiple tags at once by using commas between the words.
- Click the *Submit* button to submit the tag. *Clear* empties the field.

4.7.3 Comments and tags database management



Already entered comments or tags can be managed separately in their own Windows Adlib application (running on *adlwin.exe*).

Both the Adlib Comments and Tagging modules running on *adlwin.exe* are very simple: the only thing you can do in it (besides the standard Adlib functionality) is remove records (because you no longer want to display the comments on the website), adjust records (for instance to clean up the language in it or to remove linked files) and approve or reject records for publication. You cannot enter new comment records, but you *can* enter new tag records.



4.8 Customization examples

With the information provided in chapters 4.4 up to and including **Error! Reference source not found.** and some insight into XML, XSLT and CSS you should be able to make minor adjustments to your Internet Server web application yourself. Of course it is wise to create a backup of the current situation first and to try your customization in a test environment before you apply it to the live environment.

4.8.1 Adding a field to the searched fields

To search a field by means of the Internet Server and to display the field in a brief or detailed display of a record, are two separate things. To just add a field to search on, you'll sometimes only need to make changes to *formsettings.xml*, but often you'll also have to add some settings to *userLanguage.xml* and *userHelp.xml*, which can all be found in the *\Config* subfolder of your Internet Server system.

1. Decide to which search method(s) (*Simple search*, *Advanced search* or *Expert search*) you want to add a search field. For advanced and expert search, the fields currently available to the user are clear from the interface. Note that the search fields list may change if you select a different database to search in.

Advanced search

Search in

? Word(s) from the title / description

? Creator

? Object name

? Object category (custom)

? Subject

? Material

? Technique

? Tags

? Comment

Only records with images

? Sort by Unsorted Title Creator Object name

Sort order Ascending Descending

Expert search

Search in

? Word(s) from the title = Trunc. and

? Word(s) from the title

? Word(s) from the abstract

? Author: Unsorted Title Author Year

Corporate author Ascending Descending

Publisher

ISBN

ISSN

? Subject term 10 25 50

? Year

Tags

Comment

The *Simple search* hides the searched fields from view.

- In *formsettings.xml* you can find the settings for each search fields list, so open *formsettings.xml* in a text or XML editor and look up the `<screen>` node of your choice (`style="simple"`, `style="advanced"` or `style="expert"`). In the `<searchform>` node underneath it you'll find the name of the search form definition used by that search method. That would be `simple_database` for the *Simple search*, for example.

```
<screen style="simple" isDefault="yes">
  <databasechoice name="all">
    <databasecheck name="collect" selected="yes" />
    <databasecheck name="fullCatalogue" selected="no" />
    <databasecheck name="archive" selected="no" />
    <!--<databasecheck name="comments" selected="yes" />!-->
    <searchform>simple_database</searchform>
  </databasechoice>
</screen>
```

- Look up the relevant `<formdefinition>` node a bit further down in the *formsettings.xml* file. For the current database or for all currently available databases (`fullCatalogue`, `collect`, `archive`) you'll see the currently set up search fields list, for example:

```
<formdefinition name="simple_database">
  <searchrow>
    <fieldname>
      <label>Search_SearchSimple_FieldLabel</label>
      <helpsubject>Simpletitle</helpsubject>
      <tag database="fullCatalogue" operator="=">
        <field trunc="word">ti</field>
        <field trunc="term">au</field>
        <field trunc="term">ca</field>
        <field trunc="word">sa</field>
        <field trunc="term">tr</field>
      </tag>
    </fieldname>
  </searchrow>
  ...
</formdefinition>
```

- Now, `fullCatalogue`, for example, is only an alias for the real name of the database (or the name of the *.inf* file containing the database structure settings). We need the real database name to be able to look up some field definitions. Open *adlibweb.xml* from your `\wwwopacx` folder to find out what the real database name is. It turns out to be `document`; `fullcatalogue` is (also) the name of a dataset within that database. We only need the database name now.

```
<!-- ===== -->
<!-- Database Full Catalogue -->
<!-- ===== -->
```

```
<databaseConfiguration database="fullCatalogue"
  groups="defaultLibrary">
  <database>document&gt;fullcatalogue</database>
</databaseConfiguration>
```

5. Adlib Designer is the perfect tool to look up more information about database structures, indexes and fields. See the Designer Help for all information about using Adlib Designer. In the Application browser, open the definition of the `document` database (underneath the `\data` folder in your Adlib system). You can see what fields have been defined, what their field tag is, if they have an index and if the index is a word (`free text`) or term (`text`, or `integer` if it concerns a link reference index for a linked term field) index. This way you can find out that the document fields searched by the *Simple search* method in Internet Server are `title (ti)`, `author.name (au)`, `corporate_author (ca)`, `abstract (sa)` and `keyword.contents (tr)`.

The screenshot shows the Adlib Designer interface. On the left, a tree view displays the database structure under the 'data' folder, including various fields like 'auction', 'borrower', 'budget', etc., and a 'document' folder containing 'Datasets (9)', 'Indexes (69)', and 'Fields (252)'. The 'author.name (au)' field is selected. On the right, the 'Field properties' pane is open, showing 'Linked field properties'. The 'Field tag' is 'au'. The 'Field type' is 'Linked field'. The 'Field names' section shows a table with columns 'Language' and 'Text'.

Language	Text
English	author.name
Dutch	auteur.naam
French	auteur.nom
German	Verfasser
Arabic	المؤلف
Italian	
Greek	συντάκτης,όν

Look up the field that you want to add to a search fields list and see if it already has an index. (A field can only be searched if it has been indexed; of a linked field, like `author.name`, its link reference tag must have been indexed instead of the linked field itself.) In the properties of the index you can find out what the type of the index is. If the field you want to add hasn't got an index, you will have to create it in Designer first.

6. Now we have all the information we need to add our field(s) to the search fields list in *formsettings.xml*. In the *Simple search* example below we added three fields: *year_of_publication* (ju) for which we had create a new term (text) index, the linked *geographical_keyword* (GT) term field which link reference tag lg was already indexed, and the *notes* field (an) indexed as free text.

```
<formdefinition name="simple_database">
  <searchrow>
    <fieldname>
      <label>Search_SearchSimple_FieldLabel</label>
      <helpsubject>Simpletitle</helpsubject>
      <tag database="ChoiceFullCatalogue" operator="=">
        <field trunc="word">ti</field>
        <field trunc="term">au</field>
        <field trunc="term">ca</field>
        <field trunc="word">sa</field>
        <field trunc="term">tr</field>
        <field trunc="term">ju</field>
        <field trunc="term">GT</field>
        <field trunc="word">an</field>
      </tag>
    </fieldname>
  </searchrow>
  ...
</formdefinition>
```

7. The museum_advanced *Advanced search* form as used for *collect* (the alias for *collect.inf*) looks a bit different. Every field (or group of similar fields) is defined within its own *<searchrow>* node which results in a single entry field on the form.

```
<formdefinition name="museum_advanced">
  <searchrow>
    <fieldname>
      <label>Field_TitleDescription</label>
      <helpsubject>title</helpsubject>
      <tag>
        <field trunc="word">TI</field>
        <field trunc="word">BE</field>
      </tag>
      <truncation>on</truncation>
    </fieldname>
  </searchrow>
  <searchrow>
    <fieldname listbutton="yes">
      <label>Field_Creator</label>
      <helpsubject>creator</helpsubject>
      <tag>VV</tag>
      <truncation>on</truncation>
    </fieldname>
  </searchrow>
  ...
</formdefinition>
```

You can add fields here too. The same conditions apply. You could insert the following, for example, to add a search row for the object category:

```
<searchrow>
  <fieldname listbutton="yes">
    <label>Field_Objectcategory</label>
    <helpsubject>objectcategory</helpsubject>
    <tag>OC</tag>
    <truncation>on</truncation>
  </fieldname>
</searchrow>
```

Note that the entered `Field_Objectcategory` field label identifier must be created as `<object name="Field_Objectcategory">` (comparable to `<object name="Field_Subject">` for example) in *userLanguage.xml* first and the `objectcategory` help item identifier as `<object name="objectcategory">` (comparable to `<object name="objectname">` for example) in *userHelp.xml*, before you can use them here. The label is required because you want a label in the proper language to appear in front of the entry field on the *Advanced search* form, while the help item serves to explain the entry field to users if they click the field label.

8. Save your changes to *formsettings.xml*. Restarting your browser may be enough to be able to test the results of your adjustments, but to be sure recycle the IIS application pool for the Internet Server web application (also see chapter 4.9).

4.8.2 Adding a field to the detailed display

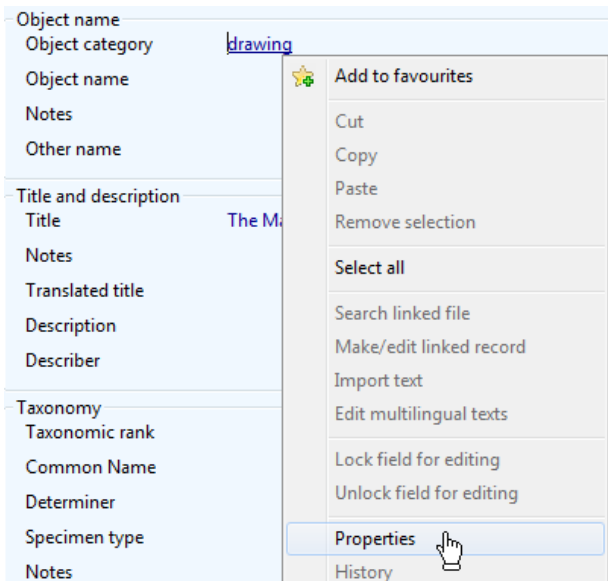
The screenshot shows the dlib Internet Server interface. The top navigation bar includes 'Introduction', 'Search', 'Results', 'Details' (highlighted), 'Selection', 'Search History', 'Status', and 'Search profiles'. The main content area displays a watercolor painting of a tree with a large, glowing apple. Below the image, the following metadata is shown:

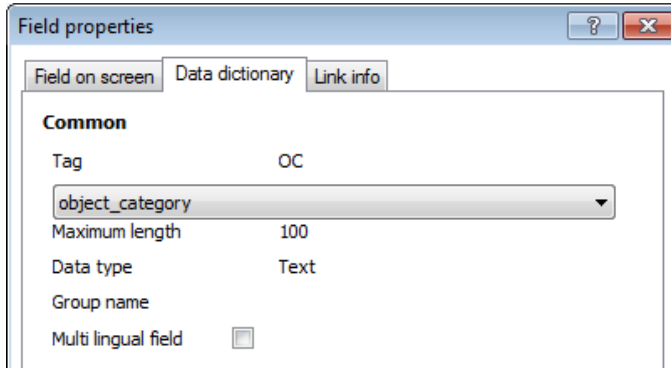
Title	The Magic Apple Tree
Object number	1490
Creator	Samuel Palmer (draughtsman)
Date	1830 - 1830
School/style	British
Material	Indian ink, watercolour, gum Arabic, paper
Technique	watercolour
Dimensions	<ul style="list-style-type: none"> • height: 349 mm • width: 273 mm

As mentioned in the previous paragraph, to display the field in a brief or detailed display of a record is separate from the searchability of the field. To just add a field to a detailed display, for example, you may need to make a change to the *adlibweb.xml* file in your *\wwwopacx* folder and you'll usually need to make changes to *userLanguage.xml* which can be found in the *\Config* subfolder of your Internet Server system, and to the XSLT stylesheet responsible for the layout of the detailed presentation (which can be found in the *\Views\Results* subfolder of your Internet Server).

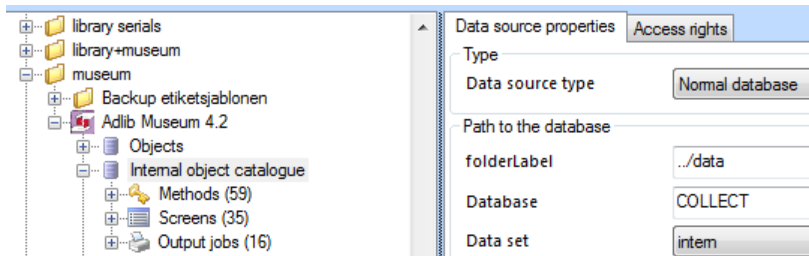
For this example, let's assume we want to add the *Object category* field to the *Extended display* presentation of a *Museum collection* record, just below the present *Title* field.

1. We first we need to find out what the data dictionary field name and tag of the *Object category* field are. You can do that via your Adlib application, for example. Open an object record in the *Internal object catalogue* of your Adlib Museum application. In the detailed display of the record, right-click the *Object category* field contents on the *Identification* tab and click *Properties* in the pop-up menu. On the *Data dictionary* tab of the *Field properties* window you'll see the real name and the tag of the field: *object_category* and *OC* in our example. We can also see it is not part of any field group, so in the grouped XML search result the field can be found directly underneath the *<record>* node.





- In Adlib Designer you can find out what the data dictionary name of the database behind the *Internal object catalogue* data source is, if you're interested. In the Application browser, open the Museum application definition and select the *Internal object catalogue*. In the *Data source properties* you can see that the real database name is `collect` (containing the dataset `intern`).



- Now open *formsettings.xml* from your Internet Server \Config subfolder. In *formsettings.xml* look for the alias of the *Museum collection* database for which you want to adjust a detail presentation. At the top of the file you can find three `<screen>` nodes representing *Simple*, *Advanced* and *Expert search* (`style="simple"`, `style="advanced"` or `style="expert"`). In the `<databasechoice>` node(s) underneath them you'll find the database aliases used by that search method. They appear in the same order as they are presented in the Internet Server user interface. For *Simple search* for example:

```
<screen style="simple" isDefault="yes">
  <databasechoice name="all">
    <databasecheck name="collect" selected="yes" />
    <databasecheck name="fullCatalogue" selected="no" />
    <databasecheck name="archive" selected="no" />
    <!--<databasecheck name="comments" selected="yes" />-->
    <searchform>simple_database</searchform>
  </databasechoice>
</screen>
```

Simple search

Search in

- Museum collection
- Library catalogue
- Archives

So `collect` is the database alias we were looking for.

- We now have to make sure that the `object_category` field from the `collect` database is actually included in the retrieved record XML. Therefore we'll need to check `adlibweb.xml`. Open it from your `\wwwopacx` folder into a text or XML editor. Look up the relevant `<databaseConfiguration>` section. First we get confirmation that `collect` is indeed the alias for the `collect` database (coincidentally with the same name).

```
<!-- ===== -->
<!-- Database Collect -->
<!-- ===== -->
<databaseConfiguration database="collect"
  groups="defaultMuseum">
  <database>collect>intern</database>
</databaseConfiguration>
```

Further we learn that `groups="defaultMuseum"`. This refers to a `<groupConfiguration>` section with this name, higher up in `adlibweb.xml`, which will look something like the following:

```
<groupConfiguration group="defaultMuseum">
  <brieffields>
    <field>reproduction.web_exclusion</field>
    <field>object_name</field>
    <field>object_number</field>
    <field>creator</field>
    <field>title</field>
    ...
  </brieffields>
```



```

    <detailfields>
      <field>object_number</field>
      <field>title</field>
      <field>creator</field>
      <field>creator.role</field>
      ...
    </detailfields>
  </groupConfiguration>

```

The `object_category` field is not yet part of the `<detailfields>` list, so we'll have to insert it in there: `<field>object_category</field>`. Save your changes to `adlibweb.xml`. (If the detail fields just list `<field>*</field>`, all database fields are already being retrieved, but note that this might lower your website performance.)

5. Now that's out of the way, we must find the proper XSLT stylesheet and adjust it so that the retrieved field will be displayed in the right place. Open `globalsettings.xml` from your `\Config` subfolder and search for the `<dataset>` node for `collect`.

```

<dataset type="collect" presentationMemory="true">
  <file selected="true" display="brief"
    label="Search_DisplayView_BriefMuseum"
    transformationFile="briefMuseum" />
  <file selected="false" display="brief"
    label="Search_DisplayView_BriefMuseumThumbs"
    transformationFile="briefMuseumThumbs" />
  <file selected="false" display="detail"
    label="Search_DisplayView_FullDetailMuseum"
    transformationFile="detailMuseum" />
  <file selected="true" display="detail"
    label="Search_DisplayView_ShortDetailMuseum"
    transformationFile="detailShortMuseum" />
  <file selected="true" display="rss" label=""
    transformationFile="rssMuseum" sortOrder="input.date" />
  <addToQuery></addToQuery>
  <downloadfields>
    <field>example_field</field>
    <field>Reproduction/reproduction.identifier</field>
  </downloadfields>
</dataset>

```

Here, two stylesheets for the brief display and two for the detailed display are listed. The default ones have `selected="true"`. In the Internet Server interface we can see the two detailed presentations labeled as the *Extended display* and *Object card*.

The screenshot shows the 'Internet Server 5' interface. At the top, there are navigation tabs: 'Introduction', 'Search', 'Results', and 'Details' (which is highlighted in orange). Below the tabs, there are navigation arrows and a page indicator '2 of'. On the left side, there are sections for 'Actions' (Send via e-mail, Print, Download) and 'Display views' (Extended display, Object card). The main content area displays the details for 'The Magic Apple Tree' (1490) by Samuel Palmer (draughtsman), a British artist. The materials listed are Indian ink, watercolour, gum Arabic, paper, and watercolour. A small thumbnail image of the artwork is visible on the right.

Since the *Object card* is the default one (as shown in the image above), the *Extended display* corresponds to the *detailMuseum.xslt* stylesheet. (Of course, you can verify this by looking up the actual label translation in *userLanguage.xml*, referred to by the `label="Search_DisplayView_FullDetailMuseum"` attribute.)

- Now open *detailMuseum.xslt* from your `\Views\Results` subfolder in a text or XML editor. To explain the entire code here is beyond the scope of this text, but note that almost at the top of the file, all currently listed fields in the *Extended display* (or the field groups those fields are part of) are referenced for the first time.

The screenshot shows the 'Extended display' view for 'The Magic Apple Tree'. On the left, there is a sidebar with 'Display views' (Extended display, Object card). The main content area features a thumbnail of the painting 'The Magic Apple Tree' by Samuel Palmer. Below the image is a table of metadata:

Title	The Magic Apple Tree
Object number	1490
Creator	Samuel Palmer (draughtsman)
Date	1830 - 1830
School/style	British
Material	Indian ink, watercolour, gum Arabic, paper
Technique	watercolour
Dimensions	<ul style="list-style-type: none"> height: 349 mm width: 273 mm

```
<xsl:call-template name="ais-detail-with-label">
  <xsl:with-param name="fieldname" select="Title" />
</xsl:call-template>
<xsl:call-template name="ais-detail-with-label">
  <xsl:with-param name="fieldname" select="object_number" />
</xsl:call-template>
<xsl:call-template name="ais-detail-with-label">
```

```
<xsl:with-param name="fieldname" select="Production" />
</xsl:call-template>
...
```

They are all called for a template named `ais-detail-with-label`, which turns out to reside in an “included” `AISDetailTemplates.xslt` file in the `\Views\Results\Libraries` subfolder (`<xsl:import href="Libraries/AISDetailTemplates.xslt" />`). This template treats all the fields the same way, so we can just add the following code to the above listing in `detailMuseum.xslt`, underneath the call for `Title`:


```
<xsl:call-template name="ais-detail-with-label">
  <xsl:with-param name="fieldname" select="object_category" />
</xsl:call-template>
```

7. We do need a new label for *Object category*. The `ais-translate-fieldname` template in `AISTranslations.xslt` (called from within the `ais-detail-with-label` template in `AISDetailTemplates.xslt`) must be able to retrieve the proper translation of this label from `userLanguage.xml` (by means of yet another template called `ais-translate-label`). The field label identifier is automatically put together in the `ais-translate-fieldname` template, so we must name it accordingly, namely: `Field_object_category_Detail`. Add something like the following code to `userLanguage.xml`:

```
<object name="Field_object_category_Detail">
  <text property="String" language="ar-SA">Object category</text>
  <text property="String" language="en-GB">Object category</text>
  <text property="String" language="nl-NL">Objectcategorie</text>
  <text property="String" language="de-DE">Object category</text>
  <text property="String" language="fr-FR">Object category</text>
  <text property="String" language="it-IT">Object category</text>
  <text property="String" language="sv-SE">Object category</text>
  <text property="String" language="nb-NO">Object category</text>
  <text property="String" language="ru-RU">Object category</text>
  <text property="String" language="es-ES">Object category</text>
</object>
```

Make sure that the translations of the label are correct as far as the languages are concerned in which your website is available. For the other languages you can enter the English translation.

8. Save your changes to the edited files. Recycle the IIS application pool for the Internet Server web application to be able to test the results of your adjustments (also see chapter 4.9).



Introduction
Search
Results
Details
Selection
Sea

Actions

Send via e-mail


Print

Download

Display views

Extended display

Object card



Title	The Magic Apple Tree
Object category	drawing
Object number	1490
Creator	Samuel Palmer (draughtsman)
Date	1830 - 1830
School/style	British
Material	Indian ink, watercolour, gum Arabic, paper
Technique	watercolour
Dimensions	<ul style="list-style-type: none"> • height: 349 mm • width: 273 mm

4.9 Visiting the Internet Server website locally

If you have just made some changes in *globalsettings.xml* or *adlib-web.xml*, then the application pool of the web application is automatically recycled to activate the changes. A subsequent refresh of the website is enough to view the effects of the new settings, although you may have to enter a new search etc. to end up on the same page. After changes in other files of the web application, you may have to restart your browser to see the results.

To visit your website locally, do the following:

1. Start your internet browser from the station on which you installed the Adlib Internet Server, and enter the following URL:
`http://localhost/AdlibWebApp` Instead of `AdlibWebApp`, fill in the name of your own virtual folder.
2. You can also visit the web application from other desktops when your server has been set up in a network environment. Start the browser from a workstation that's in the same network as the server on which you installed Adlib Internet Server. Enter the URL from the previous step again, but replace `localhost` by the name of the server in the network or the ip-address of the server, for

example `http://local_server/AdlibWebApp` or
`http://192.168.0.3/AdlibWebApp`. The Adlib Internet Server
web application now opens.

* Instead of `AdlibWebApp`, fill in the name of your own virtual
folder.

If the web application doesn't work as expected, it might be sensible to test the backend (`wwwopac.ashx`) separately from the frontend (the web application). That way you can find out more easily where the problem lies. In the browser, you can approach `wwwopac` directly and submit a query, after which the search result will be displayed as XML in the browser. See <http://api.adlibsoft.com/site/> for examples of `wwwopac.ashx` queries.

If after the installation of the Adlib Internet Server on a Windows 7 machine the menus won't display on the website and clearly no CSS layout has been applied to presented pages, then it's possible you still have to switch on a certain Windows feature. On the server, open the *Windows Control Panel*, choose *Programs and Features* and click *Turn Windows features on or off* in the left column. In the *Turn Windows features on or off* window, select *Internet Information Services > World Wide Web services > Common HTTP features* and mark the *Static Content* checkbox if that hasn't been done yet. *HTTP Errors* and *Default Document* must be switched on as well.

■ Testing the OAI Server

See chapter 3.6 in the *WWWOPAC reference guide* for information about testing an OAI Server based on `oaiserver.exe`, and see chapter 7.4 for information about testing an `oai.ashx` server.

5 User authentication

Adlib applications (Windows applications as well as the web applications) which run on an SQL database can be secured in different ways: some users should only be allowed to retrieve and view data, while others may enter and/or remove data or even get to manage the database itself. Therefore, users must be authenticated before they are allowed to work with Adlib. Authentication of users for access to your SQL database can essentially be set up in two ways:

- **SQL authentication in combination with Adlib access rights**
 - In this case, the Adlib core software always connects to the server via one and the same general user name and password which are set in the Adlib database structure files (*.inf*). That one “user” must get sufficient permissions in the SQL database, so that in principle the database can be managed in its entirety. The limiting access rights for the actual individual users, must be set in the Adlib application structure files (*.pbk*); see the *Adlib Designer Help* for more information about this. In this setup, those *.pbk* files do need to be located in a secured, e.g. virtual, Adlib folder, to prevent them from being modified by unauthorized persons; see the *Installing Museum, Library and Archive* guide for information about setting up a virtual folder for Adlib structure files. The advantage of this authentication method is that the access rights management mainly takes place in Adlib, and can be done by an Adlib application manager. This authentication method is also the easiest method for solving any individual problems with establishing a connection to the SQL database in a multi-server environment; this is because the other authentication method (see below) uses Active Directory, which may sometimes complicate user authentication in a multi-server environment. A disadvantage may be that user names and passwords are located in an Adlib *.pbk* file which needs to be secured well. Also, all Adlib users must actually be registered and managed in the *.pbk*.
- **Windows authentication by means of Active Directory, possibly in combination with Adlib access rights** – With this method, you use the Windows login data (user names and passwords) which has already been registered in Active Directory for your local network. For the benefit of Adlib, those users must, as much as possible, be divided into groups which should be assigned different access rights in SQL Server. So, access of the individual user to the SQL database depends on the name and the password with which the user is logged onto the local network. Any further

refinement of the access rights can be taken care of in Adlib. An advantage of this method is that user names and passwords are well secured in Active Directory, and that all users of the network are already registered; only for the benefit of Adlib you'll still have to divide the users into groups which can then be assigned certain access rights per group. A possible disadvantage in a multi-server environment is that each server has its own Active Directory (server 1 could have a separate domain), and because of this it may sometimes be difficult to streamline user authentication.

Anonymous internet users who retrieve data from the Adlib SQL database via your web application, enter your network under one and the same IIS (Internet Information Services) account name, by default this is `IUSR_<server>` in which `<server name>` has been replaced by the actual name of the server on which the web application runs; under Windows 2008 (and Vista and higher), the default account name is just `IUSR`.

If you use SQL authentication, in principle all users have full access rights, but because internet users can only access the database via `wwwopac` and since `wwwopac` can only *retrieve* data (unless specified differently in the web configuration file) and cannot write or delete, your database is already protected quite well this way.

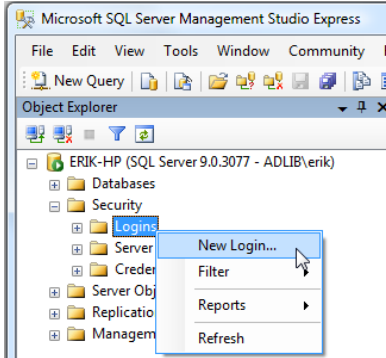
A further refinement of the access rights for anonymous internet users is possible, for instance by specifying access rights on record level in Adlib, to allow you to exclude certain records from every possible search result.

However, because the `IUSR` account cannot always be used, you'll have to create another Active Directory account yourself, under which anonymous internet users will access your website. That new account name must then be used to shield records. See chapter 5.3 for a full explanation of how to set this up.

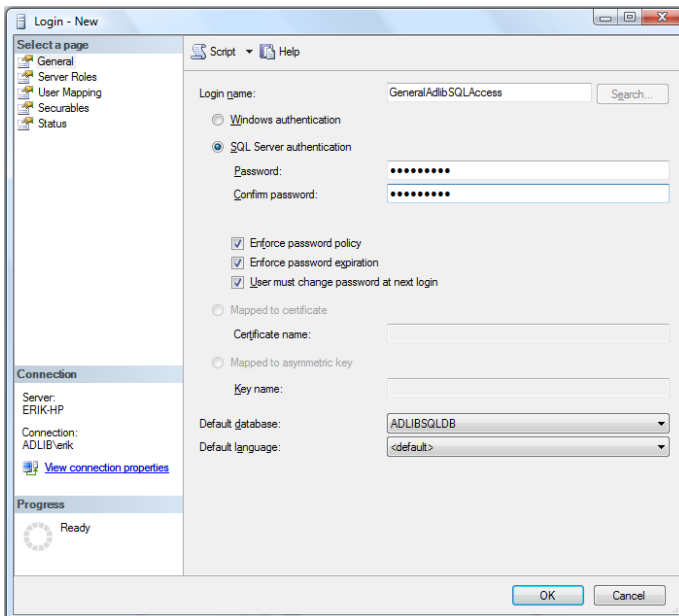
Which authentication method is to be preferred, depends on the way in which your Adlib system has been installed, on the setup of your local network, and on your own preferences and security policy. Below, you'll find a step-by-step procedure for setting up either authentication method properly.

5.1 Setting up SQL authentication

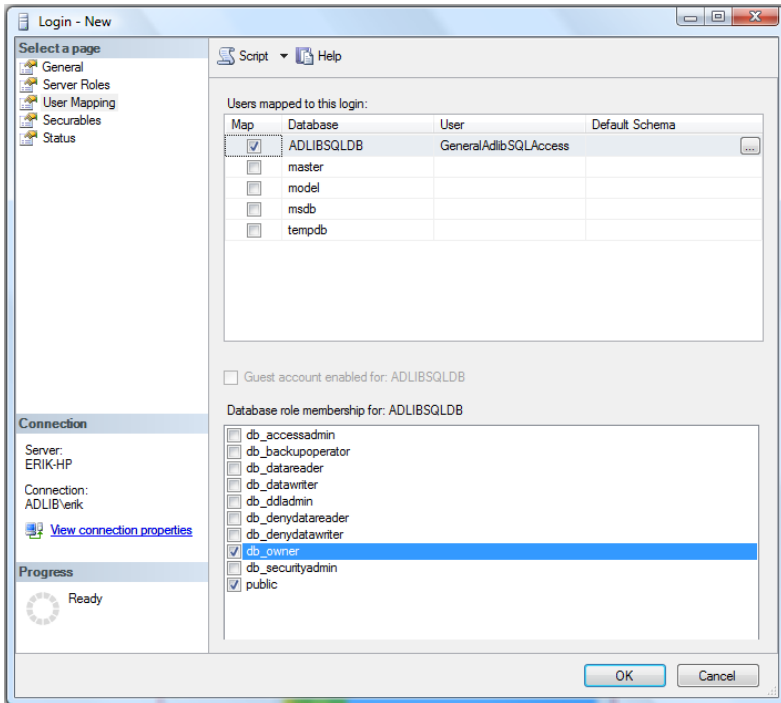
1. Start Microsoft SQL Server Management Studio (Express), open the *Security* folder underneath the SQL Server folder and right-click *Logins*. In the pop-up menu which opens, click the *New Login...* option.



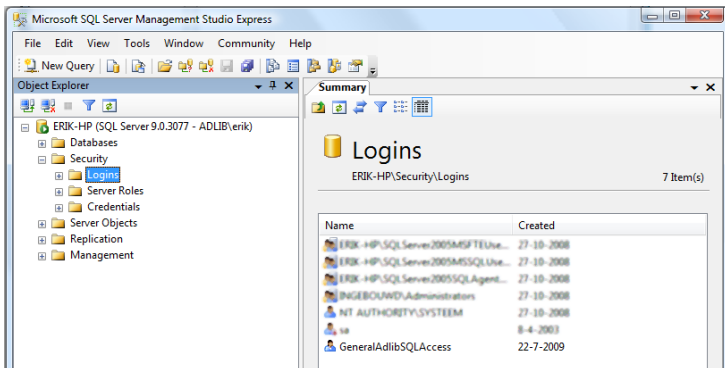
2. In the *Login – new* window, choose a sensible login name, mark the *SQL Server authentication* option, provide a password, confirm the password, and choose your Adlib SQL database as the *Default database*. (Do choose a hard to guess and sufficiently long password, otherwise the program produces an error message when you close this window.) Then unmark the *Enforce password expiration* and *User must change password at next login* checkboxes (remove the check).



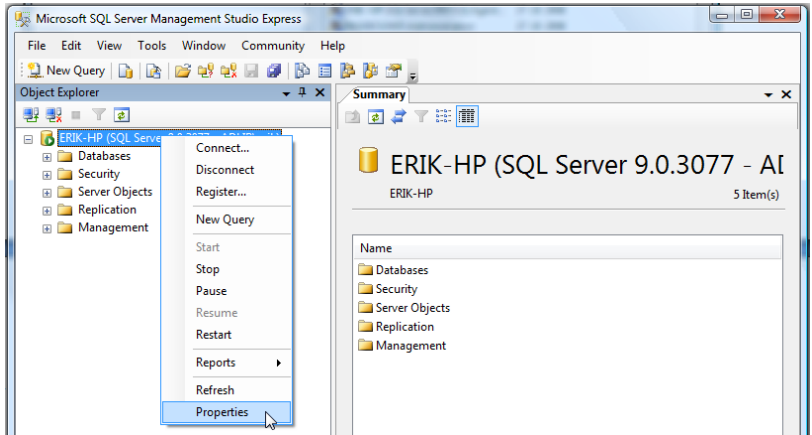
3. Select the *User mapping* page in the current window. In the list on the right, again mark your Adlib SQL database, and in the list below it, mark the *db_owner* role. (Leave the *public* role marked.)



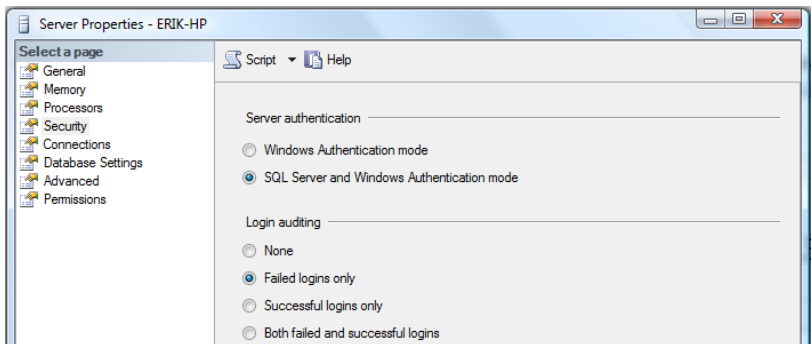
Then click **OK**. The new login is now present in the list.



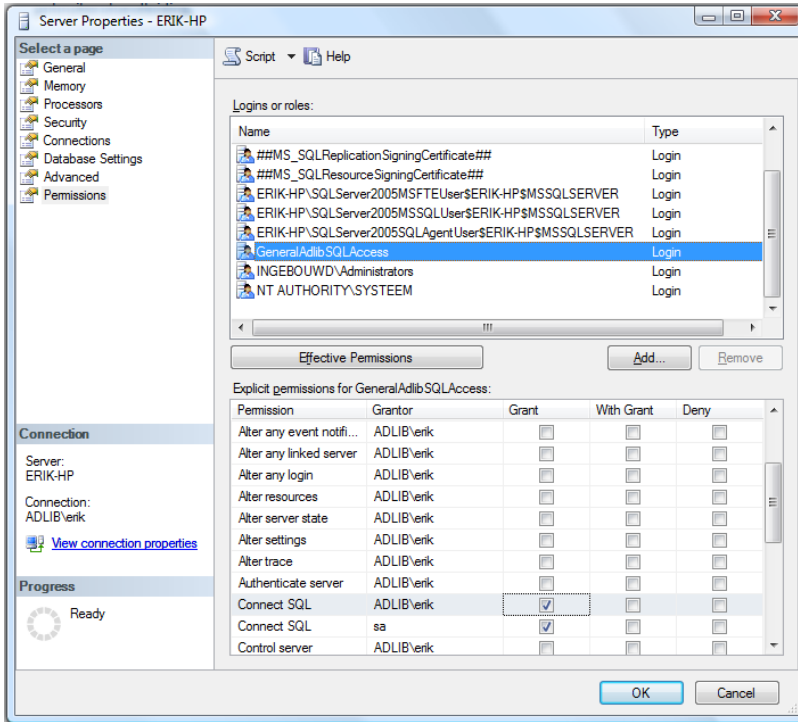
- Now check whether the login settings for the SQL Server are correct. In Microsoft SQL Server Management Studio Express, right-click the SQL Server name, and choose *Properties* in the pop-up menu which opens.



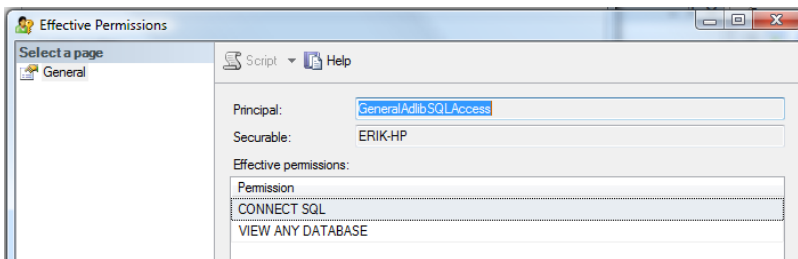
- Select the *Security* page and mark the *SQL Server and Windows Authentication mode* option, if that hasn't been done yet. (This is sometimes also called *mixed mode*.)



- Open the *Permissions* page to be able to check the access to the SQL Server for the new login. Select your login in the *Logins* list, and in the list below it mark at least the *Grant* permission for *Connect SQL*, but you may assign more rights if you wish.

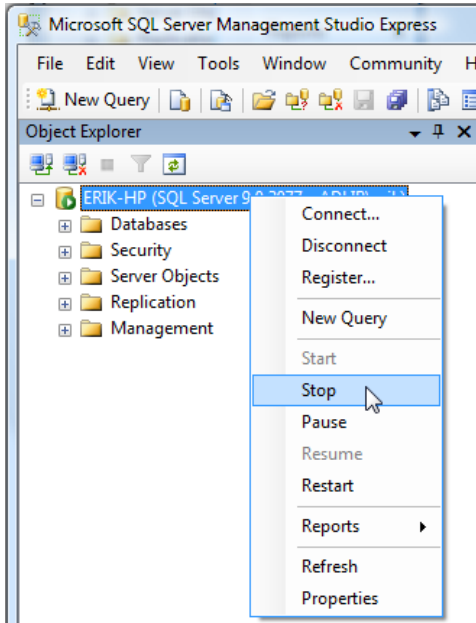


If you click the *Effective permissions* button, you can see which rights users with this login actually have.

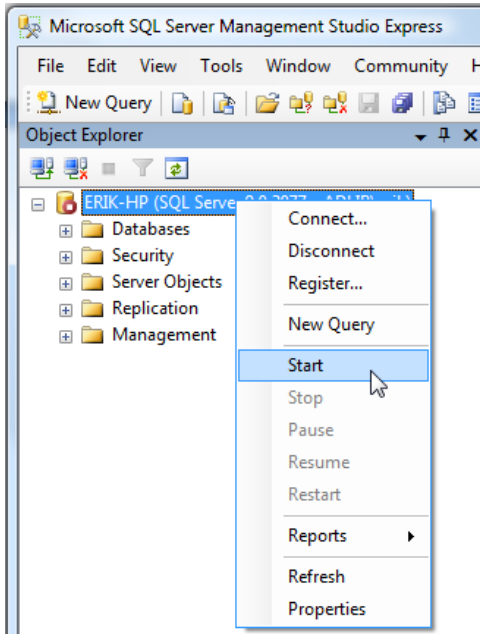



Click *OK* to close this window, and click *OK* again in the *Server Properties* window to store the changes.

7. Because you have changed settings for the SQL Server (from *Windows Authentication mode* to *SQL Server and Windows Authentication mode*), the server needs to be restarted. First make sure nobody is currently working in the database. Then stop the server by right-clicking the SQL Server and choosing *Stop* in the pop-up menu.

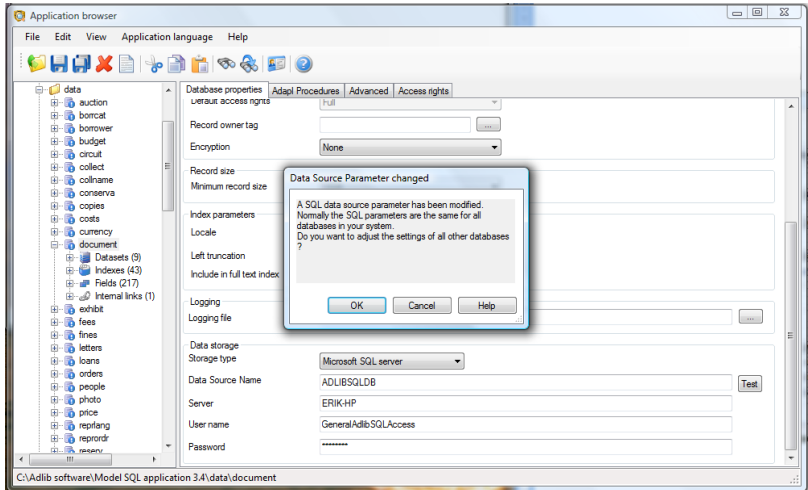


When the server has stopped, this is indicated by a red icon in front of the server name. Right-click the server name again and now choose *Start* in the pop-up menu.



If the icon turns green , it means the server is up and running again. Microsoft SQL Server Management Studio Express can now be closed.

8. Start Adlib Designer, and in the Application browser open the folder in which the *.inf* files of your Adlib SQL database are located. Select a random *.inf* file, for instance that of *DOCUMENT*. For the *User name*, enter the login name which you defined in SQL Server, in our example: `GeneralAdlibSQLAccess`. For the *Password*, enter the password which you provided for this login in SQL Server. Every time you leave one of these two field, Designer will ask you if you want apply this change everywhere; click *OK* in both occasions. This means you don't have to manually repeat your settings for the other *.inf* files. Now, save all changed files. Note that we assume here that you've already set the *Storage type*, *Data Source Name* and *Server* options on this tab correctly. If not, then do that now (for all Adlib databases).



Then click the *Test* button behind the *Data Source Name* entry field to test the connection with the SQL Server. If the connection is successful, the text *OK* appears above the button.

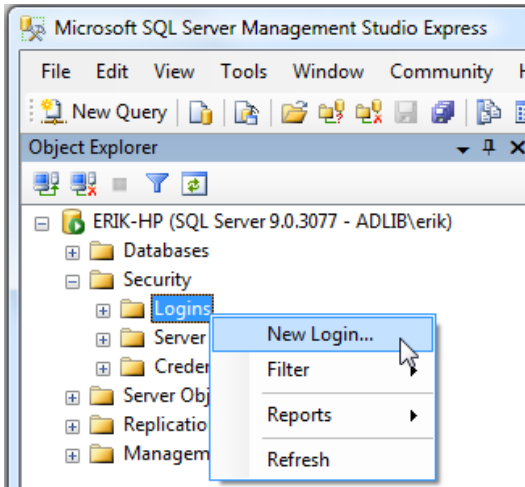


If anything goes wrong, you'll be so notified and the red text *ERR* appears above the *Test* button. Then check your settings on this tab, and the settings in the SQL Server.

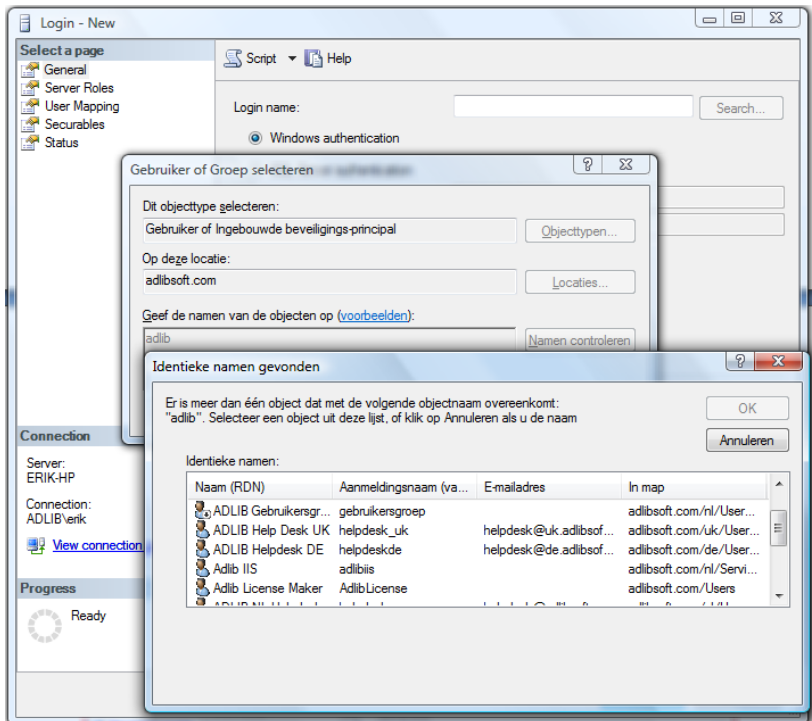
All Adlib users, and any other users who know the login name and password, now have full (dbo: database owner) access rights to the SQL database. That is probably undesirable. Therefore, use the different internal Adlib mechanisms to set access rights for individual users. See the Designer help for more information about this.

5.2 Windows authentication with Active Directory

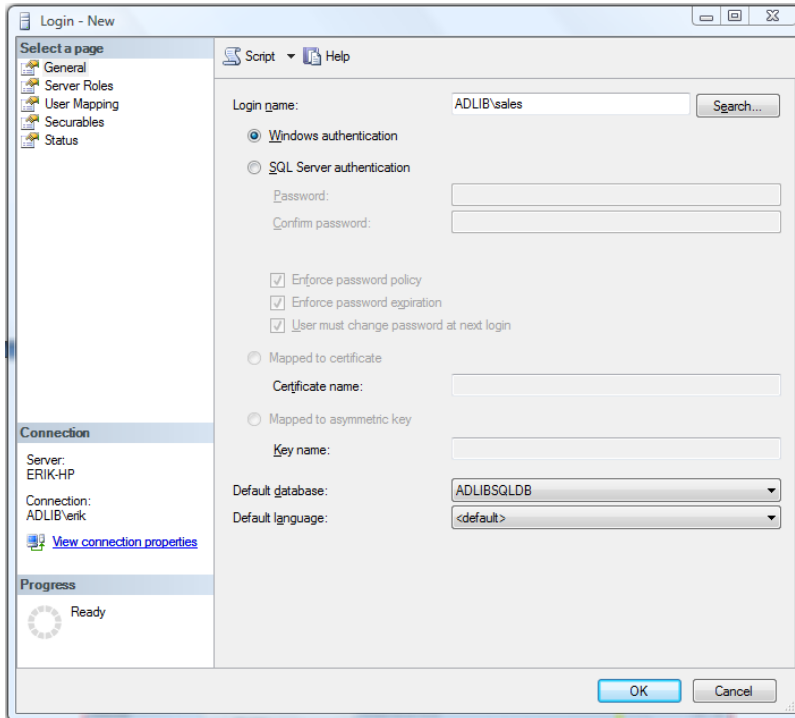
1. Divide all Adlib users in Active Directory into groups, so that in SQL Server only groups need to be entered and assigned access rights, instead of having to do that for each individual user. For example, you can put together groups for users who are only allowed to view data (e.g. trainees and visitors), for users who are allowed to view, edit, enter and delete (e.g. registrars and librarians), and for users allowed to manage the database (structure).
2. Start Microsoft SQL Server Management Studio Express, if that hasn't been done yet, open the *Security* folder underneath the SQL Server folder and right-click *Logins*. In the pop-up menu which opens, click the *New Login...* option.



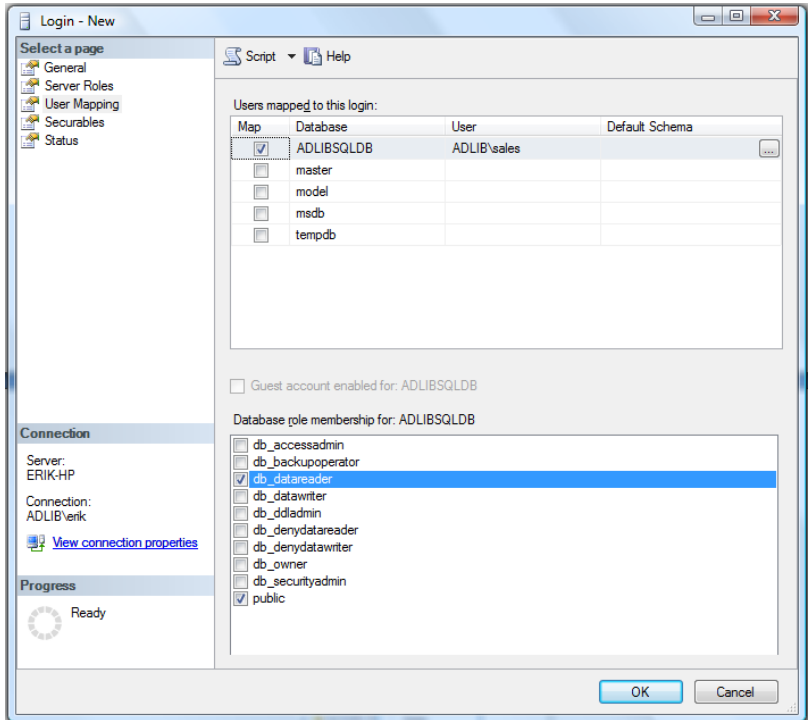
- In the *Login – new* window, click the *Search* button to be able to select an Active Directory user group. First, the *Select a user or group* window opens. In it, click the *Locations* button and select the network the Adlib users are part of, *adlibsoft.com* in our example. In the *Enter the names of the objects* field, enter the partial or whole name of a user group which you would like to set as login, and click the *Check names* button. The *Identical names found* window opens if the entered name is not yet correct. In this window, select the desired user group and click *OK*. Also click *OK* in the *Select a user or group* window.



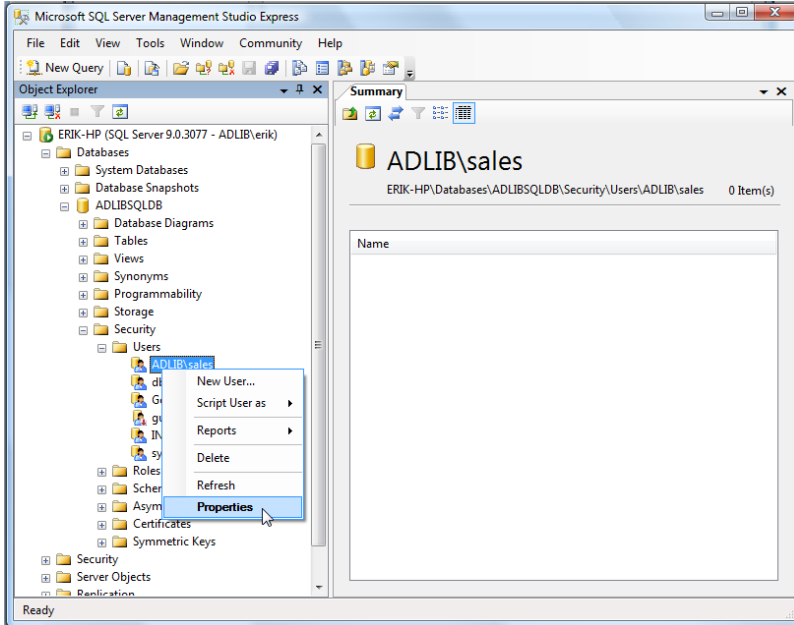
4. In this example we chose the *ADLIB\sales* user group. We are now creating an SQL Server login with the same name. On this page, choose your Adlib SQL database as the *Default database*, in this example that happens to be *ADLIBSQLDB*.



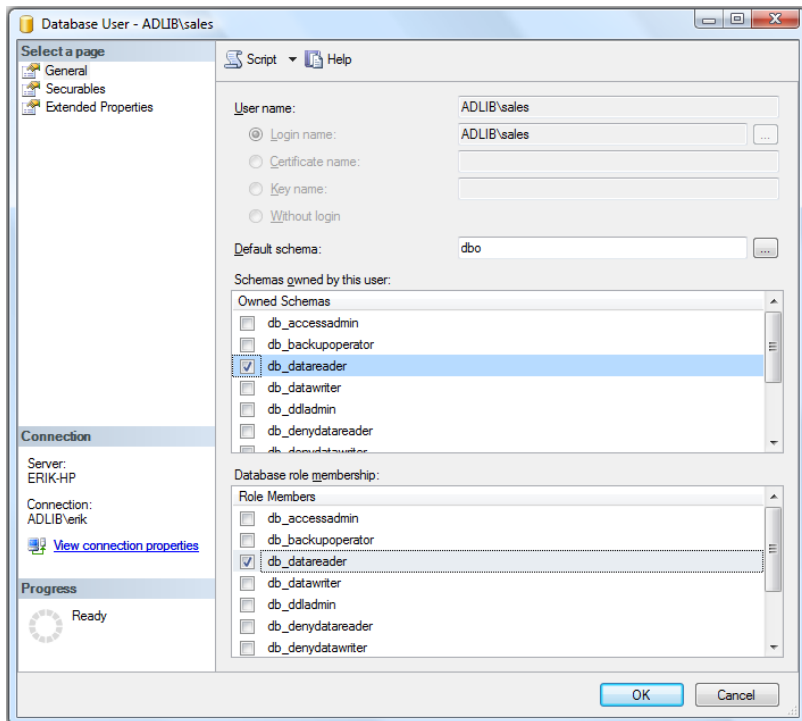
5. Leave the *Server Roles* to *public*, and proceed directly to the *User mapping* page – in the top left of the current window you select a page. On this page in the upper list, mark your Adlib SQL database, and in the list below it, mark the role(s) you want to assign to the current login, in this example: *db_datareader* (so that this user group may only view data, not edit it). Leave the *public* role marked by default. Click *OK* to close the window.



6. In the *Object Explorer*, now open your Adlib SQL database, with it the *Security* folder and subsequently the *Users* folder. Right-click the user group you just added, *ADLIB\sales* in this example, and choose *Properties* in the pop-up menu which opens.

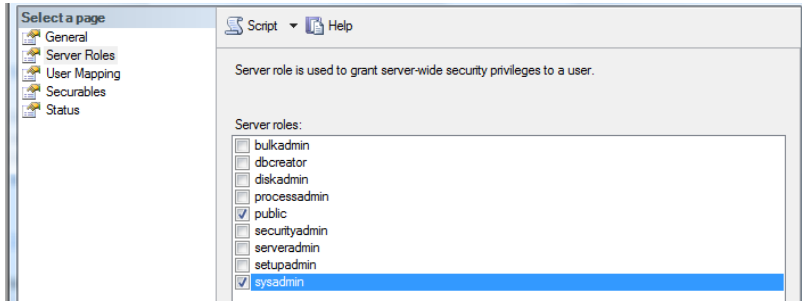


- Here, mark the desired schema for this user: it should be the same as the database role(s) marked in the list below it: *db_datareader* in this example. The Active Directory user group has now been added as an SQL Server user, with read-only access rights.



- Repeat this procedure (the steps 2 up to and including 7) for the other Active Directory users or user groups and assign the desired access rights to everyone of them. Note that if you assign the *db_datawriter* role, you should also assign the *db_datareader* role.
- Also add at least one Active Directory user, probably yourself, who gets the *db_owner* role as SQL Server user. For this user, set the *Default database* in step 4 to *master* (all database together): this in case there is more than one SQL database which you should be allowed to manage (for instance when a copy of your live Adlib SQL database has been made, for testing purposes). And in step 5 you now do open the *Server Roles* page to assign the database owner the *sysadmin* role as well. So, in the *User Mapping* you not

only mark the *public* role, but the *db_owner* role too; here, you can also select the databases which may actually be managed by this user. In step 7, assign the *db_owner* schema to this user.



- Now you can close Microsoft SQL Server Management Studio Express. Open Adlib Designer to test the connection between Adlib and the SQL Server. In the Application browser, open the folder in which the *.inf* files of your Adlib SQL database are located, and select a random *.inf* file, for example that of *DOCUMENT*. The *User name* and *Password* can be left empty, because logging onto the SQL Server is now done with the Active Directory login. Note that we assume here that you've already set the *Storage type*, *Data Source Name* and *Server* options on this tab correctly. If not, then do that now (for all Adlib databases).

 A screenshot of the 'Data storage' configuration dialog box. It contains several fields: 'Storage type' is a dropdown menu set to 'Microsoft SQL server'; 'Data Source Name' is a text box containing 'ADLIBSQLDB' with a 'Test' button to its right; 'Server' is a text box containing 'ERIK-HP'; 'User name' and 'Password' are empty text boxes.

Then click the *Test* button behind the *Data Source Name* entry field to test the connection with the SQL Server. If the connection is successful, the text *OK* appears above the button. If anything goes wrong, you'll be so notified and the red text *ERR* appears above the *Test* button. Then check your settings on this tab and the settings in the SQL Server.

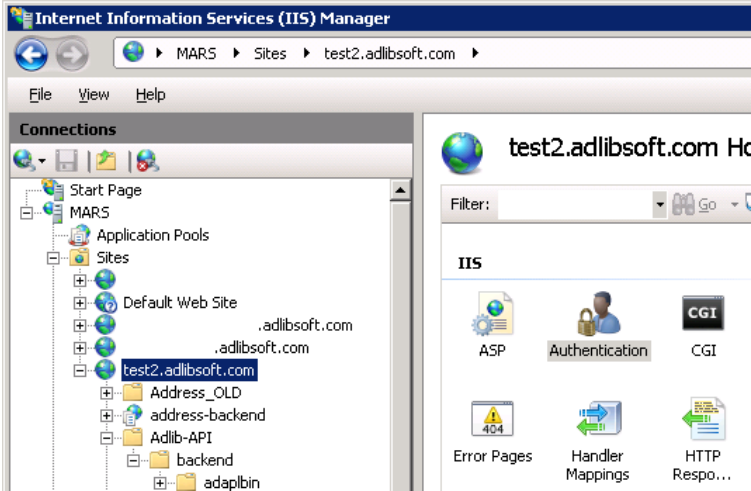
All Adlib users can now access the SQL database with their Active Directory user name and their own Windows password, with the access rights as defined for their login in SQL Server. This probably protects your database enough, but you can always still use the different internal Adlib mechanisms to refine the access rights for individual users. Do make sure that no conflicting access rights are set this way: this can of course lead to unexpected situations and confusion. You could keep an overview of SQL Server rights and Adlib access rights assigned to users. See the Designer Help for more information about access rights on Adlib level.

5.3 Excluding internet users from specific records

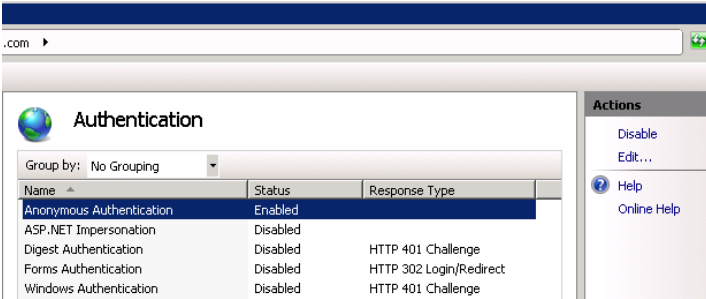
When you have an Adlib Internet Server web application, you may not just want to limit general database access for anonymous internet users to read-only access, but you may also want to prevent certain records completely from showing up in a search result on the website. You set this up as follows:

5.3.1 Setup in IIS 7 (Windows Server 2008)

1. Create a new Active Directory account which you will be using as the account under which any anonymous internet user will access your website after this setup, if that doesn't exist already. This account must have read access to the share(s) and files on the server(s) on which the web application, the *.infs* in the *\data* folder and SQL Server database are located. Choose a handy name for the new account, like "anonymous" or "internetuser". (See: <http://support.microsoft.com/kb/322684> for some information about how to create an Active Directory user account.)
2. From the *Administrative tools*, open the IIS Manager and in it select your website. Double-click the *Authentication* icon on the right.



3. Select *Anonymous Authentication* and click *Edit...* under *Actions*.

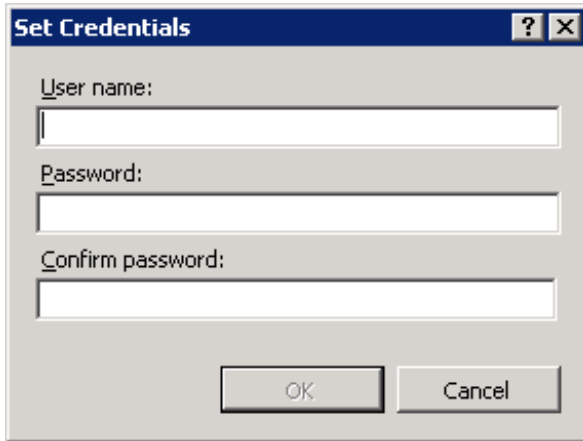


4. Mark the *Application Pool identity* option, and click *OK*.



5. In the left window pane, select *Application pools*, then select the application pool for your website and click the *Advanced settings...* option in the right window pane.

7. In *User name*, enter the account name you created earlier, preceded by your domain and a backslash (e.g. `ADLIB\anonymous`), and in *Password* and *Confirm password* enter the password you assigned to it. Click *OK*.



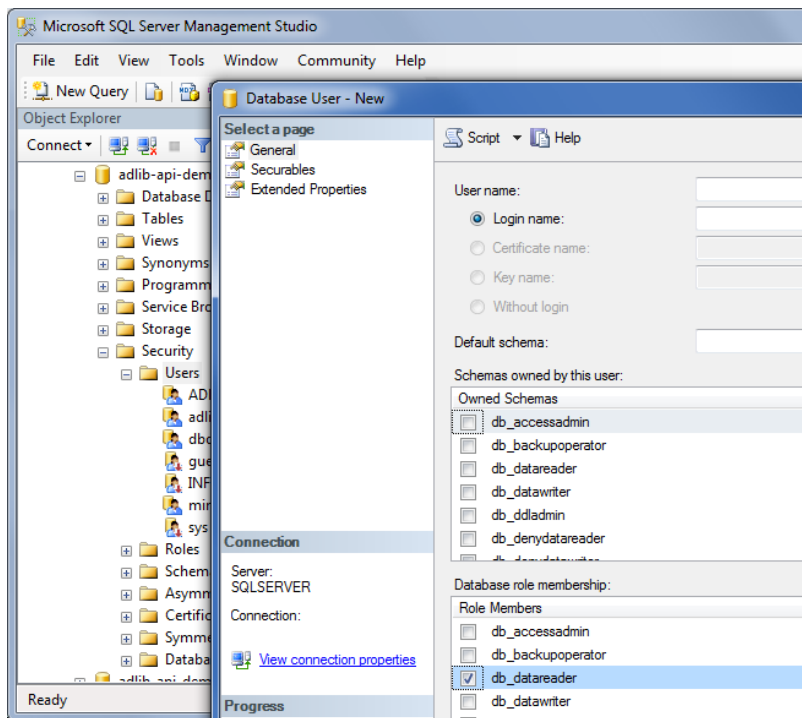
The image shows a Windows-style dialog box titled "Set Credentials". It has a blue title bar with a question mark icon and a close button (X). The dialog contains three text input fields: "User name:", "Password:", and "Confirm password:". At the bottom, there are two buttons: "OK" and "Cancel".

8. Click *OK* in the three open windows and close IIS.

5.3.2 Further setup in SQL Server Management Studio

If you use Windows authentication to handle access to the SQL Server database, you'll have to set the new account name as a user in that database, with read-only rights and possibly write access as well if users must be able to make reservations or add comments and tags to records.

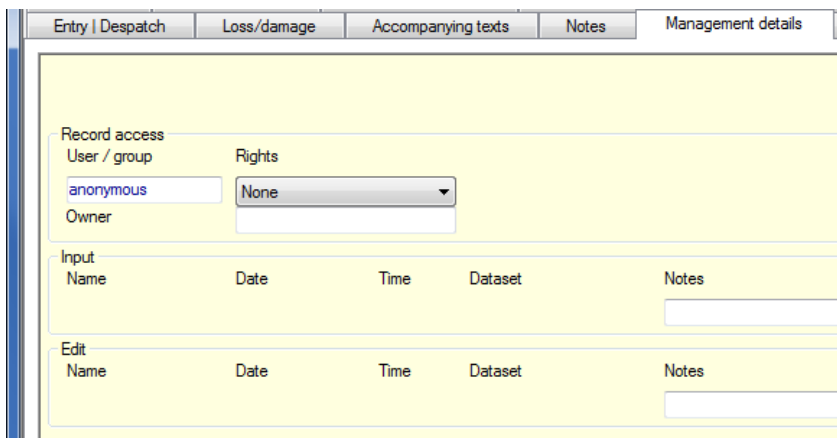
So, open Microsoft SQL Server Management Studio, select the relevant database and under *Security*, right-click the *Users* node and choose *New user*. In the *Database user – New* window, enter the account name you created earlier in both the *User name* and the *Login name* entry field, preceded by the proper domain and a backslash. In the *Database role membership* list at the bottom of the window then mark the *db_datareader* role, before clicking *OK*.



If you are using SQL Server authentication instead of Windows authentication, then the new Active Directory user account already has read-only access to the SQL Server database. So you don't need to add it as a user in the database.

5.3.3 Excluding records via the Adlib application

Assuming you have already set up the record access rights functionality in your Adlib (adlwin.exe) application, you can now use the anonymous internet user account name you created (without domain this time) to exclude records. The user must enter this name in Adlib records and possibly (depending on the setup) select the access rights *None*. If a record has *None* access rights for the relevant account it will now be excluded from wwwopac search results entirely. Note that only the record owner (who created the record) can enter or edit these access rights.



The screenshot shows a web application interface with a yellow background. At the top, there are several tabs: "Entry | Despatch", "Loss/damage", "Accompanying texts", "Notes", and "Management details". The "Management details" tab is active. Below the tabs, there is a "Record access" section. It contains two fields: "User / group" with the value "anonymous" and "Rights" with a dropdown menu set to "None". Below these fields is an "Owner" field. Underneath the "Record access" section, there are two tables. The first table is labeled "Input" and has columns for "Name", "Date", "Time", "Dataset", and "Notes". The second table is labeled "Edit" and has the same columns. Both tables have empty rows below them.

In the Designer Help, see the topic *General Topics > User authentication and access rights > Use the authorisation functionality* for information about how to set up record authorisation in your application.

In the example above the user name "anonymous" has been entered in the *User* field and in the *Rights* field *None* has been selected to specify that this user account has no access to this record. It is very well possible though that you or the application manager decided to implement record access rights slightly differently, in which case the user only needs to enter the anonymous internet user account name in the *User* field to automatically exclude these users from access to this record: in that case there is no *Rights* drop-down list to select other access rights from.

6 File types

Your sub folder for the Adlib Internet Server contains the following file types (amongst others):

.asax

For each application there is one file with the .asax extension. This file is called *global.asax* and contains code that can be executed per application and per session. The *global.asax* file defines, amongst others, variables that are used in the application as a whole. It also loads static objects such as stylesheets, once only per application. With the implementation of .NET, the source code is first compiled into a *.dll*, so *global.asax* itself does not contain code.

.aspx

These are .NET-specific files which look a lot like standard html files. In here you can specify the layout of the web application. A noticeable difference with html pages are the lines in between `<%` and `%>`. There you define which user controls you wish to use, and how the page should behave. Further you'll find normal html tags, and user controls defined by Axiell ALM Netherlands. The latter are similar to html tags, but their names have been specified by us.

web.config

This XML file is .NET-specific and contains configuration elements normally found in IIS, such as session time-outs, debugging, etc.

.css

A css (cascading style sheets) file determines the style of the elements (link, table, font, colour, background, margins, etc.) on a web page.

Anything in a css file can also be defined directly between the style tags in an HTML file. Preferably use a visual program for modifying a css file, such as Dreamweaver or Visual Studio.

.lic

You have an *adlib.lic* licence file, specific to your user agreement with Axiell ALM Netherlands (formerly Adlib Information Systems). To be able to use the Adlib software, this file (or copies of it) and the accompanying *adliblic.dll* should be present in all directories in which Adlib .exe files occur, normally the *bin* or *executables*, *tools* and *wwwopac* subfolders of your (web) application.

wwwopac.ashx

These is an executable program. The `wwwopac.ashx` is a .NET handler implementing the Adlib API. The program serves as an intermediary between a workstation and the Adlib search engine. It translates search queries from the web application into an Adlib search query, and executes them. `Wwwopac.ashx` is the only executable program that is necessary for Adlib Internet Server.

.xml

XML (Extensible Markup Language) is a standard with which data can be stored into a text file in a structured manner. Unlike an HTML document, an XML document provides information about the contents of the data, and not about its layout. So XML gives no information about presentation.

The sub folder `\Config` contains files with several settings for the web application. It can be used to determine the layout of forms, start-up settings and help texts.

Records that have been found with `wwwopac` are returned in XML. But the user on the internet will never see this XML, unless the web application is running in debug mode (if available), then these XML files can be made visible.

The `adlibweb.xml` file in the `\wwwopacx` folder initialises the `wwwopac`. When a search is started, this web configuration file is read so the Adlib search engine can use the correct conditions for the search. The `wwwopac` program uses the settings in the web configuration file to obtain information on where it should search, and in what way the result should be presented to the web user.

The web configuration file may contain both complete paths and relative paths.

.xslt

The transformation of an XML file (e.g. to HTML or XML in another format) can be specified with XSLT (Extensible Stylesheet Language for Transformations).

XML documents can be transformed with XSLT into HTML pages that can be viewed in a browser. This transformation can take place in two ways. The first way is by indicating in the XML file which XSLT file should be used for the transformation. When this file is retrieved in a browser (that supports XML transformation), the transformed document will be presented. This is called client-side transformation. We do not use this type in our Adlib web applications.

We do use the second way, which is called server-side transformation. The transformation now takes place on the web server. The transformed document can now be sent to the user. This way also allows for multiple small pieces of transformed XML to be used on the same page.

The subfolder `\Views\Results` contains files that specify the layout of forms and records. (Together with the css styles, all of the graphic design is determined.)

7 Adlib OAI Server

The OAI-PMH 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. In other words: if you would like to make one or more of your Adlib databases searchable and indexable for internet search engines, and you do not have a web application which is already handling that, then OAI is an interesting protocol for this purpose. Adlib *oai.ashx* supports OAI-PMH protocol 2.0. A full description of the OAI protocol can be found on <http://www.openarchives.org>

Through an OAI protocol request to a so-called repository (an installed Adlib OAI Server, in our case), (meta)data can be extracted in bulk from your specifically configured databases and consequently indexed, so that this data can be found by everyone on the internet via search engines.

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: in effect 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.

With the *adlibweb.xml* configuration file you determine which fields can be retrieved from an Adlib record (in Adlib XML format) and which XSLT stylesheet (with an Adlib field to Dublin Core element mapping) must be used 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.

The Adlib OAI Server (*oai.ashx*) implements this OAI-PMH protocol. This server has been implemented as a .NET http handler and can only be used for an Adlib SQL Server database, not for Adlib CBF databases (which are becoming obsolete).

We have put together a compact and relatively easy to install package, in which a lot of settings are ready-to-use (although you must edit some of them). In the package you'll find an *adlibweb.xml* file in which default settings for multiple Adlib databases have been made and some accompanying XSLT stylesheets with a mapping of Adlib fields to Dublin Core elements (see <http://www.dublincore.org/documents/dcmes-xml/>). So you don't have to make all this yourself. Little stands in your way anymore to start using OAI!

For a list of system requirements, see chapter 2 in this installation guide.

■ OAI functionality only in *oai.ashx*

Oai.ashx and *wwwopac.ashx* do not share any code and OAI functionality is only available in *oai.ashx*, not in *wwwopac.ashx*. However, the *adlibweb.xml* parameter file which is used already for the configuration of the *wwwopac.ashx* server can also be used for the configuration of the OAI server, but you may create a separate *adlibweb.xml* file just for *oai.ashx* as well. The OAI specific configuration is stored in an (optional) section with the name `<OAIConfiguration>`.

7.1 Installing *oai.ashx*: global setup

1. Go to the download page on our web site, to get the OAI package, if you haven't done that already, or request it through our helpdesk if the package is not being offered as a download.
2. E-mail our helpdesk with your organisation's details, for the password that is necessary to unpack the .zip file.
3. The Adlib OAI Server must be installed as supplementary program to an existing web server (IIS 6.0 or higher for Windows 2003 or 2008). No web application is needed to address the OAI Server.

Now, place the unpacked OAI package and configure IIS according to chapter 4.2 in this installation guide.

4. In the main OAI folder you'll find one *adlibweb.xml* file in which you must make all configuration settings. The file already contains examples of some database configurations that you can probably use as is, and a complete OAI configuration which you must adjust here and there to your own situation. You can view and edit the file in a text editor. Below you can see an example of the packaged configuration file:

```
<?xml version="1.0" encoding="UTF-8" ?>
<webConfiguration>
  <globalConfiguration>
    <databasepath>C:\Adlib\Model application\data</databasepath>
    <xmltype>grouped</xmltype>
  </globalConfiguration>

  <!-- ===== -->
  <!-- OAI Configuration -->
  <!-- ===== -->
  <OAIConfiguration>
    <OAI_REPOSITORY_NAME>Our Repository</OAI_REPOSITORY_NAME>
    <OAI_ADMIN_EMAIL>info@adlibsoft.com</OAI_ADMIN_EMAIL>
    <OAI_SETS>
      <OAI_SET>
        <SetSpec>collect</SetSpec>
        <Name>Default</Name>
        <Database>collect</Database>
        <SearchStatement>all</SearchStatement>
        <OAI_METADATAPREFIXES>
          <OAI_METADATAPREFIX>
            <Name>oai_dc</Name>
            <!-- use oai_dc34.xslt as stylesheet if your
              Adlib application version is 3.4 -->
            <StyleSheet>oai_dc.xslt</StyleSheet>
            <!--<Schema>http://www.openarchives.org/OAI/
              2.0/oai_dc.xsd</Schema> -->
            <!--<MetadataNamespace>http://www.openarchives.org/
              OAI/2.0/oai_dc/</MetadataNamespace>-->
          </OAI_METADATAPREFIX>
        </OAI_METADATAPREFIXES>
      </OAI_SET>
      <OAI_SET>
        <SetSpec>objects</SetSpec>
        <Name>Museum objects</Name>
        <Database>objects</Database>
        <SearchStatement>all</SearchStatement>
        <OAI_METADATAPREFIXES>
          <OAI_METADATAPREFIX>
            <Name>oai_dc</Name>
            <!-- use oai_dc34.xslt as stylesheet if your Adlib
              application version is 3.4 -->
```

```

<StyleSheet>oai_dc.xslt</StyleSheet>
<Schema>http://www.openarchives.org/OAI/2.0/
  oai_dc.xsd</Schema>
<MetadataNamespace>http://www.openarchives.org/OAI/
  2.0/oai_dc/</MetadataNamespace>
</OAI_METADATAPREFIX>
</OAI_METADATAPREFIXES>
</OAI_SET>
<OAI_SET>
  <SetSpec>books</SetSpec>
  <Name>Books</Name>
  <Database>document</Database>
  <SearchStatement>all</SearchStatement>
  <OAI_METADATAPREFIXES>
    <OAI_METADATAPREFIX>
      <Name>oai_dc</Name>
      <!-- use oai_dc_book_34.xslt as stylesheet if your
        Adlib application version is 3.4 -->
      <StyleSheet>oai_dc_book.xslt</StyleSheet>
      <Schema>http://www.openarchives.org/OAI/2.0/
        oai_dc.xsd</Schema>
      <MetadataNamespace>http://www.openarchives.org/OAI/
        2.0/oai_dc/</MetadataNamespace>
    </OAI_METADATAPREFIX>
  </OAI_METADATAPREFIXES>
</OAI_SET>
<OAI_SET>
  <SetSpec>adlib</SetSpec>
  <Name>Museum objects</Name>
  <Database>collect</Database>
  <SearchStatement>all</SearchStatement>
  <OAI_METADATAPREFIXES>
    <OAI_METADATAPREFIX>
      <Name>oai_adlib</Name>
      <!-- use oai_adlib.xslt as stylesheet if you want
        adlibXML as output -->
      <StyleSheet>oai_adlib.xslt</StyleSheet>
      <Schema>http://www.adlibsoft.com/adlibXML.xsd</Schema>
      <MetadataNamespace>http://www.adlibsoft.com/adlibXML/
        </MetadataNamespace>
    </OAI_METADATAPREFIX>
  </OAI_METADATAPREFIXES>
</OAI_SET>
</OAI_SETS>
</OAIConfiguration>

<databaseConfiguration database = "thesaurus">
  <database>thesau</database>
  <brieffields>
    <field>*</field>
  </brieffields>
  <detailfields>
    <field>*</field>
  </detailfields>

```

```
</databaseConfiguration>

<databaseConfiguration database = "collect" default="true">
  <database>collect</database>
  <brieffields>
    <field>title</field>
    <field>description</field>
    <field>creator</field>
    <field>dimension.type</field>
    <field>dimension.value</field>
    <field>dimension.unit</field>
  </brieffields>
  <detailfields>
    <field>*</field>
  </detailfields>
</databaseConfiguration>

<databaseConfiguration database = "objects">
  <database>collect<int>intern</database>
  <brieffields>
    <field>*</field>
  </brieffields>
  <detailfields>
    <field>*</field>
  </detailfields>
</databaseConfiguration>

<databaseConfiguration database = "document">
  <database>document</database>
  <brieffields>
    <field>*</field>
  </brieffields>
  <detailfields>
    <field>*</field>
  </detailfields>
</databaseConfiguration>

<databaseConfiguration database = "people">
  <database>people</database>
  <brieffields>
    <field>*</field>
  </brieffields>
  <detailfields>
    <field>*</field>
  </detailfields>
</databaseConfiguration>

</webConfiguration>
```

■ The <globalConfiguration> section

The <databasepath> node contains the physical path to the location where the Adlib .inf files are stored for this server, and this is the only element in this section that is relevant to the OAI server.

■ The <databaseConfiguration> sections

Just like for wwwopac.ashx, each <databaseConfiguration> section describes an available database/table to this service and the fields which can be retrieved from it. For OAI only the <database> and <detailfields> nodes are relevant (although the fields could also be specified in the <brieffields>). The database attribute of the <databaseConfiguration> node specifies an alias for the actual table name (the name of the .inf file in which an Adlib database is defined) in the <database> node beneath it. It's possible to specify that only a certain dataset inside the Adlib database must be accessible, by stating that dataset behind the database name (separated by an escaped greater-than sign: >;), for example: <database>collect>;intern</database>. Also, only one <database> node per <databaseConfiguration> section is allowed. An alias (which must be unique amongst the other aliases) can be different from the table name, so you can create multiple configurations for the same Adlib table if you wish, for instance to retrieve different fields when using wwwopac.ashx. You must then use this alias in the <OAIConfiguration> section to refer to the relevant table. Click [here](#) for more information about the discussed parameters.

■ The <OAIConfiguration> section

- <OAI_REPOSITORY_NAME> contains a human readable identification of the OAI repository. This name will be returned when an *Identify* request (aka "verb") has been submitted. The element can contain any text.
- <OAI_ADMIN_EMAIL> is used to provide the harvester with the e-mail address of the technical contact person for the server. This address will also be returned when an *Identify* request has been submitted.
- <OAI_SETS> contains a list of the sets which you want to make available through your OAI server. A set is an instant selection of records, created by predefined search statement in a particular Adlib database. A set, identified by its *setSpec* name, is typically retrieved in its entirety by a harvester using the *ListRecords* and *ListIdentifiers* verbs and the *set* argument. Within the <OAI_SETS> section there needs to be at least one <OAI_SET> present. The

first one in the list acts as the default set: this set is returned to the harvester if no *setSpec* was provided in the harvest query.

- `<SetSpec>` determines the name of the set. This is the name the harvester can (optionally) specify in the *set* argument of the *ListRecords* and *ListIdentifiers* verbs.
- `<Name>` may contain a human readable name for the set: it has no meaning for the server or the client.
- `<Database>` must contain the alias of one of the configured databases, as specified in the `database` argument of a `<databaseConfiguration>` element. Each `<OAI_SET>` section can contain only one `<Database>` element and because each alias in a `<databaseConfiguration>` needs to be unique, this implies that an OAI set cannot cross Adlib database (aka Adlib SQL table) boundaries. The name in the `<Database>` element must also appear in a `<databaseConfiguration>` element in the current *adlibweb.xml* file.
- `<SearchStatement>` contains an [Adlib search statement that is supported by the API](#), to form this set. For example, to select all records in a database this statement can simply be: `all`. Note that the `<SearchStatement>` can also contain a `pointer <nnn>` statement (replace `<nnn>` by the pointer file number), making it compatible with the older oaiserver.exe implementation.
- `<OAI_METADATAPREFIXES>` contains a list of supported OAI metadata formats, each within its own `<OAI_METADATAPREFIX>` element. At least one metadata format needs to be present for each set. The list of metadata prefixes that is returned by the *ListMetadataFormats* verb is taking its data from these lists. Multiple `<OAI_METADATAPREFIX>` elements can be specified in the `<OAI_METADATAPREFIXES>` section and the same `<OAI_METADATAPREFIX>` might occur in multiple `<OAI_SET>` sections (as would typically be the case for the *oai_dc* metadata format. In that case, the different sections might use different stylesheets.
 - `<Name>` must contain the name of the metadata format. You can enter any name. The harvester has to use this name in the *metadataPrefix* argument of a request.
 - `<StyleSheet>` must contain the name of an actual XSLT stylesheet which will be used to transform the initially

generated AdlibXML format to the requested format. See paragraph 7.2 for more information about how these stylesheets are constructed. The stylesheet file should be present in the main folder of the *oai.ashx* handler. If no stylesheet is supplied, the data is passed through in adlibXML format.

- `<Schema>` provides the URL to where the client can find a schema to validate the data that has been returned by the server. The Adlib implementation of the server does not do anything with this information except passing it to the client in the response to the *ListMetadataFormats* verb. The `<Schema>` element is not mandatory, but the OAI consortium strongly recommends supplying a schema for each metadata format.
- `<MetadataNameSpace>` specifies the URI for the namespace that is used to return the metadata elements. As with the `<Schema>` element, the Adlib OAI-PMH implementation currently does nothing with this element, except for passing it to the client in the *ListMetadataFormats* response.

See chapter 7.4 for information about testing an OAI Server based on *oai.ashx*.

7.2 MetadataFormats

The OAI-PMH protocol does not prescribe a specific format for the exchange of data (other than that it is supposed to be valid XML and should fit in with the XML framework that is returned by the server). This means that for instance library records could be expressed in MARC-XML, museum objects as LIDO and archive records as chunks of EAD. Having said that, the Dublin Core set of metadata elements is often seen as the “lowest common denominator” and every OAI-PMH implementation is at least supposed to be able to serve up records in Dublin Core format. In addition to this, it is considered to be good practice to use fully qualified XML tags for the return data. This means that the XML elements in the returned XML need to have a namespace prefix and this namespace prefix needs to be mapped to a URI determining the semantic meaning of the element. Here is an example of an unqualified XML element: `<title>` while the following is a fully qualified XML element: `<dc:title>`

To explain to the client that the prefix `dc` means “Dublin Core” a name

space URI needs to be provided. In the case of Dublin Core this is specified by the following XML attribute:

```
xmlns:dc=http://purl.org/dc/elements/1.1/
```

The OAI-PMH consortium decided that the Dublin Core format can be requested using the `metadataPrefix=oai_dc` argument. The server is free to implement any other metadata formats which it deems fit. The client has a mechanism to ask the server which metadata formats it supports (see paragraph 7.6.2).

Adlib databases deliver their data in the adlibXML schema. To transform this to Dublin Core, a mapping of fields needs to be created and subsequently an XSLT stylesheet needs to be created to implement this mapping. Obviously for the different types of data in an Adlib environment different mappings (and thus stylesheets) need to be created. This also depends on your preferences.

The Adlib implementation of the OAI-PMH protocol can deliver multiple metadata formats; if this is a requirement, multiple XSLT stylesheets need to be in place to support those formats. The basic principles on how to construct such a stylesheet are discussed in the next chapter.

7.3 Creating an XSLT stylesheet for Adlib OAI-PMH

If the default stylesheets are not what you want, you'll need to create the desired stylesheet yourself. Providing a complete manual on how to create a stylesheet for Adlib applications is beyond the scope of this document, but this paragraph is trying to get you jump-started. XSLT is short for eXtensible Stylesheet Language Transformations and can be used to transform one XML format to another format (again XML or something else, like HTML). To perform such transformations, XSLT makes use of so-called templates. These templates match elements or groups of elements and attributes of an XML node tree and then produce data in the form of new elements and attributes in the required format. XSLT itself is written in XML.

■ Adlib Grouped XML

The Adlib OAI-PMH server generates what is called "grouped" XML from the raw Adlib data. The structure of grouped Adlib XML is that of the example below:

```
<?xml version="1.0" encoding="UTF-8"?>
<adlibXML>
  <recordList>
    <record priref="10" created="2007-02-07T14:40:36"
      modification="2010-03-23T11:35:54" selected="False">
```

```

<acquisition.date>1816</acquisition.date>
<administration_name>PDP</administration_name>
...
<Technique>
  <technique>painting</technique>
  <technique.notes />
  <technique.part />
</Technique>
<Title>
  <title>Venus and Cupid</title>
</Title>
</record>
</recordList>
<diagnostic>
  <hits>1</hits>
  <xmltype>Grouped</xmltype>
  <first_item>1</first_item>
  <search>preref Equals 10</search>
  <sort></sort>
  <limit>1</limit>
  <hits_on_display>1</hits_on_display>
  <response_time unit="mS" culture="nl-NL">24</response_time>
  <xml_creation_time unit="mS" culture="nl-NL">792
  </xml_creation_time>
  <link_resolve_time unit="mS" culture="nl-NL">593
  </link_resolve_time>
  <dbname>collect</dbname>
  <dsname>intern</dsname>
  <cgistring>
    <param name="database">objects</param>
  </cgistring>
</diagnostic>
</adlibXML>

```

The main element (or *root* element) of this XML format is the `<adlibXML>` element. Within this element there are two other elements: the `<recordList>` element and the `<diagnostic>` element. The `<recordList>` element contains a list of `<record>` elements, the latter containing a single Adlib record each. The `<diagnostic>` element contains information about the search against the database, amongst which the number of records that were found and some timing information.

For the purpose of OAI-PMH we are mainly interested in the `<recordList>` element and its records. These will need to be transformed to another form of XML using our stylesheet. Let's view the structure of a single record:

```

<record preref="10" created="2007-02-07T14:40:36"
modification="2010-03-23T11:35:54" selected="False">
  <acquisition.date>1816</acquisition.date>

```

```

<administration_name>PDP</administration_name>
...
<Technique>
  <technique>painting</technique>
  <technique.notes />
  <technique.part />
</Technique>
<Title>
  <title>Venus and Cupid</title>
</Title>
</record>

```

Within the record elements we will find two types of sub-elements: simple fields and grouped fields. The element names are the same as the (English) field names in the Adlib data structure. Grouped fields are repeatable fields that repeat together in a group. The XML element name that bundles these fields together is the same as the group name in the Adlib data structure. Note that this implies that the field and group names in the Adlib datadictionary need to be compliant with the rules for XML elements. This means that these names cannot contain spaces, for instance.

■ The XSLT framework

It is the task of the XSLT stylesheet to transform the Adlib grouped XML to the required output format. To do this, we can use a fairly standard framework of templates:

```

<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:oai_dc="http://www.openarchives.org/OAI/2.0/oai_dc/"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
>
  <xsl:output method="xml" indent="yes"/>

  <xsl:template match="adlibXML">
    <xsl:apply-templates select="recordList"/>
  </xsl:template>

  <xsl:template match="recordList">
    <xsl:element name="recordList">
      <xsl:apply-templates select="record"/>
    </xsl:element>
  </xsl:template>

  <xsl:template match="record">
    <xsl:element name="targetRecordTag">
      <xsl:apply-templates select="fieldOrGroupName"/>
      <xsl:apply-templates select="fieldOrGroupName"/>
    </xsl:element>
  </xsl:template>
...

```

```

    </xsl:element>
</xsl:template>

<!-- Group matches -->
<xsl:template match="groupName">
  <xsl:apply-templates select="fieldName"/>
</xsl:template>

<!-- Field matches-->
<xsl:template match="fieldName">
  <xsl:element name="targetFieldTag">
    <xsl:value-of select="."/>
  </xsl:element>
</xsl:template>
</xsl:stylesheet>

```

The first section tells the system that this XML is actually a stylesheet and that we are using the prefix `xsl:` to indicate XSL instructions. The namespace declaration for XSL is mandatory and the URI for the namespace should be spelled correctly otherwise the XSL cannot be executed. The stylesheet element can contain other namespace declarations as well, particularly for prefixes that we intend to use in our stylesheet. In the above example there are two additional namespaces declared: `oai_dc` and `dc` (respectively for making `oai_dc` tags and Dublin Core tags possible).

The `<xsl:output>` element tells us that we are going to generate XML using this stylesheet and that we want it to be nicely indented; we could have switched the indentation off, which would have made no difference for the result except that the data would be less readable for the human reader, but smaller.

The next three templates match the `adlibXML` structure down to the record level. In the template that matches the record element we apply further templates for the groups and fields that we want to output. For each group of fields that we want to output we apply a separate template, in this example.

One should replace the black `targetRecordTag` by the tag that you want to have appear in the output, for instance `oai_dc:dc`. The same applies to the field and group names and the tags of the target fields (all listed in black). So an actual stylesheet could look like this:

```

<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:msxsl="urn:schemas-microsoft-com:xslt"
  exclude-result-prefixes="msxsl"
xmlns:oai_dc="http://www.openarchives.org/OAI/2.0/oai_dc/"
xmlns:dc="http://purl.org/dc/elements/1.1/">

```

```

<xsl:output method="xml" indent="yes"/>

<xsl:template match="adlibXML">
  <xsl:apply-templates select="recordList"/>
</xsl:template>

<xsl:template match="recordList">
  <xsl:element name="recordList">
    <xsl:apply-templates select="record"/>
  </xsl:element>
</xsl:template>

<xsl:template match="record">
  <xsl:element name="oai_dc:dc">
    <xsl:apply-templates select="Title"/>
    <xsl:apply-templates select="Description"/>
    <xsl:apply-templates select="Production"/>
    <xsl:apply-templates select="Dimension"/>
  </xsl:element>
</xsl:template>

<!-- Group matches -->
<xsl:template match="Title">
  <xsl:apply-templates select="title"/>
</xsl:template>

<xsl:template match="Description">
  <xsl:apply-templates select="description"/>
</xsl:template>

<xsl:template match="Production">
  <xsl:apply-templates select="creator"/>
</xsl:template>

<xsl:template match="Dimension">
  <xsl:element name="dc:format">
    <xsl:apply-templates select="dimension.type"/>
    <xsl:apply-templates select="dimension.value"/>
    <xsl:apply-templates select="dimension.unit"/>
  </xsl:element>
</xsl:template>

<!-- Field matches-->
<xsl:template match="title">
  <xsl:element name="dc:title">
    <xsl:value-of select="."/>
  </xsl:element>
</xsl:template>

<xsl:template match="description">
  <xsl:element name="dc:description">
    <xsl:value-of select="."/>
  </xsl:element>
</xsl:template>

```

```

<xsl:template match="creator">
  <xsl:element name="dc:creator">
    <xsl:value-of select="."/>
  </xsl:element>
</xsl:template>

<xsl:template match="dimension.type">
  <xsl:value-of select="."/>
  <xsl:text> </xsl:text>
</xsl:template>

<xsl:template match="dimension.value">
  <xsl:value-of select="."/>
</xsl:template>

<xsl:template match="dimension.unit">
  <xsl:text> </xsl:text>
  <xsl:value-of select="."/>
</xsl:template>
</xsl:stylesheet>

```

7.4 Testing and validating your OAI repository

To test your *oai.ashx* server locally, do the following:

- Start an internet browser from the computer on which you installed the Adlib OAI Server, and enter the following URL: `http://localhost/<your application alias>/OAITest.htm` and enter the name of the IIS application you set for the OAI server instead of `<your application alias>`. The *OAITest.htm* file that came in your OAI package contains some example requests which yield results in our demo database. Just click one of the hyperlinks to execute the underlying OAI request. The *Get record* example for instance, executes the following request in the current (`http://localhost/<your application alias>/`) directory (as can be seen in the *Address* box of the browser): `oai.ashx?verb=getRecord&identifier=collect:100&metadataPrefix=oai_dc`
This request may not produce a result in your database if there exists no record with record number 100 in *collect*, or you receive an error message if *collect* or *oai_dc* do not appear in your setup. Then adjust the URL in the *Address* box to your own circumstances (or edit the entire *OAITest.htm* file to your own setup) to get verifiable search results. (If you want, you can compare your search results with those of our demo installation on: <http://test2.adlibsoft.com/oai/OAITest.htm>.)

- You can also execute the *OAITest.htm* file (or the OAI requests directly) from other desktops when your server has been set up in a network environment. Start the browser from a workstation that's in the same network as the server on which you installed the Adlib OAI Server. Enter the URL from the previous step again, but replace `localhost` by the name of the server in the network or the ip-address of the server, for example `http://local_server/<your application alias>/OAITest.htm` **OR** `http://192.168.0.3/<your application alias>/OAITest.htm`.

To test your *oai.ashx* server via the internet, do the following:

- For instance, go to <http://dl.cs.uct.ac.za/projects/re/> to download software to test OAI locally.
- See <http://www.openarchives.org/Register/ValidateSite> to have your repository validated automatically. If validation succeeds, you can also officially register the repository here as well, so that it can be found and harvested more easily. All other registered repositories can be found [here](#) too.

7.4.1 What if every OAI request generates a login error?

The problem is probably located in the user authentication of the SQL Server database, if you are using Windows authentication. The application pool under which your oai server runs, has an *Identity* (which can be observed or edited in IIS when you have selected the relevant application pool). This identity is used to log in to the database in your SQL server. So that identity must have been set as a *login* for that server and also as *user* with that login name for the relevant database, wherein that user must have `db-datareader` rights in the *Database role membership* list. See chapter 5 for more information about user authentication.

7.5 OAI repositories and search engines

Whether and how internet search engines search OAI repositories, differs per search engine, and usually the manner and extend of indexing is classified information. To increase the chances of having your OAI repository indexed as much as possible, there are a couple of things you can do:

- If you own a website 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 website won’t be searched/indexed.
- 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 *oai.ashx*.
- On: <http://www.oclc.org/oaister/> you can submit your repository to OAIster, after which it will start 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.

7.6 How to harvest an OAI repository

To be able to test your own OAI repository in detail or to be able to harvest repositories of third parties, you need to know a little bit about the OAI-PMH protocol (see <http://www.openarchives.org>). In this paragraph and the following paragraphs you’ll find an introduction to get started quickly. This text is geared towards using the OAI-PMH protocol in the Adlib *oai.ashx* server.

7.6.1 Basic principle

The purpose of an OAI server is to serve records from a repository. This repository can be divided into so-called “sets” which are nothing but a collection of records. The client or the harvester can request records of a certain set. To do this, the client provides a `setSpec`. In addition to this `setSpec` the harvester can also provide “from” and “until” dates. The repository will then provide records that have been modified in the given time span. The `from` and `until` arguments are optional. (A repository declares the “granularity” in which the time span can be expressed. This can be in days, or even to the second precise.) Further, a protocol request by means of a `verb` must be provided to specify the type of the desired search result: a single record, a list of records, a list of record identifiers, a list of available metadata formats, a list of available sets or a request to have the server identify itself. Finally, the harvester can provide a `metadataPrefix` specifier to request the data in a certain form of XML.

So here is a summary of the five available parameters:

- `set` = name of the set to be harvested;
- `verb` = a protocol request to specify the search result type;
- `from` = earliest editing date (and optionally time) of records which must be harvested;
- `until` = latest editing date (and optionally time) of records which must be harvested;
- `metadataPrefix` = name (alias) of the form of XML in which the records should be delivered.

7.6.2 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 query.

The six available requests are summed up underneath. Detailed information about their syntax and their responses can be found on <http://www.openarchives.org/OAI/openarchivesprotocol.html>. (Requests are case-sensitive.)

GetRecord

This verb is used to extract an single 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 (record numbers) of all records that are available in the requested set. Optional arguments enable selective gathering, for instance based on the date they were last modified, or as a part of a specific set.

In the OAI world every record has a unique identifier. In Adlib each record is identified by a combination of the Adlib database (table) alias and its unique record number (prirref), e.g.: `collect:100` or `document:123`.

ListRecords

This verb is used to gather all the records available in the requested set. Optional arguments enable selective gathering, for instance based on the date they were last modified.

ListMetadataFormats

This verb is used to retrieve a list of metadata formats supported by the repository.

ListSets

This verb returns a list of available sets in the repository.

When the `verb` parameter is entered in the function call of `oai.ashx`, the server will process the query as an OAI-request. The syntax of a query and the ways you can enter them are the same as for a normal queries to the `wwwopac`, with the difference that other arguments must be used. To make an Identify-request, for example, use the following syntax:

```
...wwwopac.exe?verb=Identify
```

For example:

```
http://test2.adlibsoft.com/oai/oai.ashx?verb=Identify
```

The `Identify` value provides information about the relevant OAI repository, such as the repository name (e.g. the name of your collection), the baseURL to address queries to, the protocol version that is supported, the e-mail address of the repository administrator, and any additional information.

A standard OAI-protocol request has at least one parameter/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 parameter/value pairs depend on the arguments for the specific protocol request, e.g.:

```
http://test2.adlibsoft.com/oai/oai.ashx?verb=GetRecord&
identifier=100&metadataPrefix=oai_dc
```

```
http://test2.adlibsoft.com/oai/oai.ashx?verb=GetRecord&
identifier=collect:100&metadataPrefix=oai_dc
```

for a `GetRecord`-call of the record with identifier `collect:100`. Since in this case `collect` is the default set, just providing the record number `100` suffices as well.

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, then with the `metadataPrefix` parameter you specify (indirectly) which stylesheet should be used .

7.6.3 Harvesting in batches

It is up to the server to determine how many records are served per request maximally. The default value in the Adlib OAI implementation is 100 records per request. Obviously a set can contain much more records than just these 100. So the protocol has to provide a mechanism for the client to request the next batch of records after it retrieved the first 100. To facilitate this functionality the server provides a so-called resumption token to the client that it can use to request the next batch. You can find this resumption token at the bottom of the XML search result. Resumption tokens take the form of an arbitrary string of characters. In the Adlib OAI implementation we decided to use a GUID (Global Unique Identifier) as a resumption token. Here is an example of an Adlib resumption token:

```
28b9b5c5-32c3-4751-a4f3-3393f0342aa7
```

The harvesting client cannot decode any information (or meaning) out of this string, all it can do is return it to the server to request the next lot of records.

The resumption tokens have a limited validity. This is typically 24 hours. Within the lifetime of the resumption token the client can use the same token to retrieve the same batch of records. This is very useful in situations where for instance through network failure a batch of records did not arrive at the client. If the harvester requests a set of records using a resumption token and it is still not at the end of the requested set, then the server will provide a new resumption token for the next lot of records. Resumption tokens have application scope, meaning that another client can access the set using the resumption tokens from another client. One could see this as a security risk, but since the set is completely open for harvesting anyway, it really is not a threat.

An example of a request containing a resumption token would be:

```
http://test2.adlibsoft.com/oai/oai.ashx?verb=ListRecords&
resumptionToken=5b11a800-0afc-4e2a-858b-738f0251b501
```

If you use a resumption token in a request, you must not provide a `metadataPrefix`, nor any other argument: the last used arguments are automatically applied again.