
Axiell Collections

Installation



Axiell ALM Netherlands BV

Copyright © 2024 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.

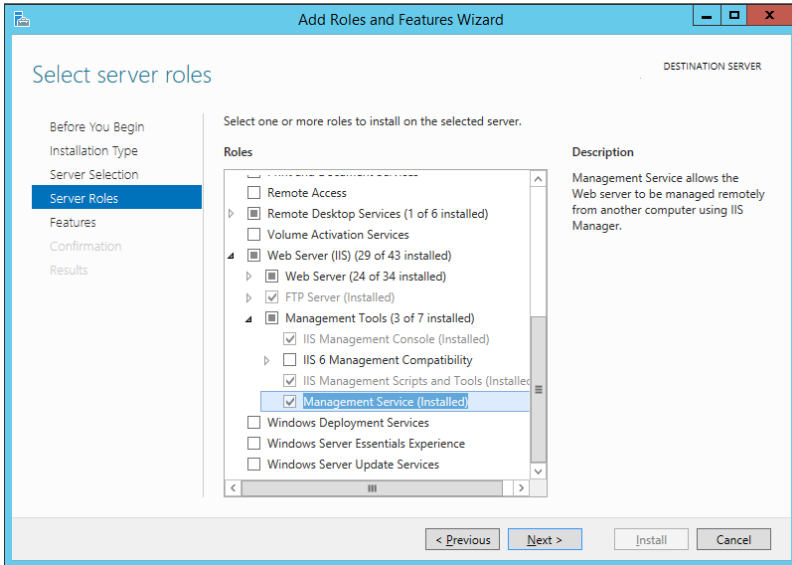
As a result of continuous improvements, later versions of the products may vary from those described here. Under no circumstances may this document be regarded as a part of any contractual obligation to supply software, or as a definitive product description.

CONTENTS

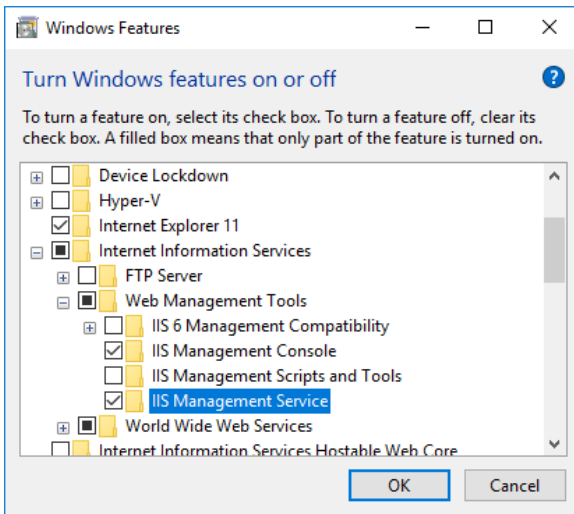
1 Requirements	1
2 Installing Axiell Collections	7
<i>Access rights</i>	24
<i>Login</i>	25
<i>Active Sessions</i>	25
<i>Maximum file upload and download settings</i>	30
Appendix A: installing Web Deploy	35
Appendix B: using SDI with smtp.office365.com	38
<i>Installation</i>	39
<i>Creating the output format</i>	41
<i>Scheduling SDI in the Windows Task Scheduler</i>	46
<i>Testing and possible errors</i>	47

1 Requirements

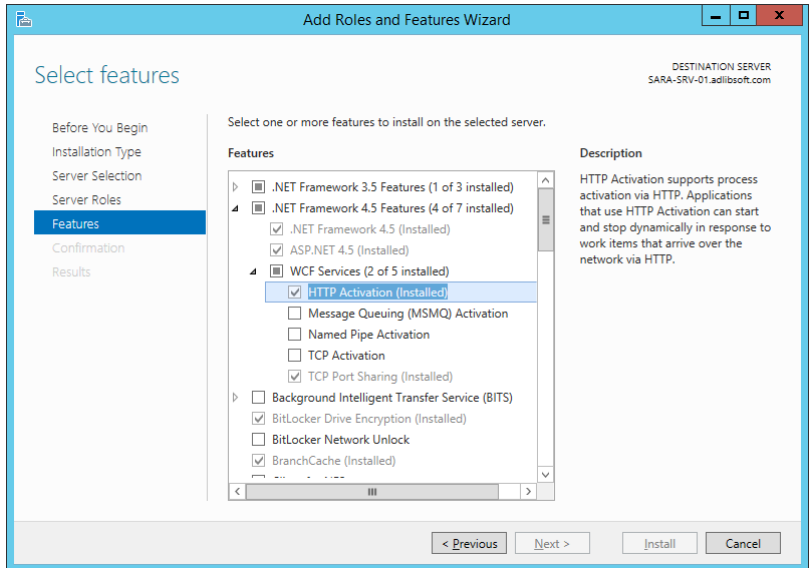
- Minimally a Windows Server or Windows version which is still supported by Microsoft too, within an Active Directory domain*, with at least 300MB free disk space.
- Some requirements differ depending on whether you are about to start using Axiell Collections in combination with Adlib or Calm:
 - **Adlib**: an Adlib application (with a subfolder *\data*). A further requirement is that the application data directory is actually accessible. If the application resides locally on a server, this accessibility is in principle not a problem. However, should it be located on a different server, then you'll have to check its access rights.
 - **Calm**: a Calm application version 11.0 or higher. The server version of Calm must be installed; the client version can also be installed but is not necessary. We recommend that security is enabled in the Admin program; otherwise, anybody with access to the web server could view or edit data. Ports to the Calm server have to be open through the firewall (as with CalmView) to allow users to access Collections.
- HTTP server software must be installed on the server on which the web application will be placed, such as IIS 7.0 for Windows Server 2008 or [IIS 8.5 for Windows Server 2012 R2](#). For the required Windows versions, these services are probably already available but might still have to be enabled by setting the *IIS Management console* and *Management service* options for the web server. To do so, open the *Server Manager* and on the *Dashboard* click the *Add roles and features* option. Then under *Server Roles*, look up the relevant *Web Server > Management tools* options and mark them if necessary.



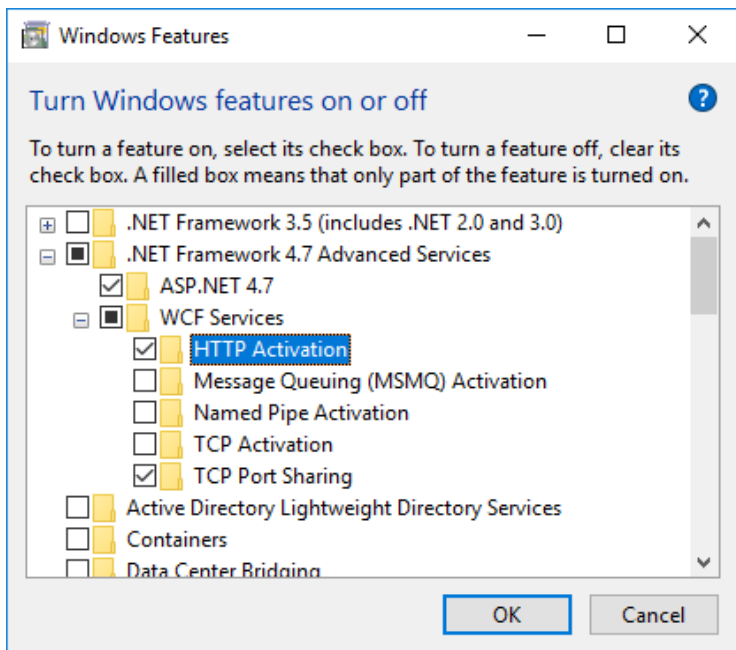
In Windows 10 you would do this via the *Windows Features* dialog (*Control Panel > Programs > Programs and Features > Turn Windows features on or off*).



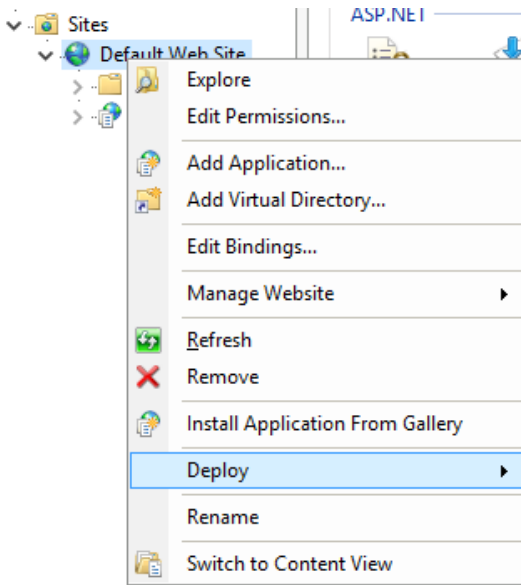
Further make sure that (under *Features*) for the .NET Framework the *WCF Services > HTTP Activation* option is marked.



In Windows 10 you would find it in the *Windows Features* dialog.



- The IIS *Web Deploy* extension should have been installed on the IIS Server. In IIS you can check whether the extension is already available or not, by right-clicking the *Default Web Site* to open the pop-up menu: if the *Deploy* option is present in the options list, the extension is present. If not, download and install the latest version (choose the 64-bit version for 64-bit Windows) from <https://www.microsoft.com/en-us/search/result.aspx?q=web+deploy>. Restart IIS after installation and check if the *Deploy* option is present yet. (If you've marked the web server *Management Service* option (see previous requirement) *after* installing *Web deploy*, you may have to run the *Web deploy* installation again and choose the *Repair* installation option. Restart IIS and check again.)



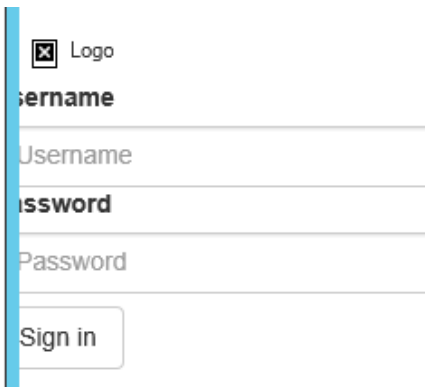
If the *Deploy* option still doesn't become available in the pop-up menu, then deinstall the extension you just installed and see the procedure in Appendix A.

- You can deploy Collections in any folder, but by default it will be deployed in `C:\inetpub\wwwroot` with the web application name you provide during the deployment. Before you can start deployment, do make sure you have full access rights to the desired target folder, otherwise the application cannot be deployed and you'll get an error message during deployment.

- The Microsoft .NET Framework version 4.8 (or higher, but 4.8 is currently the recommended 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 the .NET Framework, see: <https://msdn.microsoft.com/en-us/vstudio/aa496123>.
(If the .NET Framework 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 Axiell Collections application, must run in this mode.
- Running Axiell Collections requires a browser: Chrome, Firefox, Safari or Edge.
- A possible issue on Windows 2022 Server only (depending on what other software already has been installed on it) is that after installing Collections 1.19, when trying to run Collections, you'll get an "Error loading SqlServerSpatial160.dll (ErrorCode: 126)" error. This just means that the latest Microsoft Visual C++ Redistributable Version is not yet present on your server. Just install it from <https://learn.microsoft.com/en-us/cpp/windows/latest-supported-vc-redist?view=msvc-170> : select the proper executable for your architecture (so e.g. https://aka.ms/vs/17/release/vc_redist.x64.exe for X64) and install it. After that, Collections 1.19 should run without problems.
- Some requirements differ depending on whether you are about to start using Axiell Collections in combination with Adlib or Calm:
 - **Adlib**: if Active Directory authentication will be used for access to the database, instead of SQL Server authentication, then the application pool must be configured to use an account which has access to the SQL Server.
The easiest way to set up access rights for multiple users, is if you use SQL Server authentication and set the process model identity of your application pool to *NetworkService*. Then users can log in with their Windows user name, but on SQL database level all users will have access via one and the same SQL Server user name. Via the Adlib internal access rights mechanism (as can be set up through Adlib Designer), individual users can still be assigned differentiated access rights, but the one SQL user does have to have database owner rights because some user actions cause the creation of new SQL database tables (like the first time a user creates

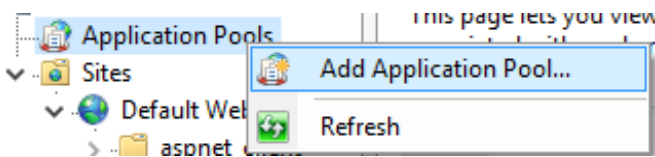
a saved search in a data source).
- **Calm:** <no similar requirement>

* Although not recommended, Axiell Collections, the Adlib application and the database server can also be installed in a peer-to-peer network, within a workgroup, in which case each computer/server in the workgroup needs to have its own (matching) user accounts and security control to be able to share their resources. In this case, the Collections web application folder and the Adlib application folders require sharing and the anonymous IIS_USR user needs to have read-only access rights to these folders. Another requirement is that in IIS the *Anonymous Authentication* for the *Collections* web application needs to be set to *Application pool identity* (instead of *Specific user*) to allow the user normal access to the web application: without this setting the login screen will be presented on a white page without proper layout (see screenshot below).

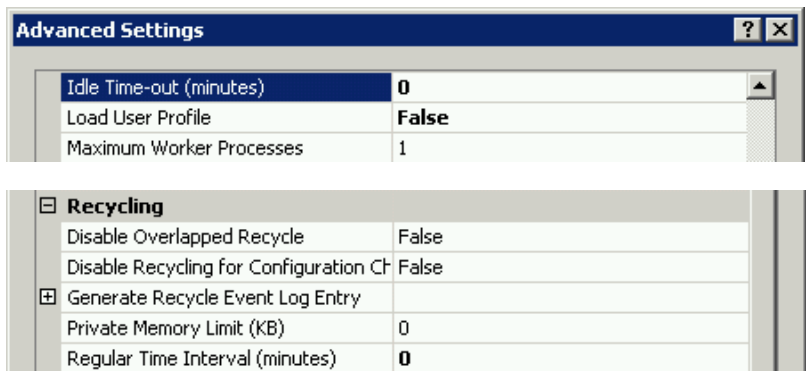


2 Installing Axiell Collections

1. You have received the Axiell Collections deployment package for a first time installation or an update of Collections (*AxiellCollections.zip*) from the Axiell ALM support desk. Place this file in a folder on the server you intend to use for Axiell Collections. Later during this procedure, the physical files of the web application will be installed in this folder if desired (name it appropriately in that case) or in *C:\inetpub\wwwroot*, depending on what you do in step 7.
2. For a first time installation only, open IIS on the server and create a new .Net v4.0 application pool, *AxiellCollectionsPool* for example, and leave the default basic settings (*Integrated* mode) as they are.

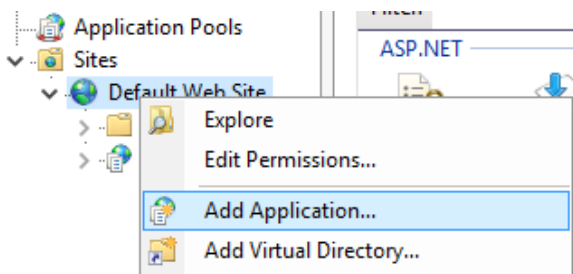


In the *Advanced settings* of the new application pool, either set the (*Process Model*) *Identity* to use the *ApplicationPoolIdentity* (*Built-in account*) or a custom (usually a general) Active Directory account, depending on your SQL server/network access rights. To increase performance on startup of Collections and to avoid having to log in again after leaving Collections idle for some time, also set the (*Process model*) *Idle time-out (minutes)* and (*Recycling*) *Regular time interval (minutes)* to 0: this avoids unnecessary recycling of the application pool.



Also for performance improvement on startup, set the *Start Mode* to *AlwaysRunning*: this launches the worker process for the application pool as soon as IIS is started instead of launching it only when the first request for the web application is received. This option is available from IIS 8.

3. We recommend using HTTPS as the data transfer protocol. You would first need to acquire an SSL certificate from your local certification organisation and install it on the server. (In IIS you can find any installed certificates when you select the server in the left window pane and double-click the *Centralized certificates* icon.) For a first time installation only, right-click your default website in IIS and select *Edit bindings* in the pop-up menu. In the window that opens, click *Add*, select the *https* binding type and then click the *Select* button to be able to select your SSL certificate. Next you'll have to redirect all HTTP requests to HTTPS, for example by using the (Collections 1.12) `<SecureConnection>true</SecureConnection>` option in *settings.xml* (see further down in this manual) or by using [URL Rewrite](#), a plugin for IIS. Information about how to set up a redirect can be found elsewhere: [here](#) for example.
4. For a first time installation only, in IIS, add a new application called `Collections` for example (**no spaces or dots allowed** in this name, otherwise you'll get an *HTTP error 405.0 Method not allowed* after logging in!), underneath the *Default Web Site*, assign it your new application pool and mark the *Enable preload* checkbox (available from IIS 8) to speed up performance.

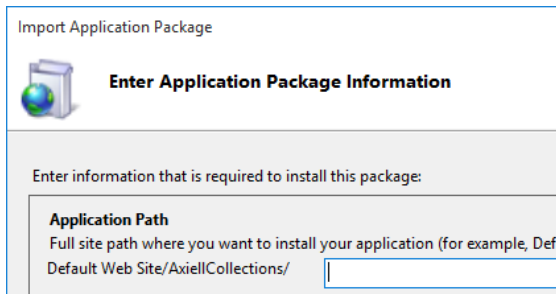


In the *Add application* window, use the *Test Settings* button to check whether a connection can actually be made.

5. Underneath the *Default Web Site*, your new application is now visible. (If you are updating an existing installation, you'll find your existing application here.) For both a first time installation, right-click the *Default Web Site* (if you'd like the physical files to be in-

stalled in `C:\inetpub\wwwroot\<my folder name>`) or right-click the application underneath the *Default Web Site* (if you'd like the physical files to be installed in the folder containing the deployment package) and select *Deploy > Import application*. When you are updating an existing installation of Axiell Collections, just right-click the application underneath the *Default Web Site*.

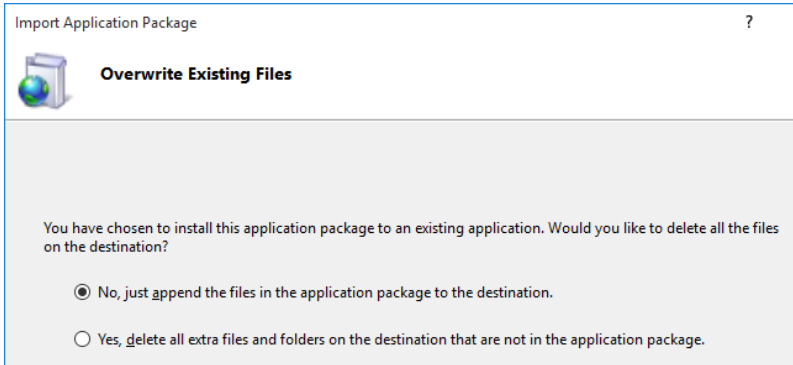
6. For both a first time installation and an update: in the *Import Application Package* wizard that has opened, *Browse* to your deployment package (*AxiellCollections.zip*) and open it. Click *Next*.
7. In the following step, leave the selected *Contents of the Package* as is and click *Next*.
8. For a fresh installation, in the *Application Path*, either remove any text from the entry field to deploy the application underneath the remaining fixed path (in which case the physical files will be installed in the folder containing the deployment package) or leave the suggested folder name in the entry field (or type a new one) to deploy the application there (in which case the physical files will be installed in `C:\inetpub\wwwroot\<my folder name>`). Click *Next*.



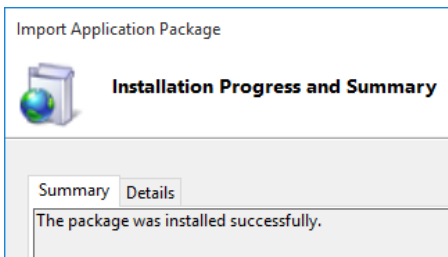
The screenshot shows a wizard window titled "Import Application Package". Below the title bar is a sub-header "Enter Application Package Information" next to a globe icon. A grey instruction box says "Enter information that is required to install this package:". Below this is a section titled "Application Path" with a text box containing "Full site path where you want to install your application (for example, Default Web Site/AxiellCollections/" and an empty input field.

In case you are just updating an existing application, remove the text from the entry field to have the files in the remaining fixed application path updated.

9. In the *Overwrite Existing Files* step, choose the first option if you already have an Axiell Collections installation in place on that location, which you'd only like to update: any custom settings in `\App_Data\settings.xml` remain as they were. Conversely, select the second option if this is a fresh installation and any files and folders in the target location can be deleted.



10. Click *Next* to start the actual deployment now. After installation has finished, the wizard should report a successful installation. Click *Finish*. (All following steps can be skipped if you have just updated an existing installation.)



11. For a first time installation only, in IIS, right-click your Axiell Collections application and select *Explore* in the pop-up menu to open the installation folder in Windows Explorer. Open the `\App_Data` folder.
12. In the `\App_Data` folder you'll have to create a `settings.xml` file if you've just installed Axiell Collections for the first time. To do this, just copy and paste the `settings.example.xml` file and rename the copy to `settings.xml`. The `settings.xml` file will have to contain some custom settings to tell the web application where to look for your application, amongst others. The `settings.example.xml` file contains an explanation for each possible setting, but some further explanation will follow below.
13. This step in the installation procedure differs depending on whether you are about to start using Axiell Collections in combination with Adlib or Calm:

- **Adlib**: open *settings.xml* in an appropriate editor (like Notepad++ for example).

Making sure that the Help is accessible

The Collections web server will always check if a *topicmapping.xml* file (required by browsers to open the Collections Help) is available at the specified location in the <Help> node (<https://help.collections.axiell.com/> by default) and will only offer the Help function in Collections if that check succeeds. Since this check will fail if your web server doesn't have internet access, you'll have to replace that URL by `~/App_Data/`. So the node literally becomes `<Help>~/App_Data/</Help>`. Then the system will look for and load *topicmapping.xml* from the `\App_Data` folder which also contains the *settings.xml* file you are currently editing. The *topicmapping.xml* file in this folder will be updated automatically with every Collections update.

Using a secure connection

The following recommended setting redirects all http requests to https (which is more secure) if set to true. It will then also add the Strict-Transport-Security header to all responses for better security.

```
<SecureConnection>true</SecureConnection
```

If the setting is true and the Collections request comes from insecure HTTP then the request will fail if the system can't redirect to HTTPS.

If it is false (the default if the SecureConnection element is missing –for compatibility with old installations) then insecure HTTP is allowed.

Using a singular Adlib application

In case of a default, singular Adlib installation where all users use the same application, you can leave `<SessionManagers Current="SQL">` as is. (The "SQL" id points to the `SessionManager Id` with the same name, so if you change the session manager id, you'll have to match it in the `SessionManagers Current` setting.), for example:

```
<SessionManager Id="SQL" Scope="Collections"  
  Assembly="Axiell.Alm.Database.Sql" Type="Axiell.
```

```

Alm.Database.Sql.SqlApplicationSessionManager">
  <Setting Key="Languages" Value="en,nl" />
  <Setting Key="CacheScripts" Value="false" />
  <Setting Key="Path" Value="C:\Adlib\xplus" />
  <Setting Key="ImageCacheFolder" Value="" />
  <Setting Key="Preload" Value="Full" />
  <Setting Key="DomainController" Value="" />
  <Setting Key="DomainContainer" Value="" />
  <Setting Key="DomainUsersOnly" Value="true" />
  <Setting Key="UseOriginalMediaFileName"
      Value="false" />
</SessionManager>

```

The optional `Scope` attribute for the `<SessionManager>` exists to differentiate front ends, since the Collections API can be used across multiple applications (Collections and Adlib Internet Server, for example). For a Collections application, set it to `Collections`. The option is relevant if indeed different front ends use this API and you don't want actions like the resetting of all user interface settings to their default (as can be done via the *Settings* button in the Collections main menu) to affect other front ends.

Note that with the current session manager set to "SQL", all domain users can access your application. To prevent that from happening you may either specify further (limiting) access rights, using the Adlib access rights mechanism, or set the current session manager to "Multi" and set up a matching multi-tenancy session manager to restrict login to a specific Active Directory group. Setting up multi-tenancy is described further down in this chapter.

Languages is an optional setting to improve performance when a user logs in for the first time. By default (without the setting), Collections attempts to load all interface languages (languages used for field and button labels and such) available in Adlib Designer, even if there are no actual interface translations for those languages. This automatic detection of languages leads to a substantial pause the first time a user logs in. To speed things up, include this setting with a list of comma-separated [ISO 639-1 language codes](#) for the interface languages that are needed by your users and are actually available in Axiell Collections too. (Invalid codes will be silently ignored by Collections.)

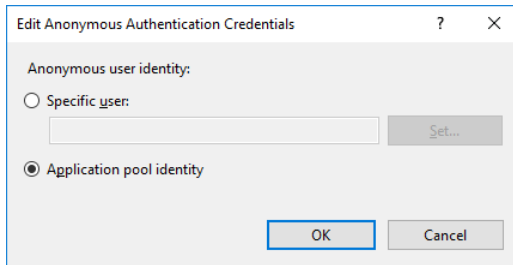
CacheScripts (set to `false`) is an optional setting for development purposes, which enables the reloading of an `adapl` bin file each and every time it is called, so that recycling of the application pool isn't required after making changes to the `adapl`. Leave

out this setting to increase performance.

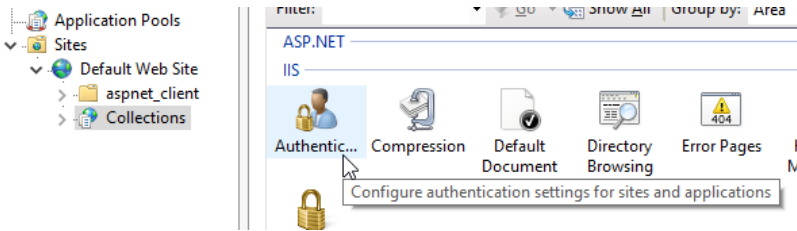
Path: the *Value* for the "SQL" session manager should be set to the actual, full path to your Adlib application folder. You can't use drive letters for shares: use the original path to the server in a format like: \\myserver\adlib\xplus. Instead of setting the path to a specific Adlib application folder (like *xplus*), you can also set it to the Adlib software root folder (containing multiple application folders, like *xplus*, *library*, *museum*, etc.) if you'd like to offer the user a choice of applications to log on to.

ImageCacheFolder: with the *ImageCacheFolder* setting set to a full local or UNC path* of an existing folder which Collections is allowed to use as a cache folder for resized image files. Every time images are displayed resized in Collections (for thumbnail display for example) a .bmp copy of the image in the displayed size(s) is stored in an application-specific, database-specific and field-specific subfolder of this folder so that next time the image must be displayed in the same size it can be retrieved directly without need for (time consuming) resizing.

* Note that the application pool user must have enough access rights to allow the implicit creation of a new subfolders and the writing of files in that folder. Check that the anonymous authentication credentials (which apply even before a user has logged in to Collections) have been set to *Application pool identity*.



You'll get there by first selecting your Collections application in IIS, then double-click the IIS *Authentication* icon, after which you right-click the *Anonymous Authentication* name and select Edit... in the pop-up menu.



Preload: the *Value* indicates the level at which to pre-load parts of the web application to reduce delays for users who are the first to log on to Collections: it concerns delays before the login dialog is displayed, before the *Select database* dialog and before a record is displayed after performing a search. You can assign one of four values: *None* (pre-load is switched off), *Basic* (pre-loads only database/data source info), *More* (pre-loads basic schema info and commonly-used objects such as screens) and *Full* (pre-loads everything, including adapls if required). For values other than *None*, Collections will load the relevant schema objects when the application pool starts, so delays will be concentrated before the login dialog is opened for the first time during a session. Note that the application pool user must have at least read access rights to the application for pre-loading to work.

Enabling this option isn't always the best choice, like for a large multi-tenancy system. It forces a preload of the application regardless of whether the tenant uses it and therefore increases baseline server memory and resource usage.

DomainController and **DomainContainer:** these settings are useful for tightening up security for local demo installations that are not connected to the AD domain. If a user has locked down their default Active Directory configuration, we need to be able to query non-default domain controllers and containers, for which you can use these settings.

`DomainController` can be an IP address in the format `<number>.<number>.<number>.<number>:<port>` (leave out the brackets, so a fictional example would be `123.456.789.012:636`) or a fully qualified domain name and should include the desired port number. Default ports:

389 - No SSL (Secure Sockets Layer), regular AD server

636 - SSL, regular AD server

3269 - The server is a Global Catalog server

`DomainContainer` is the name of the domain as a DC (Domain-Container) string. Fictional examples of two such strings:

`DC=456,DC=789,DC=012` or `DC=axiell,DC=com`. This determines

which domain/host (within the network) to bind to since an AD server can handle multiple domains.

DomainSSLEnable enables SSL (Secure Sockets Layer) support.

DomainSSLTrustCertificate should be set to `true` if you are using SSL and a self signed certificate. Set it to `false` if the SSL certificate comes from a trusted provider (in other words: if you paid for it).

DomainUsersOnly: with this option set to `false`, if a Collections user can't authenticate against Active Directory then Collections will automatically authenticate against local Windows accounts on the Collections server. Although the account will be on the Collections server (which should be secure), the behaviour may not be as expected and allows access to Collections that can't be controlled through Active Directory. Use this setting (`Value="false"`) for local demos that are disconnected from the domain, otherwise you can't log in. By default, the option is set to `true` to allow only domain users to log in, which is the more secure choice.

UseOriginalMediaFileName: with this option set to `false` (which is also the default when the option hasn't been set), after uploading a media file in an image or application field (often labeled *Reference*), the greyed out text <<New>> will appear in the field. At this point, the uploaded file only exists in a temporary holding area. Now, only when you save the record will the uploaded file actually be processed further: only then will the file either be copied to the file system or written to an optional Digital Asset Management System, depending on the file storage system used by your application, at which point the new, unique file name will be generated and stored in the record. By default, Collections will generate a GUID (Globally Unique Identifier) file name first, under which name the file will be stored in either the file system or a DAMS: in the case of the file system this initial GUID file name will become visible in the *Reference* field, while in the case of a DAMS, the DAMS may generate its own GUID (which is also associated with the stored file) and returns it to Collections after which that second GUID file name will be stored in the field. Instead of this default behaviour, Collections 1.7.19350.2 (and higher) offers the optional `UseOriginalMediaFileName` setting (set to `true`) to store an uploaded file under its original name. Then, uploading a file will still initially put the text <<new>> in the field, but upon storage of the record, Collections will store the file

under its original name, either in the file system or in a DAMS. If the DAMS returns a GUID file name (which will also be associated with the stored file), it'll be that name that will be saved in the *Reference* field. The file system however, will generate no GUID so the original file name will also be stored in the *Reference* field. If the file name of the uploaded file does already exist in the target location, that won't be a problem for a DAMS but the file system will return a warning saying *A media item with the id ... already exists*, after which the record is not saved, but left in edit mode. In the latter case you'll have to rename the file in Windows Explorer before you can try to upload it again in the *Reference* field in Collections - you may have to empty the field first if it still contains <<new>>.

Security (CSP)

Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross-Site Scripting (XSS) and data injection attacks. From 1.12.1, Collections supports the A+ HTTP observatory level of this policy to enhance the security of Collections. It basically sends an HTTP response header to the browser to tell it which content to allow and which content to block, like scripts on another domain. The *settings.example.xml* now sets the `Csp` option underneath the `Security` node to `enabled="true"`. You can copy these settings to your existing *settings.xml* file to profit from this extra security. Optional you can provide a list of trusted URLs (to websites or images, like a third-party IIIF image server or openstreetmap URLs you would use for a GIS implementation) in the `Whitelist` and/or `WhitelistImages` nodes as well, but typically you won't need to use these.

Logging

With `Logging` you can have a log file created and maintained for any errors and warnings reported by Collections. This may help our programmers to find out what the cause of a problem is. So you only need set the option really if issues occur while working with Collections and the Axiell Helpdesk asks for it. In the setting you should at least provide an existing path to a folder in which to create the log file. Optionally you may set `Warning`, `Error`, `Debug` and `Trace` to `true` or `false`, depending on how extensive you'd like the logging to be. Example:

```
<Logging Path="c:\Logs" Warning="true" Error="true"
Trace="false" Debug="false" />
```

`Error` designates real errors, causing Collections to stop working properly, while `Warning` is about situations which are potentially harmful to the normal operation of Collections. `Trace` and `Debug` provide verbose information, such as route trace logs, increasing the size of the log file substantially, so these should only be turned on only if requested. If you specify none of these log levels in the `Logging` setting, logging will default to `Error` level logging. The log file name defaults to `FrontEnd.txt`. This file may become very large though, which requires you to delete it once in a while. To prevent this file from becoming so large that it becomes unmanageable, Collections 1.12 and up allows you to provide an optional format string for the file name, to create a new logging file each day, month or year. Simply add the following string to the fixed part of the file name: `${date:format=yyyy-MM-dd}` to create a new file every day. (Leave out `-dd` or `-MM-dd` to create a new file only once every month or year respectively.) You can also provide an optional `MaxOldFiles` argument to specify how much log files to keep (of the most recently created ones) while purging older log files when creating a new one. If `MaxOldFiles` is omitted or 0, purging is disabled. Example:

```
<Logging Path="c:\Logs"
File="FrontEnd.${date:format=yyyy-MM-dd}.txt" MaxOldFiles="10" />
```

Multi-tenancy

In most cases you'll be using a singular Adlib application with a "SQL" `SessionManager`, which makes the above settings all you need. In the case of a singular Adlib application that needs Active Directory group protection at login or in the case of an enterprise multi-tenancy situation however (multiple instances of Collections running under the same web application), where users from different branches are allowed access to only one (or some) of many different applications (each with their own user list and access rights), the current session manager should be set to "Multi". Within this `SessionManager` definition you must then list all applicable Active Directory groups (without domain) as `Key` attribute with an accompanying, appropriate id as `Value` attribute. For example:

```
<SessionManager Id="Multi" Assembly="Axiell.Alm"
Type="Axiell.Alm.Database.MultiTenant.MultiTenantApplicationSessionManager">
  <Setting Key="ThesauManager" Value="All-app00" />
  <Setting Key="PersonsManager" Value="All-app00" />
```

```

    <Setting Key="Guest" Value="All-app00-G" />
    <Setting Key="app01users" Value="MuseumA-app01" />
    <Setting Key="app02users" Value="MuseumB-app02" />
    <Setting Key="app03users" Value="MuseumC-app03" />
    <Setting Key="app04users" Value="MuseumD-app04" />
    <Setting Key="app05users" Value="MuseumE-app05" />
</SessionManager>

```

The ids for the different user groups that you've now specified must then be defined as separate session managers too, so that any user can be associated with the appropriate application(s) in Axiell Collections. So, for example:

```

<SessionManager Id="All-app00" Assembly="Axiell.Alm.
  Database.Sql" Type="Axiell.Alm.Database.Sql.
  SqlApplicationSessionManager">
  <Setting Key="Path" Value="G:\Adlib\Collections\
    museums\All-app00\model\plusplus" />
</SessionManager>

```

```

<SessionManager Id="All-app00-G" Assembly="Axiell.Alm.
  Database.Sql" Type="Axiell.Alm.Database.Sql.
  SqlApplicationSessionManager">
  <Setting Key="Path" Value="G:\Adlib\Collections\
    museums\All-app00-G\model\plusplus" />
</SessionManager>

```

```

<SessionManager Id="MuseumA-app01" Assembly="Axiell.Alm.
  Database.Sql" Type="Axiell.Alm.Database.Sql.
  SqlApplicationSessionManager">
  <Setting Key="Path" Value="G:\Adlib\Collections\
    museums\MuseumA-app01\model\plusplus" />
</SessionManager>

```

```

<SessionManager Id="MuseumB-app02" Assembly="Axiell.Alm.
  Database.Sql" Type="Axiell.Alm.Database.Sql.
  SqlApplicationSessionManager">
  <Setting Key="Path" Value="G:\Adlib\Collections\
    museums\MuseumB-app02\model\plusplus" />
</SessionManager>

```

```

<SessionManager Id="MuseumC-app03" Assembly="Axiell.Alm.
  Database.Sql" Type="Axiell.Alm.Database.Sql.
  SqlApplicationSessionManager">
  <Setting Key="Path" Value="G:\Adlib\Collections\
    museums\MuseumC-app03\model\plusplus" />
</SessionManager>

```

```

<SessionManager Id="MuseumD-app04" Assembly="Axiell.Alm.
  Database.Sql" Type="Axiell.Alm.Database.Sql.
  SqlApplicationSessionManager">
  <Setting Key="Path" Value="G:\Adlib\Collections\
    museums\MuseumD-app04\model\xplus" />
</SessionManager>

```

```

<SessionManager Id="MuseumE-app05" Assembly="Axiell.Alm.
  Database.Sql" Type="Axiell.Alm.Database.Sql.
  SqlApplicationSessionManager">
  <Setting Key="Path" Value="G:\Adlib\Collections\
    museums\MuseumE-app05\model\xplus" />
</SessionManager>

```

Note that the application folder paths you set for a multi-tenancy configuration must always point to a specific Adlib application folder, like `\xplus` for example: you cannot point it to the Adlib software root folder to offer the user a choice of applications. This is because a user must have logged in first, for Collections to be able to determine which applications can be accessed by the user (so a list of available applications cannot be offered beforehand). If a root folder is specified anyway, the multi-tenancy session manager will automatically pick the first matching application in the referenced session manager.

If you're also using single sign-on, make sure the pbk's *Default access rights* are set to *Full* for all applications because these are checked too before users are allowed to access them (or get to select them in the first place if they have access to more than one application) after logging in: this poses no extra risk because the access rights are already checked via SSO and the multi-tenancy setup.

Setting up direct printing from within Collections

From version 1.18, Axiell Collections allows users to print the result of an output format directly to a selectable printer (instead of the result opening as a document in Word or the browser first). The new *Print* button and the following *Select printer* dialog will only be present though if available printers have been set up in the Collections *settings.xml* file first. This option can only be implemented if you have a local installation of Collections: Collections installations hosted by Axiell cannot access your local (on-site) printers (yet).

In *settings.xml*, insert a `<Devices>` section underneath the main `<Settings>` node with a `<Device>` node per printer, like so for instance:

```

<Devices>
  <Device Id="DefaultPrinter" Type="Printer">
    <Name>
      <Text lang="en">Default printer</Text>
      <Text lang="nl">Standaard printer</Text>
    </Name>
    <Settings>
      <PrinterName>\\ourserver\RICOH 5000</PrinterName>
      <PageLayout>
        <Orientation>Portrait</Orientation>
        <PageSize>A4</PageSize>
        <Size Width="210" Height="297" />
        <Margins Top="1" Bottom="1" Left="1"
Right="1" />
        <DefaultFont Name="Times New Roman"
Size="12" />
      </PageLayout>
    </Settings>
  </Device>
  <Device Id="Pdf" Type="Pdf">
    <Name>
      <Text lang="en">Print to PDF</Text>
    </Name>
    <Settings>
      <PrinterName>Microsoft Print to PDF</PrinterName>
    </Settings>
  </Device>
</Devices>

```

The available settings are the following:

Type: The type of the printer:

Pdf = Print to a PDF file (no PageLayout settings required)

Printer = Print to a printer

Device Id: Your unique ID for the printer. (This can be stored by Collections in Saved search jobs and in other places, so once used this should be retained.)

Name: Localised UI name of the printer for displaying to the user. This is what appears in printer selection dropdowns in the Collections user interface. Specify a name in all interface languages used by your Collections users. If no name hasn't been specified in the current interface language, then by default the English name will be displayed to the user.

PrinterName: The `PrinterName` is the name of the Windows printer driver. If the printer is a network printer (on a different server), you may need to specify the network share path to the printer, like `\\other-server\RICOH 5000`. If all printers on a server have the comment "(redirected)" behind them, it means they are redirected from your local pc during your active Remote Desktop Connection, although the printer name with the path in front of it also seems to work when no RDP connection is present. If you leave the `PrinterName` empty, it should choose the default printer driver.

PageLayout: The entire `PageLayout` section is optional, but if you include it, `PageSize` will default to A4 if not specified and `DefaultFont` is optional too. If `PageSize` is specified, `Size` can be omitted. Instead of a standard `Pagesize` you can also specify Custom as `PageSize` and then specify `Size`, `Margins` and `DefaultFont`.

Orientation: Portrait OR Landscape.

PageSize: Supported standard `PageSizes` are: Custom, A0, A1, A2, A3, A4, A5, RA0, RA1, RA2, RA3, RA4, RA5, B0, B1, B2, B3, B4, B5, Quarto, Foolscap, Executive, GovernmentLetter, Letter, Legal, Ledger, Tabloid, Post, Crown, LargePost, Demy, Medium, Royal, Elephant, DoubleDemy, QuadDemy, STMT, Folio, Statement.

The available printer page sizes don't come from the document that you are trying to print, but from the printer driver itself. Under the hood, the report will be rendered to PDF according to the page size that the printer supports. If you have different paper trays then each will need to be configured as a separate printer.

Size: When `PageSize` is Custom, specify the page dimensions in mm. Ignored otherwise.

Margins: Page margins in mm.

DefaultFont: Default font to use when the report doesn't specify a font. Used as defaults for text or HTML.

- **Calm:** open `settings.xml` in an appropriate editor (like Notepad++).

Using a singular Calm application

In case of a default, singular Calm installation where all users use the same application, change `<SessionManagers Current="SQL">` to `<SessionManagers Current="CALM">`. (This tells Collections to look at the "CALM" SessionManager section below. If you change the session manager id, you'll have to match it to the `SessionManagers Current` setting). The CALM SessionManager section looks like:

```
<SessionManager Id="CALM" Scope="Collections" Assembly=
"Axiell.Alm.Database.Idle" Type="Axiell.Alm.Database.Idle.
DScribeApplicationSessionManager">
  <Setting Key="Version" Value="11.0"/>
  <Setting Key="Server" Value="localhost"/>
  <Setting Key="Port" Value="2143"/>
  <Setting Key="User" Value="admin"/>
  <Setting Key="Password" Value="drum"/>
  <Setting Key="SecurityRequired" Value="true" />
</SessionManager>
```

The following Setting keys can be used to configure Collections:

Version: The version of Calm you're running (e.g. version 11.0).

Server: The server Calm is located on.

Port: The port used to reach Calm. This is required if you are not running on default ports.

User and **Password:** To access Calm, Axiell Collections needs the credentials of a user set up in the Calm Admin program as an Administrator. This can be done in two ways:

- The best way to supply those credentials is by entering them into the Application Pool Identity section in step 2, above. This must be an active directory user who appears in the Admin program as an Administrator.
- It is also possible to replace `admin` with the username and `drum` with the password of a Calm user. This does not have to be an active directory user—it can be a user created in the Admin program—and it does not have to be a user profile in use at your organisation. This is not secure, though, as this file is stored in plain text.

SecurityRequired: If this is `true`, Calm security must be enabled (in the Admin program) for users to log in to Axiell Collections. If this is `false`, any users could log in with any credentials—Calm security will

not be set up. For that reason, it is strongly recommended to only set this to false for certain kinds of testing.

Multi-tenancy

In most cases you'll be using a singular Calm application, which makes the above settings all you need. In the case of an enterprise multi tenancy situation however, where users from different branches are allowed access to only one (or some) of many different applications (each with their own user list and access rights), the current session manager should be set to "Multi". Within this `SessionManager` definition you must then list all applicable Active Directory groups (without domain) as `Key` attribute with an accompanying, appropriate id as `Value` attribute. For example:

```
<SessionManager Id="Multi" Assembly="Axiell.Alm"
Type="Axiell.Alm.Database.MultiTenant.MultiTenantApplicationSessionManager">
  <Setting Key="CalmALMusers" Value="CalmALM" />
  <Setting Key="CalmRMusers" Value="CalmRM" />
</SessionManager>
```

The ids for the different user groups that you've now specified must then be defined as separate session managers too, so that any user can be associated with the appropriate application(s) in Axiell Collections. So, for example:

```
<SessionManager Id="CalmALM" Assembly="Axiell.Alm.
Database.Idle" Type="Axiell.Alm.Database.Idle.
DScribeApplicationSessionManager">
  <Version>10.0</Version>
  <Server>localhost</Server>
  <Port>2143</Port>
  <User>admin</User>
  <Password>drum</Password>
</SessionManager>
<SessionManager Id="CalmRM" Assembly="Axiell.Alm.
Database.Idle" Type="Axiell.Alm.Database.Idle.
DScribeApplicationSessionManager">
  <Version>10.0</Version>
  <Server>localhost</Server>
  <Port>2143</Port>
  <User>admin</User>
  <Password>drum</Password>
</SessionManager>
```

These settings only work with active directory users. Calm application users (created in the Admin program) cannot join these groups.

14. If you're installing a model 4.5.2 or 5.0 Collections application, a certain standard original file name settings section needs to be present in your settings.xml. See the **Error! Reference source not found.** and **Error! Reference source not found.** paragraphs in chapter **Error! Reference source not found.** for the relevant sections.
15. Save the changes in the *settings.xml* file and close it.
16. In IIS, right-click your new application pool and choose *Recycle* in the pop-up menu.

Access rights

- **Adlib:** Adlib applications must be secured (typically on database and field level) using the Adlib access rights mechanism to restrict unauthorized users from access to certain or all data, because with a (session manager SQL) singular Axiell Collections installation any domain user can in principle log on and access the application if no Adlib access rights have been specified. So typically you would set applicable access right *Read*, *Write* or *Full* for specific authorized users or user groups, and assign *None* access right to \$REST per database to keep out all other users.

When you've configured your application for multi-tenancy, users can only log on to their designated application if they are a member of the proper Active Directory group, so typically you would populate these AD groups only with people that have at least read access to some or all of the databases, and further specify their access rights with the Adlib access rights mechanism.

Note that for Collections, the *Default access rights* for a *.pbk* cannot be set to *None*, since that would cause the application to be invisible for all Collections users. Also note that the *Exclude* and *Include* database *Authorisation types* currently (November 2019) have not been implemented, while *Specified rights* has been. So when preparing an Adlib application for Axiell Collections, you may want to check these settings in case a different access rights solution is required.

- **Calm:** User access is controlled using the Calm Admin program. Security must be enabled in the Admin program. If you are using Windows Active Directory logins to access Calm, any users that you would like to be able to access Collections must appear in or be added to the *User* tab of the *Security* dialogue box in the Admin program.

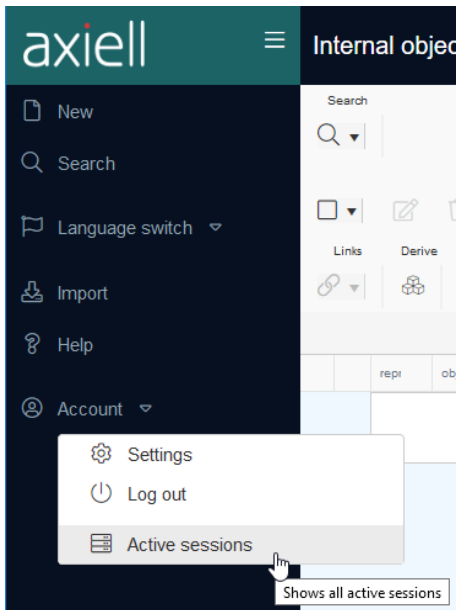
Login

The installation is now complete. You can start using Axiell Collections by entering the proper URL in your browser. (An Axiell Collections application installed locally on your PC can be accessed in your browser by entering `http://localhost/` in front of your web application name, for example: `http://localhost/collections`.) In most cases you'll subsequently have to log in using your own Windows login name and password.

(If there's no Active Directory on the server on which you've installed and access your (singular) Collections web application, you can log in with your Windows credentials, but you'll have to specify the name of the relevant server or pc in front of your user name, like: `my-machine\user`.)

If during login you get a server error stating that the system cannot contact a domain controller to service the authentication request, it means Axiell Collections has no access to the server which stores all network user names and passwords (aka the domain controller). So check whether the domain controller is accessible from the server on which Axiell Collections was installed.

Active Sessions

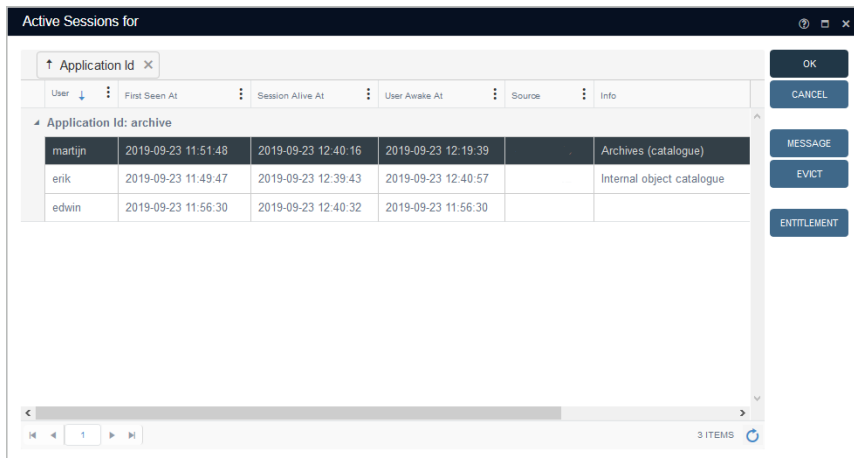


(Active Directory) users mapped in the `.pbk` to the `$ADMIN` role (no application authentication required) get an *Active sessions* option in

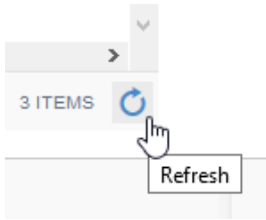
the *Account* menu, that other users won't see. It allows system administrators or other admins to monitor who's logged into Collections, send messages to those users, close their sessions via the *Evict* button and view some license details for Axiell Collections. In an enterprise environment where a single Collections installation runs multiple Adlib applications, this means that the admin can see all users logged into any of those Adlib applications.

The *Active Sessions* window shows 6 columns:

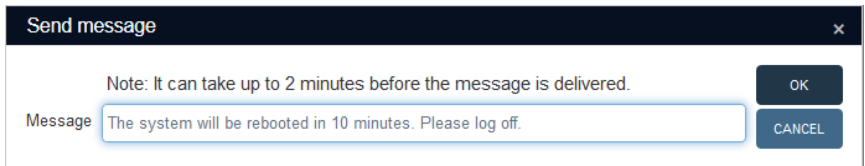
- *User name* – as registered in Active Directory;
- *First seen at* – date and time of login;
- *Session alive at* – the latest date and time the user session was still alive;
- *User awake at* – the latest date and time the user performed some action in Collections;
- *Source* – ip address of the relevant computer;
- *Info* – active data source, *Internal object catalogue* for example



The *Active Sessions* window is not refreshed automatically currently (23-09-2019). Click the *Refresh* icon in the bottom right corner of the grid view to refresh the data shown in the grid view.



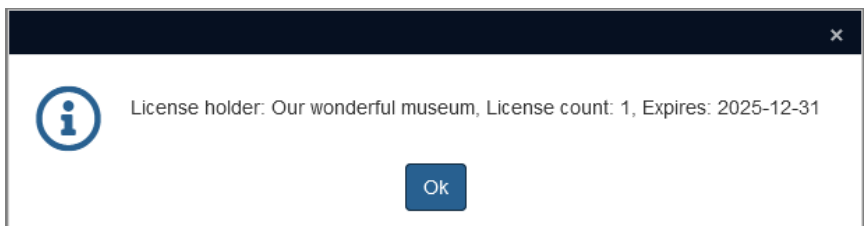
Select a single user and click the *Message* button if you'd like to notify that user of something. (You cannot select multiple users.)



Type your message in the *Send message* window and click *OK*. The relevant user will get to see a pop-up message in his or her Collections application, stating the message and the source (the domain name) of the message. The Windows task bar will also notify the user of the event in Collections. The user has to click *OK* in the message to acknowledge it.

Click *OK* or *Cancel* in the *Active Sessions* window to close it.

In the *Active sessions for <license holder>* window, click the *Entitlement* button to view some license information. The holder name, license count and expiry date will only be present if *settings.xml* contains this information.



In *settings.xml* you can optionally include license information per `<SessionManager>` node. If you don't, the license will be implicitly unlimited. If you do, note that inclusion is only sensible in hosted environments where the customer cannot edit this information in *settings.xml*. An example of these settings is the following:

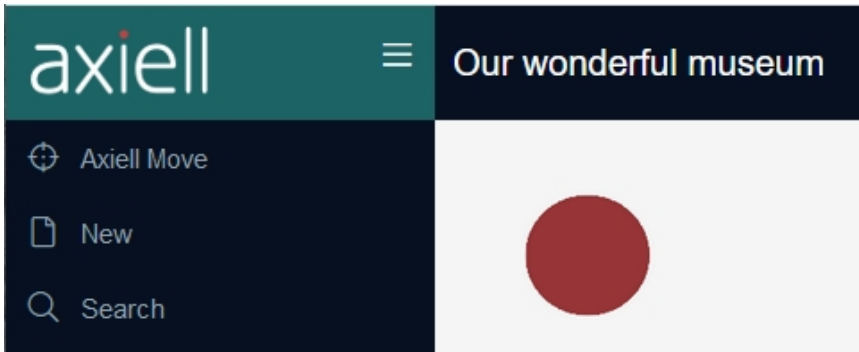
```
<LicenseInfo>
  <Holder>Our wonderful museum</Holder>
  <Count>1</Count>
```

```
<Expires>2025-12-31</Expires>
</LicenseInfo>
```

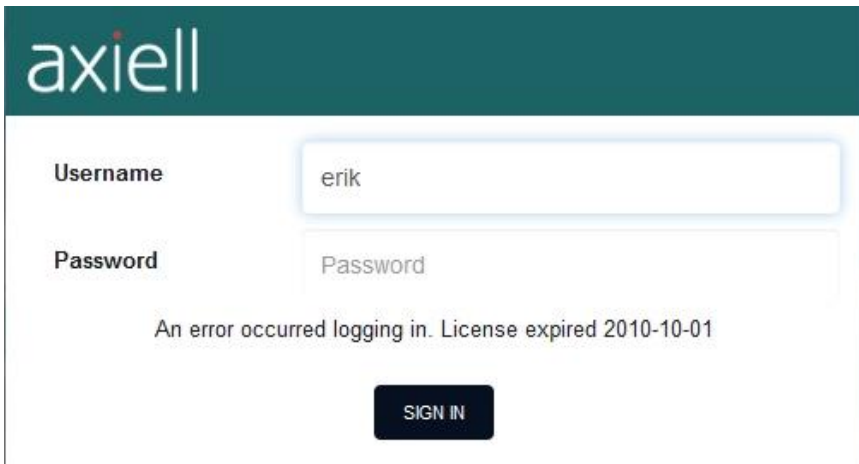
<Count> should contain the number of licenses.

In a multi-tenancy environment you could include license info in the *Multi* session manager to apply to all tenants and specify other license info per tenant if required, in which case the latter has precedence for a tenant. But you can also just specify license info per tenant.

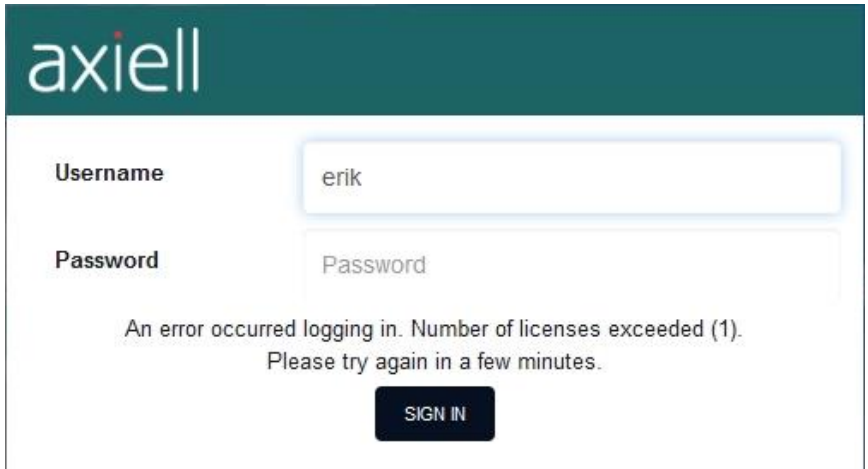
The holder name will appear in Collections directly after logging in but disappears after searching or creating records.



If the license has expired, users get a message similar to the following when trying to log in (after which login fails):



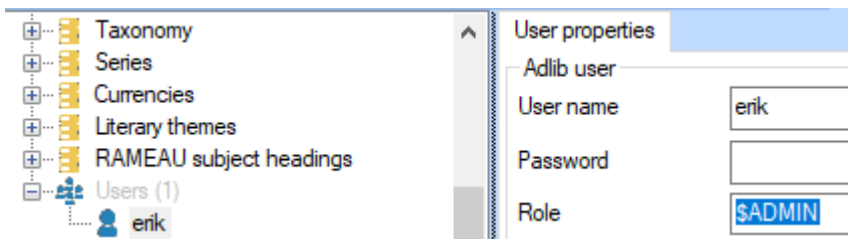
When a user tries to log in while already as many other users are logged in as there are licenses, then a message similar to the following appears (after which login fails):



Collections counts licenses in use per browser, so if a single user has a session open in browser x and one in browser y, it will count as two active licenses.

Also note that from 1.9.6, if a user does not properly close the session by logging out, but instead by just closing the browser or browser tab, it takes about five minutes before the license is cleared and another user is able to log in. In older versions, inactive licenses were not cleared automatically at all.

Designer setup: the application doesn't need application authentication, but users who should have access to the *Active Sessions* functionality, should be registered as a user with the *\$ADMIN* role in the application definition (*.pbk*). The user name must match their Active Directory user name (AD groups do not function).



Maximum file upload and download settings

From Collections 1.17, it is possible to upload and download files (images, documents, zip files etc.) larger than 2GB to and from image and application fields. This is part of the core software functionality and doesn't require any setup in Designer.

However, Collections applications are hosted with IIS and IIS has to allow certain maximum file sizes for upload and download too. This is handled by the `httpRuntime maxRequestLength` attribute and the `requestLimits maxAllowedContentLength` attribute in the `web.config` file relevant to the Collections application. A `web.config` file is a web application's settings file which is read and maintained by IIS (and indirectly the ASP.NET Core module) to configure that web application. In principle you can edit a `web.config` file manually in a text editor like Notepad++, but everything you need to do can be done from within the more user-friendly interface of IIS too (see further down this topic).

- The `maxRequestLength` property indicates the maximum allowed file upload size supported by ASP.NET and is specified in **kilobytes**. By not setting it too high on *Default* site level you can prevent any denial of service attacks from users posting many large files with the intention of overloading the server. Setting it as high as the maximum expected file size to be uploaded on Collections site level is required to be able to upload and download your very large files.
- The `maxAllowedContentLength` property specifies the maximum length of content in a single request to a web server and is specified in **bytes**. This value is limited by IIS and depends on the version of IIS. IIS 6 still had a limit of 4 MB while IIS 7 had a limit of 28.6 MB, but for IIS 10 it's around 4.3 GB already. However, the `maxAllowedContentLength` property is not relevant to your maximum file upload size: this is because Collections (from version 1.17) breaks up large files into chunks of about 1 MB only. The current property only really needs to be large enough to handle a single chunk and regular web application communication to the server. So the default limits set by IIS for this property can always be left as is.

Both settings apply to uploads as well as downloads.

This seems easy enough but there are a few considerations to take into account:

- Every site level in IIS has its own `web.config` file (in the root folder of the relevant site). So the *Default* site has a `web.config`, the

Collections site has its own config and any level(s) in between as well. However, as far as maximum file upload size is concerned you only need to care about the web.config on the Collections site level.

- With every update of Collections to a new version, a new, default web.config file will be installed in the Collections root folder. An obvious disadvantage of this is that any custom Collections-specific maximum values settings will be reset to their defaults and will have to be set again manually to the desired higher value.
- A last consideration is the performance toll that uploading and downloading large files has on Collections. While uploading is accompanied by a progress indicator in the field itself, showing the uploaded percentage of the file (making the wait acceptable), when saving the record however, you'll get no progress bar and for each GB you'll have to wait around a minute longer for the record to save. So a record with an uploaded file of 10 GB may take around 10 minutes to save and all you see is a wait symbol...

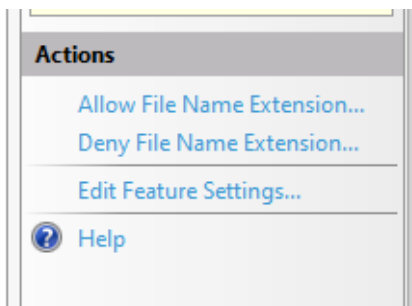
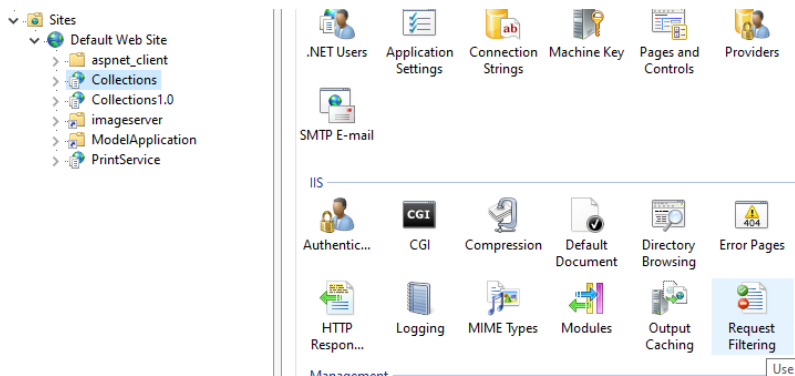
Note that during uploading of the file and saving of the record, the disk space of the server* on which Collections has been installed temporarily decreases with the size of the uploaded file, so if you're uploading a file of 10 GB, then at least 10 GB of free space must temporarily be available on the relevant disk, otherwise you'll get an error in Collections and the upload can't complete. The used disk space (of a successful upload at least) is freed up after the record has been saved. During upload and before saving of the record, the chunks into which the upload has been split up, are saved in the *App_Data\FileCache\chunks* subfolder of your Collections installation by default. If an upload was cancelled (because you closed the browser during an upload or when your disk was full before the upload could complete) then the cache might not have been cleared. You'll need to do that manually then, by deleting all subfolders from the *App_Data\FileCache\chunks* subfolder: you can do that in Windows Explorer. Make sure no-one else is working in Collections when you do that.

* The `<add key="GlobalStateFolder" value="" />` property is a Collections-specific setting added by a Collections installation to the web.config. For the `value` you could enter a path to an appropriate cache folder (other than the default local *App_Data\FileCache\chunks* subfolder of your Collections installation). By default, the disk is used where Collections is installed. If you use the `GlobalStateFolder` setting to point to a cache folder on another server, there will be a performance penalty of course.

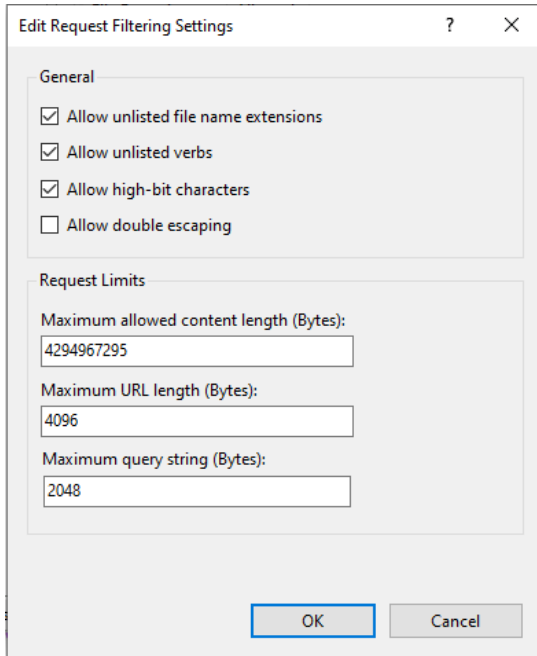
Further note that the `<add key="maxFileUploadSize" value="4294967295"/>` is an earlier Collections-specific setting (added by Collections to the web.config) with which a limit for file upload size could previously be set. By default it gets the size of the default IIS `maxAllowedContentLength`. It has no function any more.

Using IIS to make the desired settings

1. In IIS, double-click the *Request Filtering* icon for the Collections site and then click the *Edit feature settings...* option underneath *Actions* in the column on the right.

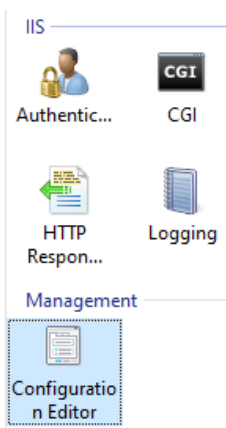


2. After an update of Collections or a new Collections installation, the *Maximum allowed content length* (in bytes), for the Collections site at least, shows the default setting for this version of IIS and cannot be set higher. If it had been set to a lower value than the maximum previously, you can still set it to the max again.

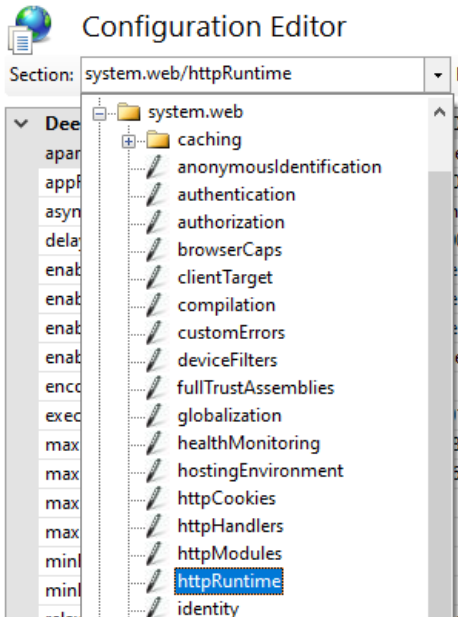


The `maxRequestLength` property can be set via IIS too:

1. Select the Collections site.
2. Double-click the *Configuration Editor* icon in the Management section.



3. In the *Section* drop-down list at the top of the *Configuration editor*, select *system.web/httpRuntime*.



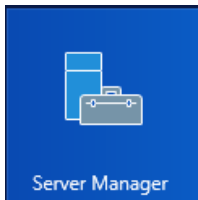
4. In the properties list beneath the drop-down, enter a new value behing *maxRequestLength*. Enter it in kilobytes, e.g. 6000000 kilobytes (representing 6 GB). It can be as high as you like (max 2 TB, but obviously not recommended).
5. Click the *Apply* option in the *Actions* column to save the changes.

Recycle the site's application pool if this hasn't happened automatically already.

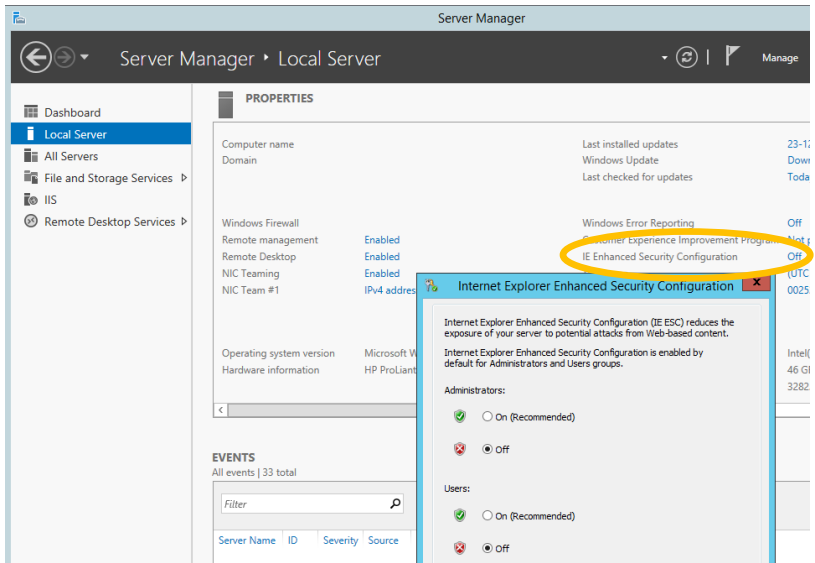
Appendix A: installing Web Deploy

On a host server, the *Web Deploy* tool must be present to be able to install Axiell Collections. If the tool is not present yet, you may install it as follows:

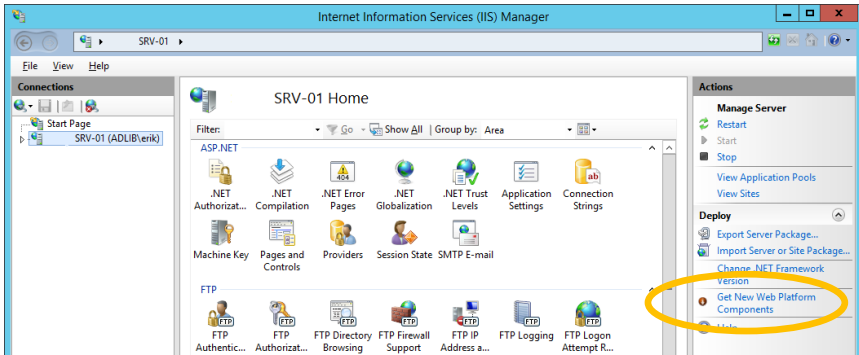
1. On the server where you'd like to install Axiell Collections, open the *Server Manager*.



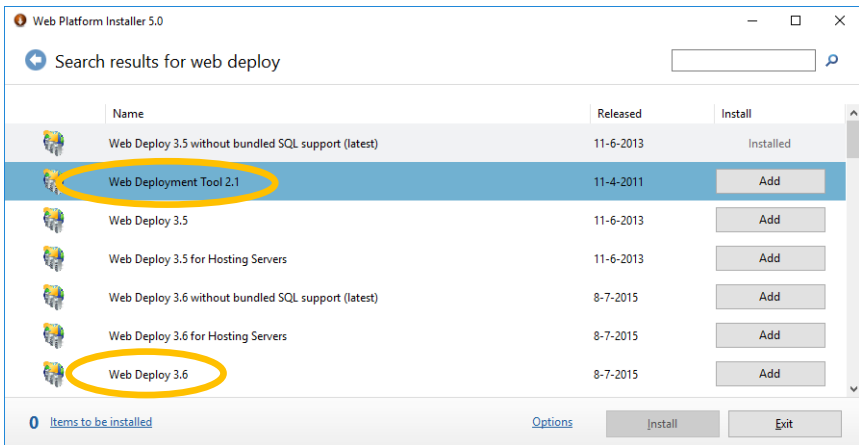
2. Click the *Local Server* option in the left column. In the *Properties* box, check the *IE Enhanced Security Configuration* setting. If it is set to *On*, click the underlined *On* to open the *Internet Explorer enhanced security configuration* and set either *Administrators* (if you're the administrator) or *Users* (if you are a different user) to *Off*. This will give you administrator installation rights on the server.



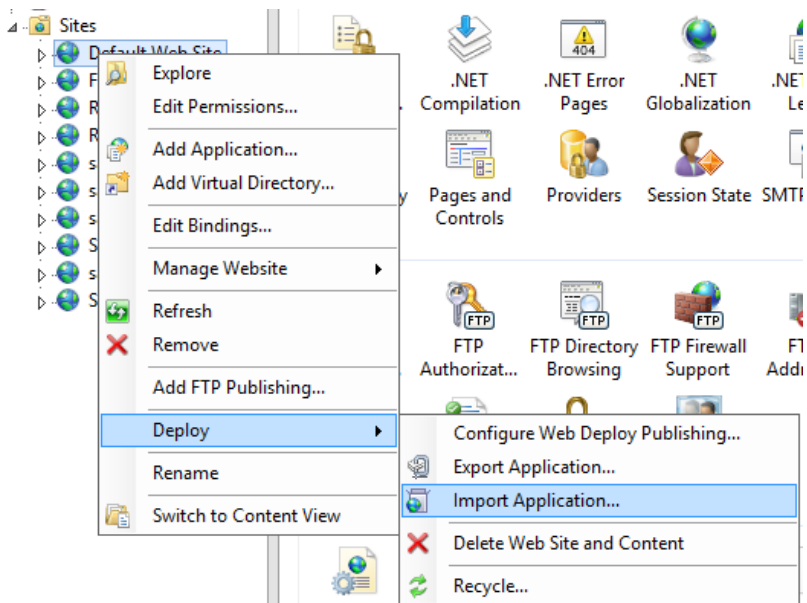
- Open IIS. In the tree view on the left, select the server level node. In the *Action* column on the right, click the *Get new web platform components* option.



- Type "Web deploy" in the search box in the right upper corner of the *Web platform installer* and press **Enter**.



- Add both the *Web deployment tool 2.1* and *Web deploy 3.6* and click *Install* to install them.
- Close and restart IIS and the *Deploy* option should now be available in the right-click pop-up menu for an application.



Appendix B: using SDI with smtp.office365.com

SDI (Selective Dissemination of Information) is an automated way to keep users or customers up-to-date with the latest acquisitions, or to send reminders, information about reservations or other current data, via e-mail or print. A (Windows task-scheduled) SDI server (Axiell *sdi.exe*) re-runs so-called [scheduled saved searches](#) (as created in Axiell Collections) at specified intervals and usually e-mails the resulting data nicely laid out by means of an XSLT stylesheet, to e-mail addresses listed in the relevant scheduled saved searches.

So in your Collections application a user creates a scheduled saved search, really just a saved search based with extra properties specific to SDI. An existing scheduled saved search can be edited via the same application.

An SDI server needs to be created first, based on *sdi.exe*. Whenever that program is being run it will execute the search statement stored in the scheduled saved search again if its frequency settings allow it to, then it will update the *Last run* field in the schedule and will print or e-mail the search result. An XSLT stylesheet (or *adapl*) set up in the scheduled saved search will determine which data ends up in the output and how it will look: Axiell Collections itself is not involved in the e-mailing or printing process.

Although you can run *sdi.exe* manually from the server via a batch file or (DOS) command line, it is much more convenient to use the Windows Task Scheduler on that server to regularly run the program via a batch file automatically so you won't have to bother anymore.

(*Sdi.exe* does not run continuously in the background: after it has processed the relevant scheduled saved searches, it closes automatically.)

Often, *sdi.exe* will be set up to use a local network SMTP server and by default it does not support TLS (Transport Layer Security) for encrypted e-mailing. However, when *sdi.exe* is combined with Axiell *ad-mailer.exe*, it is possible to use smtp.office365.com as the (TLS encrypted) SMTP server for SDI, although this requires a Microsoft office365 account: for smtp.office365.com mail boxes to be available to send e-mail through SMTP, the user (represented by a "from" e-mail address) sending the e-mails requires a subscription to either Microsoft Exchange Online Kiosk, Exchange Online (Plan 1) or Exchange Online (Plan 2) (which are not included in any Axiell product license). Sending e-mail through SMTP has the advantage that you can send multiple e-mails at once and specify any "from" e-mail address as long as you have the accompanying office365 account password. A disad-

vantage of SMTP is that an e-mail can only have one attachment: if an adapl tries to send an e-mail with multiple attachments anyway, an error 332 will be generated.

Installation

Apart from the Exchange Online subscription, you'll need some files to set up the SDI server. Most of these files are already present in the `\executables` subfolder of your Adlib software folder if you have Adlib for Windows as well as a local Axiell Collections installation, otherwise the Axiell Helpdesk can provide the missing files to you: you must create `sdi.bat`, `adlibweb.xml` and `adlibsmarthost.xml` yourself. Place all files together in a new subfolder of your Adlib software folder, `\SDI server` for example:

- `adlib.lic` – required license file
- `adlibimg.dll` – only required if images must be processed
- `adliblic.dll` – required for license file processing
- `adlibsmarthost.xml` – required settings file for office365 e-mailing
- `adlibu.dll` – required general function library
- `adlibweb.xml` – required settings file for `sdi.exe`
- `admailer.exe` – required executable for office365 e-mailing
- `evalsmtp.dll` – required SMTP function library
- `sdi.bat` – required to run `sdi.exe` via the Windows Task scheduler
- `sdi.exe` – required executable (7.6.19081.2 or higher) for general SDI functionality
- `<mySDIstylesheet>.xslt` – any stylesheet, required to lay out HTML e-mails

You may create the `adlibweb.xml` file in a text editor of choice, but the `adlibsmarthost.xml` file MUST be [created via Axiell Designer](#) because your office365 password needs to be encrypted. (Use Designer 7.6.19081.1 or higher if the password is longer than 16 characters: older versions only support shorter passwords.)

Example of `adlibweb.xml`:

```
<?xml version="1.0" encoding="UTF-8" ?>
<webConfiguration>
  <globalConfiguration>
    <databasepath>C:\Adlib\Model4.5.2\data</databasepath>
    <xmltype>grouped</xmltype>
    <sdifrom>myemail@ourmuseum.com</sdifrom>
```

```

    <dm>dm</dm>
    <di>di</di>
</globalConfiguration>

<databaseConfiguration database="collect">
    <database>collect<int>intern</database>
</databaseConfiguration>
</webConfiguration>

```

Setting	Meaning
databasepath	the path to the <i>\data</i> folder of your Axiell/Adlib software folder.
xmltype	must be grouped.
sdifrom	e-mail address from which the mailing must be sent.
dm	the Adlib tag or field in which the modification date of a record is saved. This has to be an indexed field. The dm-setting is optional and only relevant if the scheduled saved search <i>Pruning</i> option ever gets set to <i>DM/DI changed</i> .
di	the Adlib tag or field in which the entry date of a record is saved. This has to be an indexed field. The di-setting is optional and only relevant if the scheduled saved search <i>Pruning</i> option ever gets set to <i>DM/DI changed</i> .
database	a specified database alias to be able to process scheduled saved searches per database. The database name and optional dataset must be separated by <i>&gt;</i> ;

Example of the *adlibsmarthost.xml* file created via Designer:

```

<?xml version="1.0" encoding="utf-8"?>
<adlibSmartHost
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <server>smtp.office365.com</server>
    <userName>myemail@ourmuseum.com</userName>
    <password>7894j kfe523jklfds89dd89s0</password>
</adlibSmartHost>

```

Setting	Meaning
server	always smtp.office365.com in this case.
userName	e-mail address from which the mailing must be sent. This e-mail address must match the <sdifrom> e-mail address in <i>adlib-web.xml</i> , otherwise the smtp server will return an error. The office365 server does not allow you to impersonate someone else. The e-mail address doesn't need to be your own though: it can be a general address of your institution for example, as long as it has its own office365 account (with Exchange Online subscription).
password	the hashed password (hashed by Axiell Designer) for the office365 account identified by the e-mail address provided in <code>userName</code> .

On the SDI server, MSXML4 or higher also needs to have been installed. Version 4.0 SP2 of the MSXML parser (or a newer version) from Microsoft should be installed. You can check this in your registry (start `regedit` from the Windows search box). In it, search for the text `MSXML`. If the parser is not present, then you can download this software from the Microsoft website (<http://www.microsoft.com>); from their homepage, search for `MSXML4` or `MSXML 6.0`. Click a link similar to *Download Microsoft XML Core Services (MSXML) 4.0 Service Pack 1*. Then choose the *msxml.msi* file and store it on your hard disk. Now install the software by double-clicking the file.

Creating the output format

You must create an XSLT stylesheet or an `adapl` to specify which fields from the search result of an executed scheduled saved search should end up in the e-mail or printed output and to apply a nice layout to it.

For information about how to write an SDI layout `adapl` (if you don't want to use an XSLT stylesheet), please see chapter 2.2 in the [Reference guide SDI and e-mail from within Adlib](#) (the `adapl` part applies to SDI in combination with Axiell Collections too).

To create an XSLT stylesheet, you should know the following. Under the hood of Axiell Collections, all data is processed as XML. With an XSLT stylesheet that XML can be transformed to HTML, while HTML can be displayed in (SDI) e-mails.

So you have to create an XSLT stylesheet to transform XML data of records from a certain database into HTML. Assume this is grouped XML, although grouped XML generated by Axiell Collections may differ slightly from that generated by Adlib for Windows (so pre-existing stylesheets may need adjusting for use in Collections). You can use the sample stylesheet below for museum objects to start with. You are of course free to adjust this stylesheet to your requirements. Place the stylesheet in the earlier created SDI server folder and give it a sensible name.

As an illustration, the sample code also contains a template for the *reproduction.reference* field, which makes sure that a link to the images belonging to a record are included in the output, so that those images become visible in the printout or e-mails as well. This template assumes that the relevant field only contains a file name (without path) and that the images folder in this example can be found on `\\server1\Axiell\images\`. You should change this UNC path to the UNC path to the folder which actually contains your images. The images themselves are not included in the e-mail – an e-mail could easily become too large – only a link to it. This means that this template has limited applicability for e-mails with images, namely only for e-mails sent within your local network in which everyone has access to this UNC path. SDI e-mails opened on computers outside the network, would not show any images.

If you have an Axiell Web API image server, you could use the `wwwopac` calls (URLs) for retrieving images to include links to images in e-mails which should also work outside the local network. You can also leave out the processing of images entirely of course, by simply not including a template for it in the stylesheet.

Further note that if SDI doesn't send any e-mails while you did set it up like that, one of the possibilities is that something is wrong with your XSLT stylesheet. E-mails can only be sent when your stylesheet is 100% okay syntactically.

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <xsl:output method="html" encoding="UTF-8" indent="yes"/>

  <xsl:template match="/adlibXML">
    <html>
    <head>
      <STYLE type="text/css">
        p {font-family:"verdana"; font-size: 10pt}
      </STYLE>
    </head>
    <body>
```

```

        <xsl:apply-templates select="recordList"/>
    </body>
</html>
</xsl:template>

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

<xsl:template match="record">
    <p>
        <xsl:apply-templates select="object_number"/>
    </p>
    <xsl:if test="count(Object_name) != 0">
    <p>
        <table border="1" cellpadding="10"
            style="border-collapse: collapse" width="100%">
            <tr>
                <td width="100%" bgcolor="#EAEAFF">
                    <p>
                        <xsl:apply-templates select="Object_name"/>
                        <xsl:apply-templates select="other_name"/>
                    </p>
                    <p>
                        <xsl:apply-templates select="Description"/>
                    </p>
                    <xsl:if test="Production/creator != ''">
                    <p>
                        <xsl:apply-templates select="Production"/>
                    </p>
                    </xsl:if>
                </td>
            </tr>
        </table>
    </p>
    </xsl:if>

    <p>
        <table border="1" cellpadding="10"
            style="border-collapse: collapse" width="100%">
            <tr>
                <td width="100%" bgcolor="#F9F9F9">
                    <p>
                        <xsl:if test="physical_description != '' or
                            Dimension/dimension.value != ''">
                            <xsl:text>Physical description:</xsl:text>
                            <br/><br/>
                        </xsl:if>
                        <xsl:apply-templates select="physical_description"/>
                        <xsl:if test="Material/material != '' or
                            Dimension/dimension.value != ''">
                            <p>
                                <xsl:if test="Material/material != ''">
                                    <xsl:apply-templates select="Material"/>
                                </xsl:if>

```

```

        <xsl:if test="Dimension/dimension.value != ''">
            <xsl:if test="Material/material != ''">
                <xsl:text>, </xsl:text>
            </xsl:if>
            <xsl:apply-templates select="Dimension"/>
        </xsl:if>
        <xsl:text>.</xsl:text>
    </p>
</xsl:if>
<xsl:if test="Reproduction/reproduction.reference != ''">
    <p>
        <xsl:apply-templates
            select="Reproduction/reproduction.reference"/>
    </p>
</xsl:if>
</td>
</tr>
</table>
</p>
</xsl:template>

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

<xsl:template match="object_number">
    <xsl:text>Object: </xsl:text>
    <xsl:value-of select="."/>
    <xsl:apply-templates select="../object_category"/>
</xsl:template>

<xsl:template match="object_category">
    <xsl:text> (</xsl:text>
    <xsl:value-of select="normalize-space(.)"/>
    <xsl:text>)</xsl:text>
</xsl:template>

<xsl:template match="Object_name">
    <strong>
        <xsl:if test="position() > 1">
            <xsl:text> &amp; </xsl:text>
        </xsl:if>
        <xsl:value-of select="object_name"/>
    </strong>
</xsl:template>

<xsl:template match="other_name">
    <xsl:text>, other name: </xsl:text>
    <xsl:value-of select="."/>
</xsl:template>

<xsl:template match="physical_description">

```



```

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

<xsl:template match="Description">
    <xsl:if test="description != ''">
        <xsl:value-of select="description"/>
    </xsl:if>
</xsl:template>

<xsl:template match="Material">
    <xsl:choose>
        <xsl:when test="position() = 1">
            <xsl:text>Made of </xsl:text>
        </xsl:when>
        <xsl:otherwise>
            <xsl:text> &amp; </xsl:text>
        </xsl:otherwise>
    </xsl:choose>
    <xsl:value-of select="material"/>
</xsl:template>

<xsl:template match="Dimension">
    <xsl:if test="position() &gt; 1">
        <xsl:text>, </xsl:text>
    </xsl:if>
    <xsl:value-of select="dimension.type"/>
    <xsl:if test="dimension.part != ''">
        <xsl:text> (</xsl:text>
        <xsl:value-of select="dimension.part"/>
        <xsl:text>)</xsl:text>
    </xsl:if>
    <xsl:text> </xsl:text>
    <xsl:value-of select="dimension.value"/>
    <xsl:text> </xsl:text>
    <xsl:value-of select="dimension.unit"/>
</xsl:template>

<xsl:template match="Production">
    <xsl:variable name="pos" select="position()"/>
    <xsl:choose>
        <xsl:when test="$pos = 1">
            <xsl:text>Created by </xsl:text>
        </xsl:when>
        <xsl:otherwise>
            <xsl:text> &amp; </xsl:text>
        </xsl:otherwise>
    </xsl:choose>
    <xsl:apply-templates select="creator"/>
    <xsl:if test="position() = last()">
        <xsl:apply-templates select="../Production_date[1]"/>
    </xsl:if>
</xsl:template>

<xsl:template match="Production_date">
    <xsl:text> </xsl:text>

```

```

<xsl:choose>
  <xsl:when test="(production.date.start !=
    production.date.end) and (production.date.end != '')">
    <xsl:text>between </xsl:text>
    <xsl:value-of select="production.date.start"/>
    <xsl:text> and </xsl:text>
    <xsl:value-of select="production.date.end"/>
    <xsl:text>.</xsl:text>
  </xsl:when>
  <xsl:otherwise>
    <xsl:text> in </xsl:text>
    <xsl:value-of select="production.date.start"/>
    <xsl:text>.</xsl:text>
  </xsl:otherwise>
</xsl:choose>
</xsl:template>

<xsl:template match="creator">
  <xsl:choose>
    <xsl:when test="contains(., ',')">
      <xsl:value-of select="substring-after(. , ',')"/>
      <xsl:text> </xsl:text>
      <xsl:value-of select="substring-before(. , ',')"/>
    </xsl:when>
    <xsl:otherwise>
      <xsl:value-of select="."/>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
</xsl:stylesheet>

```

Scheduling SDI in the Windows Task Scheduler

You may create a batch file to run *sdi.exe* with parameters. Simply create a text file with the *.bat* extension, name it *sdi.bat* for instance, place it next to *sdi.exe* and put a command line like the following in there:

```
sdi "SDICMD=run" > sdilog.xml
```

This will execute all applicable SDI profiles after which an XML log file (here *sdilog.xml*) will be created in the same folder, containing the result of the action and any errors.

In the Windows Task Scheduler you must make some settings if you'd like your *sdi.bat* batch file running *sdi.exe* to be executed automatically on a regular* basis. Of course it depends on the version of Windows in exactly what way you can make these settings. See Microsoft's documentation for more information about that.

On Windows Server 2012 R2 for example, this can be done as follows:

1. Search for 'schedule tasks' to open the *Task Scheduler*.
2. Click the *Create basic task* option in the right window pane.
3. A wizard will guide you through the settings for a regular execution of your batch file.

To test your SDI server before you have the Task Scheduler execute the SDI profiles on a regular basis, you can double-click the saved batch file in Windows Explorer to execute it this once.

* A note about the frequency: in your Axiell Collections application you set the frequency with which the scheduled saved search ideally should be executed. Remember however, that the frequency can be limited by the Windows Task Scheduler settings for the SDI program on the server. If the system administrator has set the program to run only once a day, users will not be able to receive a new SDI update more often than that.

Testing and possible errors

Possible errors (as logged in *sdilog.xml*) causing e-mailing to fail, include:

- ```
<pointerfile stat="332" info="Failed to send email with admailer.exe, the process returned error code 332, with output: Admailer 1.0.0, TLS capable email client for use with Adlib for Windows. Use option -h for command line help. Sending email... Failed to send message: dial tcp 43.104.151.31:587: connectex: No connection could be made because the target machine actively refused it. " location="pf000009.ptr">
```

**A *No connection* error like this could indicate that your local network does not accept SMTP traffic from the ip address from which e-mails are attempted to be sent. Your system administrator should be able to help.**

- ```
<pointerfile stat="332" info="Failed to send email with admailer.exe, the process returned error code 332, with output: Admailer 1.0.0, TLS capable email client for use with Adlib for Windows. Use option -h for command line help. Sending email... Failed to send message: 535 5.7.3 Authentication unsuccessful [CB89R05fd2348.eurprd06.prod.outlook.com]
```

```
" location="pf000009.ptr">
```

An *Authentication unsuccessful* error like this is usually caused by an incorrect user name/e-mail address and/or password. You need an e-mail address and accompanying, hashed password (encrypted by Axiell Designer) for a valid office365 login.

- `<pointerfile stat="333" info="applying XSLT SDIrecentAcquisitions.xslt gives errors: ', = ', check XSLT code please" location="pf000009.ptr">`

An error like this indicates a programming error in the XSLT stylesheet. No e-mails will have been sent.

To test the SDI server separately from the SDI client-side, you could use a URL or a batch file on the server containing a command line to execute *sdi.exe* with certain parameters. Please see chapter 2.4 in the [Reference guide SDI and e-mail from within Adlib](#) for more information about that.

Admailer.exe itself can also be used for testing the connection to *smtp.office365.com* and to see if an e-mail can be sent at all, without using any of the settings in the XML configuration files. You can start it from the (DOS) command-line with the `-h` parameter to get a list of all possible parameters.

For example, to check if a connection can be made at all:

```
admailer -server smtp.office365.com -check
```

or to actually send an e-mail with just a subject, in which case you must use the real, unhashed office365 password(!) because we are circumventing *sdi.exe* now:

```
admailer -server smtp.office365.com -from erik@ourmuseum.com -to john@ourmuseum.com -title testing -username erik@ourmuseum.com -password myoffice365pw*0
```