

A large, light gray decorative graphic consisting of two overlapping circles connected by a narrow, curved bridge, resembling a stylized '3' or a continuous loop. It is positioned in the background, partially behind the text.

# **Axiell Collections 3.0**

## On-premise installation and configuration

## Axiell ALM Netherlands BV

---

Copyright © 2025 Axiell ALM Netherlands BV® All rights reserved.

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 provide services, supply software, or as a definitive product description.

---

# CONTENTS

<b>1 Preface</b>	<b>1</b>
<b>2 Requirements</b>	<b>2</b>
<b>3 First-time environment setup</b>	<b>7</b>
3.1 Application folder structure and access	7
3.2 MS SQL Server	8
<i>Initializing the new SQL Server databases</i>	9
<b>4 Installing Axiell Collections</b>	<b>11</b>
<b>5 Configuring Collections</b>	<b>16</b>
<i>Setting the base URL</i>	16
<i>Making sure that the Help is accessible</i>	16
<i>Security (CSP)</i>	17
<i>The Collections Scheduler</i>	17
<i>Using a secure connection</i>	17
5.1 Using session managers	18
<i>Using a single (non-tenanted) Collections application</i>	18
<i>Using a multi-tenanted Collections application</i>	19
5.2 SessionManager settings	22
<i>LicenseInfo</i>	22
<i>Setting Key=Path</i>	22
<i>Setting Key=ImageCacheFolder</i>	23
<i>Setting Key=Preload</i>	24
<i>Setting Key=Domain...</i>	24
<i>Setting Key=Languages</i>	25
<i>Setting Key=AlwaysPromptBeforeSave/</i>	
<i>EnableResultSetEdit</i>	26
<i>Setting Key=UseOriginalMediaFileName</i>	27
<i>Setting Key=DashboardVisible</i>	28
<i>Setting Key=ConnectionString</i>	28
<i>Setting Key=DefaultExpand</i>	28
<i>Setting Key=Theme</i>	28
<i>Setting Key=CacheScripts</i>	29
<i>DesignatedJobs</i>	29
5.3 Setting up direct printing	30
5.4 AlmSettings (for GIS and image server)	33
<i>ImageServerConfiguration</i>	33
<i>GIS</i>	34
5.5 Default Record details view screens filter	37
5.6 Logging	37
5.7 Theme Name=dark	37
5.8 SSO, OpenId and SAML	37
5.9 Configuring the path to uploaded templates	37
5.10 Maximum file upload and download settings	38
<i>Considerations</i>	39

<i>Using IIS to make the desired settings</i>	40
5.11 Access rights	43
5.12 Editing settings json files (optional)	43
<i>Setting up Collections logging</i>	45
<i>Multi-factor authentication setup</i>	47
<i>Assorted extra configuration in settings.json</i>	49
<b>6 Collections login</b>	<b>51</b>
6.1 Application selection	54
6.2 Active Sessions	55
<b>Appendix A: preparing the Standard model application</b>	<b>59</b>
Populate two tables with default terminology	59
Relevant settings in Settings.xml	60
Setting the Application Id	60
Uploaded templates setup	61
Default application permission groups	61
<b>Appendix B: updating to Collections 3.0</b>	<b>63</b>
<b>Appendix C: possible IIS/Collections software issues</b>	<b>66</b>

# 1 Preface

This manual is mainly\* intended for application managers about to perform a first-time installation of an Axiell Collections system consisting of the Collections core software version 3.0 or up and a Collections model application (possibly customized) which contains the application's configuration files for e.g. data sources, screens and fields. The current model application is called the Standard Model application and has no version number, but the previous version was version 5.2. The Collections core software needs a Collections application to run.

The core software gets regular new versions to fix any bugs and introduce new functionality, while your model application is more static and meant for the long run. Core software updates are easy and quickly to install and do usually not require any changes to your application.

\* If you are just looking for some concise information about updating your existing Collections system from a core software version older than 3.0 to a 3.0 or higher version, please see Appendix B. You may then skip the rest of this manual.

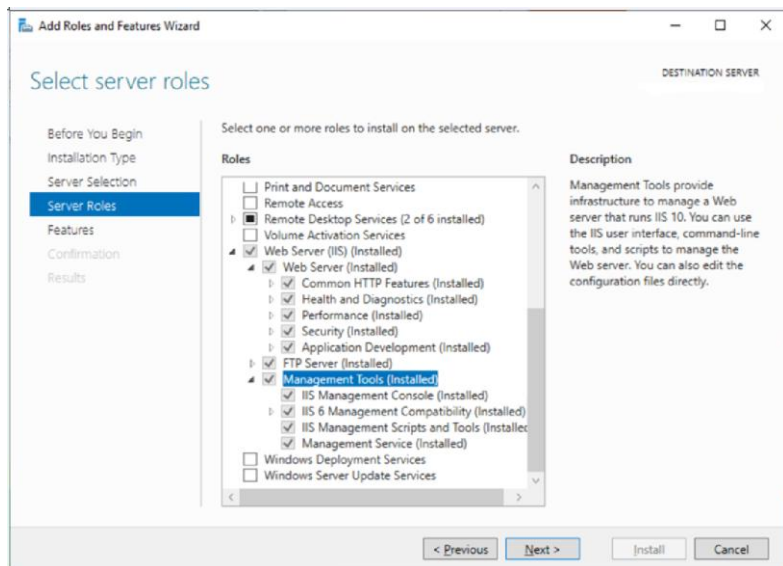
## 2 Requirements

See also the [Axiell Collections Technical Specifications and Architecture](#).

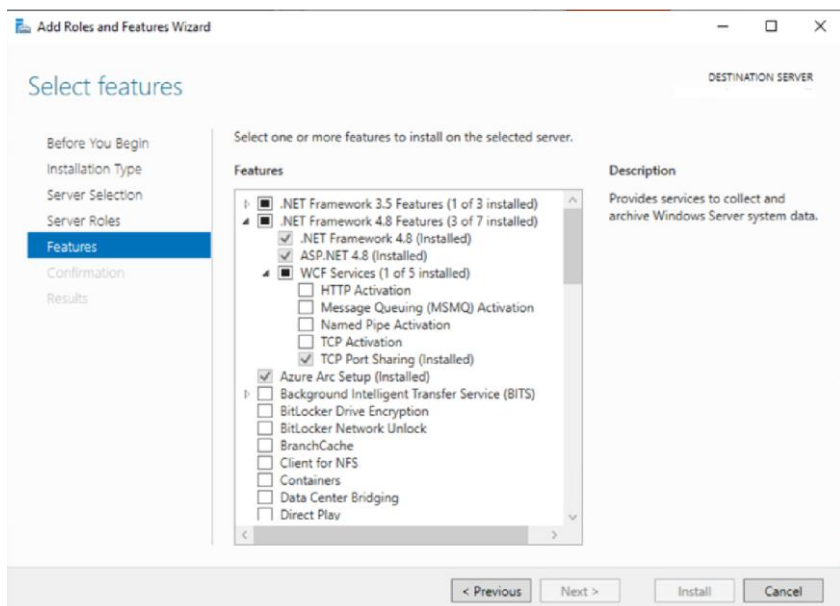
- Minimally a **Windows Server (2019 or later)** or Windows version which is still supported by Microsoft too, within an Active Directory domain\*, with at least 300MB free disk space.  
For Windows 2022 Server only, make sure that the latest Microsoft Visual C++ Redistributable Version is present on your server. If not present yet, 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](https://aka.ms/vs/17/release/vc_redist.x64.exe) for X64) and install it.
- An **SSL certificate** acquired from your local certification organization.
- HTTP server software must be installed on the server on which the web application will be placed, such as **IIS 10 or later**. For the required Windows versions, these services are probably already available but might still have to be enabled.  
When you need to **setup IIS features on a empty Windows server** you can use the following Powershell command in an elevated (run as administrator) Powershell session (which will install all IIS features, except the Deploy feature):

```
Get-WindowsFeature -Name Web-* | Install-WindowsFeature
```

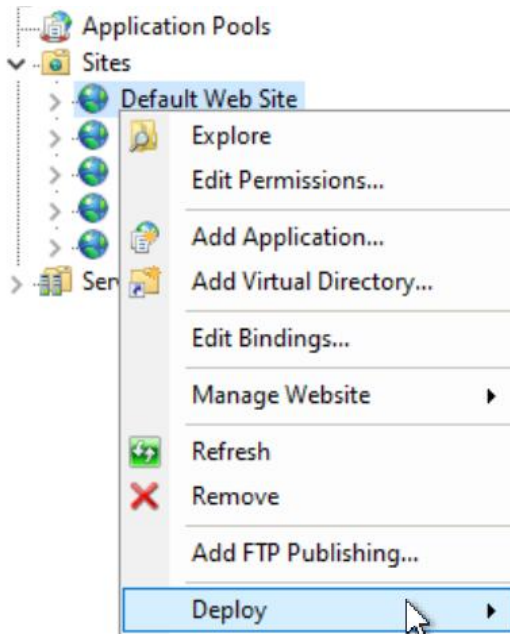
**To manually check the server settings**, open the *Server Manager* and on the *Dashboard* click the **Add roles and features** option. Then under *Server Roles*, look up the relevant *Web Server > Management tools* options and mark them if necessary.



And under *Features* for the .NET Framework 4.8 also mark the *HTTP Activation* checkbox, although that's not really required for Collections 3.0:



- The **IIS Web Deploy** extension should have been installed on the IIS Server to allow installation of Axiell Collections. 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://learn.microsoft.com/en-us/iis/install/installing-publishing-technologies/installing-and-configuring-web-deploy-on-iis-80-or-later> or a similar page. 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.)



- **Your licensed Axiell software**, consisting of at least the Axiell Collections core software and an Axiell Collections model (possibly customized) application (which contains the application's configuration files and media sub folder(s) and doesn't need to be located on the same server where the Collections core software will be deployed, per se). A 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. Each working environment you require, must have its own copy of the model application. Typical paths for the model application environments are `\Application\Primary\` or `\Application\Production\` and `\Application\Secondary` or `\Application\Testing\`.

Axiell Designer is required to change some settings in the application's configuration files (like to which SQL data source each application environment links), but may not be included in your licensed Axiell software for further use.

If you are just updating an existing Collections system with the Collections 3.0 core software, you already have a model application in place and you usually don't need to change its configuration.

- **Axiell Collections is browser-based.** The latest version of either Chrome, Firefox, Safari or Edge is recommended.
- **MS SQL server 2019 or later.** Important if, alongside Collections 3.0, you are also installing a new model application 5.1 or up: check that Full Text Search is/has been installed/enabled on your SQL Server instance, because in a default installation that is not the case. For a new SQL Server instance, mark the *Full-Text and Semantic Extractions for Search* checkbox in the feature selection window, while if SQL Server is already installed, you can re-run the installer, choose *Add features to an existing instance of SQL Server*, and then select the *Full-Text and Semantic Extractions for Search* checkbox. Please see the SQL Server documentation or e.g. [here](#) for complete information about how to check and/or install this option. Axiell Collections will not function properly without this option.

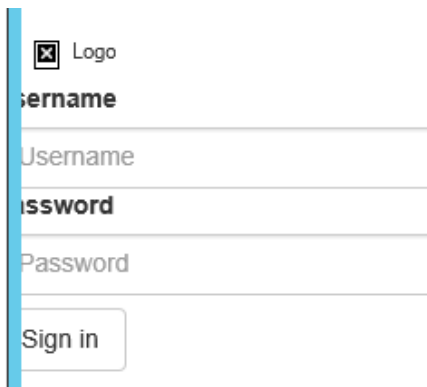
Typically, you, as a customer, are responsible for configuring and maintaining SQL Server backups for on-premises installations. If agreed otherwise, the Axiell Statement of Work for the project should explicitly state so.

**SQL Server Management Studio** is required to configure the SQL database. It is typically bundled with the installation of SQL Server, but if not it can be obtained [here](#). If you start with migrated data and a new model application, a SQL Server database backup file, may be provided by Axiell to initialize and restore your database for use with Axiell Collections: the Axiell Statement of Work should provide more details about that.

If you are just updating an existing Collections system with the Collections 3.0 core software, you already have a model application and correctly configured SQL database in place and you don't need to change its configuration.

- **.NET:** the Microsoft [.NET Core 8.0 Hosting bundle](#) (consisting of the ASP.NET Core Runtime for IIS support and the .NET Runtime to run Collections as a console app with its own built-in web server) is required for Collections 3.0 and needs to 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 might be missing from the installation. So on a new server, always install IIS before you install the .NET Framework. (If the .NET Framework still has to be installed, then please take into account that the web server might need rebooting after this installation. Also, if you do install IIS, it is a good idea to reboot after that installation and before the .Net Framework installation.) On IIS, ASP.NET must operate in integrated mode (which is the default configuration). The application pool which we will create for the Axiell Collections IIS application, must run in this mode.

\* Although not recommended, Axiell Collections, the Axiell model 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 model 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).



The screenshot shows a login interface. At the top left, there is a small square icon with an 'x' inside, followed by the text 'Logo'. Below this, the word 'Username' is partially visible, followed by a text input field. Underneath the first field, the word 'Password' is partially visible, followed by another text input field. At the bottom left, there is a button labeled 'Sign in'.

## 3 First-time environment setup

---

### 3.1 Application folder structure and access

The Collections core software and a Collections model application together are required to run Collections in a browser. For a first-time installation of both the core software and a model application, consider the following:

- Amongst other files, a **Collections model application folder** contains the model application configuration files, like e.g. *.inf* files in the *\data* sub folder defining your database tables structures, a *.pbk* to define your application structure and *.fmt* files to define screens, and media sub folders to contain linked images and documents. This folder and its contents must be accessible to the IIS application pool(s) that are running Axiell Collections and any Collections WebAPI(s). Please ensure that the application pool identities have *Read/Execute* access to the application folder and *Modify* permission to the media folder within it.

Out of the box, the current Standard Model application requires some extra preparation and configuration for it to get ready for use. See appendix A for more information.

- You can deploy the **Collections core software** into any folder on the IIS server but consider the preferred folder structures outlined below. If your server has access to multiple drives, we recommend that you leave the C: drive for Windows software and utilize an alternative drive for managing the Collections software installations. Please see the Axiell Collections installation section for more details.

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 **preferred main folder structure and naming** for the deployment of the Axiell core software is, per working environment something like:

```
<drive:>\>Axiell\Software\Web\Collections\  
<drive:>\>Axiell\Software\Web\Collections\API\  
<drive:>\>Axiell\Software\Web\Collections\AIS\
```

The preferred main folder structure and naming of the location of

(copies of) the model application is something like:

<drive:\>Axiell\Application\

Different working environments (like a production environment and optional testing and/or training environments, depending on the requirements and the agreed upon Statement of Work), should be further separated using sub folder names like *Primary* and *Secondary*, or *Production*, *Training* and *Testing* or *UAT* (User Acceptance Testing), as long as in turn their sub folder structures are copies of each other. So for two environments of model applications (each consisting of sub folders like \adapls, \screens, \data, \xplus, \main etc.) you could have main paths like e.g.:

G:\Axiell\Application\Primary

G:\Axiell\Applcation\Secondary

Variations are possible of course, but the **paths to the core software are not allowed to have spaces or dots** in them, otherwise you'll get an *HTTP error 405.0 Method not allowed* after logging in!).

Axiell desktop software like Designer or other tools (if included in the license) can be placed in folders like  
<drive:\>Axiell\Software\Desktop\, e.g.:

G:\Axiell\Software\Desktop\AxiellDesigner\Designer\_v7.17.

---

## 3.2 MS SQL Server

1. Using SQL Server Management Studio, create a database for each desired Collections working environment, so that test, training and production data is completely separate. Be clear in the naming of these databases, like *CollectionsProduction*, *CollectionsTesting* and *CollectionsTraining* or *CollectionsPrimary* and *CollectionsSecondary* for example.
2. Choose the same, appropriate *Collation* for the databases. The collation chosen can affect the sorting used during Full Text indexing/searching. Typically, we deploy using:  
SQL\_Latin1\_General\_CP1\_CI\_AS (accent-sensitive) or  
SQL\_Latin1\_General\_CP1\_CI\_AI (accent-insensitive) collations which work well with North American and Western European data.
3. If your server has multiple disks, you may want to consider designating a separate drive for the database transaction log files.

Please see the SQL Server manual for discussions relevant to the version of SQL Server you are installing.

If permitted, enable mixed-mode authentication for your databases to give you flexibility to use SQL Server-based user authentication as well as Windows authentication for accessing the databases.

If Active Directory authentication is 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 model application's internal access rights mechanism (as can be set up through Axiell 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).

Tip: to conserve disk space, set the *Recovery model* for the testing database to *Simple*. The *Recovery model* should be set to *Full* for the production database to reduce data loss should the database need to be recovered to a point in time.

## Initializing the new SQL Server databases

1. If available, restore the .bak file supplied by Axiell in each of the new databases to create the initial data structure needed by Axiell Collections. If a data migration was done, the .bak will also contain the migrated data itself.  
If you are installing the current Standard model application as well (instead of model 5.2 or older), two new database tables need to be populated with default values too. These might already be included the .bak file (in which case no further action is required on this point) or have to be imported separately after finishing the complete Collections installation. See appendix A for more information about this.
2. Create logins with the appropriate permissions in each database. If you utilize Windows security for your IIS application pools, then it is the application pool identity that needs to be added to *Security/Logins*. This user should minimally be granted *db\_ddladmin*, *db\_datawriter*, and *db\_datareader* access to each database. If permitted, *db\_owner* access can be granted instead, for ease of access.

3. A login should also be created for each application administrator. The Axiell Collections configuration tool (Axiell Designer) requires access to the SQL Server to perform certain actions (e.g. table/index creation). These users should be granted *db\_owner* membership for each database to allow them to create all of the necessary objects and to run ad hoc backups and restores and perform other actions like gathering database statistics.

Tip: if you restore a database from another environment, you will need to reconnect the logins to the restored database. You can do this via the *User Mapping* tab in the *Login Properties* window for each login.

## 4 Installing Axiell Collections

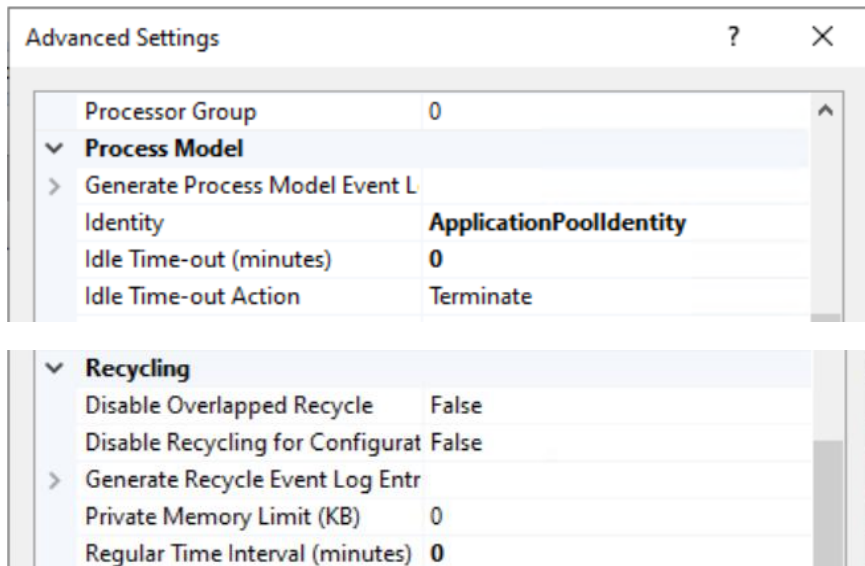
1. You have received the Axiell Collections core software deployment package for a first time installation or an update of Collections (**AxiellCollectionsCoreWebDeploy.zip** for .NET Core 8.0) from the Axiell ALM support desk. Place this file in e.g. the ..\Axiell\Software\Web\Collections folder you created earlier, which is a good place to store maybe multiple versions of the zip files.  
If you're only looking to update your existing Collections system with the new 3.# core software version, please see Appendix B for concise deployment instructions and skip the rest of this manual.
2. In Windows Explorer, also create new folders on the server where you'd like to install the Collections physical core software files for your different working environments later on in this procedure. If for example, you have an H: drive and you store your Collections .zip deployment files in H:\Software\Web\Collections you could create sub folders like:  
H:\Axiell\Software\Web\Collections\CollectionsProduction,  
H:\Axiell\Software\Web\Collections\CollectionsTesting and/or  
H:\Axiell\Software\Web\Collections\CollectionsTraining (but other names are fine too) depending on which environments you need.
3. For a first time installation only, open IIS on the server and create a new **.Net v4.0 application pool**, CollectionsPROD or CollectionsProduction for your production environment for example, and leave the default basic settings (*Integrated* mode) as they are.



Create separate application pools for any test and training environments, like CollectionsUAT or CollectionsTesting and CollectionsTRAIN or CollectionsTraining.

In the *Advanced settings* of each 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.

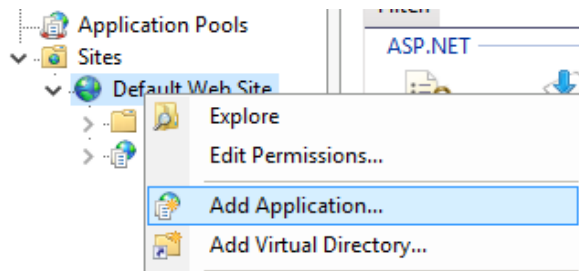


Also for performance improvement on startup, set the *Start Mode* to **AlwaysRunning**: this launches the worker process for the application pool as soon as IIS is started instead of launching it only when the first request for the web application is received.

4. Axiell recommends using **HTTPS** as the data transfer protocol. If SSL is not already configured, install the SSL certificate from your local certification organization 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 *Server 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 `<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.
5. For a first time installation only, right-click the *Default Web Site* in IIS and choose **Add application** in the pop-up menu in IIS and in




the dialog which opens, name the new application `CollectionsProduction` 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!), assign it your new matching application pool and mark the *Enable preload* checkbox to speed up performance. The name you provide here will be the last part of the URL to this application, so if your base URL is `https://mymuseum.com/` and your application name is *CollectionsProduction*, then the URL will become `https://mymuseum.com/CollectionsProduction`. In *Physical path*, browse to the appropriate folder created in step 2 where you'd like to install the physical files of the Collections core software for the current working environment.



In the *Add application* window, use the *Test Settings* button to check whether a connection can actually be made. Repeat this step for your other working environments and name the IIS applications appropriately.

6. 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.) In either case, right-click one of your new IIS applications underneath the *Default Web Site* and select ***Deploy > Import application*** from the pop-up menu.
7. For both a first time installation and an update: on the ***Select the package*** page in the *Import Application Package* wizard that has opened, *Browse* to your deployment package (*AxiellCollectionsCoreWebDeploy.zip*) and open it. Click *Next*.
8. In the following step, leave the selected *Contents of the Package* as is and click *Next*.
9. The ***Application Path*** entry field defaults to *Collections*. Clear this value because you don't want to create a new sub folder which would also become part of the URL. (The physical files will be installed underneath the *Physical path* for the IIS application that you set in step 5. Click *Next*.

Import Application Package

 **Enter Application Package Information**


Enter information that is required to install this package:

**Application Path**  
Full site path where you want to install your application (for example, Default Web Site/AxiellCollections/)

Default Web Site/AxiellCollections/

10. In the **Overwrite Existing Files** step, choose the first option if you already have a Axiell Collections 3.0 or higher installation in place on that location, which you'd only like to update with a new version of the core software: any custom settings in `\Configuration\settings.xml` remain as they were.

Import Application Package ? ;

 **Overwrite Existing Files**

You have chosen to install this application package to an existing application. Would you like to delete all the files on the destination?

☒ No, just append the files in the application package to the destination.

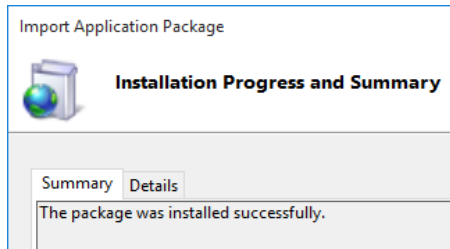
☐ Yes, delete all extra files and folders on the destination that are not in the application package.

Conversely, select the second option if you're updating a Collections pre-3.0 version (for .NET 4.8) to a 3.0 or later version (for .NET Core 8.0), but then do first **make a copy of your `settings.xml` file** from the `\App_Data` folder and paste it somewhere in a backup folder because the file will be deleted from the target folder while you need it back in the new `\Configuration` folder after completing this deploy wizard! (If you are using custom background images for the Collections login screen or a custom CSS, you should back up those as well from the `Content\Backgrounds\Custom` and `Content\themes` folders.) Do this for each of your existing working environments: the `settings.xml` file is different per environment!

The second option must also be selected if this is a fresh installa-

tion and any files and folders (also any *settings.xml*!) in the target location can be deleted.

11. Click *Next* to start the actual deployment now. After installation has finished, the wizard should report a successful installation. Click *Finish*.



12. If you updated a Collections pre-3.0 version (for .NET 4.8) to a 3.0 or later version (for .NET Core 8.0), then copy back your *settings.xml* file to the new *\Configuration* sub folder underneath the physical path for the IIS application. Any custom background images and CSS can be copied back to the new *\wwwroot\Content\Backgrounds>Login\Custom* and *\wwwroot\Content\themes* sub folders.

## 5 Configuring Collections

After a first-time installation of a Collections system you will need to configure the installation further via the *settings.xml* file (and possibly *appsettings.json* for advanced options) to link the IIS application to the proper Axiell model application and such.

And after an update of a Collections pre-3.0 version (for .NET 4.8) to a 3.0 or later version (for .NET Core 8.0), you may need to tweak your existing *settings.xml* a bit.

1. In IIS, right-click the Axiell Collections application you'd like to configure and select *Explore* in the pop-up menu to open the installation folder in Windows Explorer.
2. Open the *\Configuration* folder.
3. In the *\Configuration* 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, where to write log files, and more. The *settings.example.xml* file contains an explanation for each possible setting, but some further explanation follows below.
4. Open your new or pre-existing *settings.xml* in an appropriate editor (like Notepad++ for example).

### Setting the base URL

For Collections 3.0 or later (for .NET Core 8.0), this option is **mandatory** and must always contain the public base URL (case-sensitive) of the Collections application. For example:

```
<BaseUrl>https://museum.com/CollectionsProduction</BaseUrl>
```

### 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.

## 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. Collections supports the A+ HTTP observatory level of this policy to enhance the security of Collections. It 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. Set the `Csp` option underneath the `Security` node to `enabled="true"` to profit from this extra security.

Optionally you can provide a list of trusted URLs (to web sites or domains with 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 needed. Wildcards are allowed. If no domains are whitelisted, the `BaseUrl`'s domain will be used.

For the GIS implementation, add the following URL to the whitelist:

```
<Whitelist>https://*.openstreetmap.org</Whitelist>
```

For a third-party IIIF image server, add the following URL to the whitelist:

```
<WhitelistImages>http://localhost:8184</WhitelistImages>
```

Note that underneath `<AlmSettings>`, for a IIIF image server also an `<ImageServerConfiguration>` section must be made and that for GIS a `<Gis>` section must be provided (see chapter 5.4).

The `<Headers>` sub section from the *settings.example.xml* should always be included in your *settings.xml*. These headers will be added to every reply the browser gets from Collections. The ones listed simply add some extra security, like telling the browser how to behave in certain situations.

## The Collections Scheduler

Set up of the "Collections Scheduled Service" is out-of-scope of this manual and is described in an internal Axiell document.

## 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.

---

## 5.1 Using session managers

### Using a single (non-tenanted) Collections application

In case of a single Collections 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">
  <LicenseInfo>
    <Holder>Customer name</Holder>
    <Count>10</Count>
    <Expires>2030-12-31</Expires>
  </LicenseInfo>
  <Setting Key="Languages" Value="en,nl" />
  <Setting Key="CacheScripts" Value="false" />
  <Setting Key="Path"
    Value="C:\Application\Primary\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. 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 model application 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.

## Using a multi-tenanted Collections application

In most cases you'll be using a single Collections model application with a "SQL" `SessionManager`, which makes the above settings all you need. In the case of a single model 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:\Axiell\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:\Axiell\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:\Axiell\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:\Axiell\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:\Axiell\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:\Axiell\Collections\
museums\MuseumD-app04\model\plusplus" />
</SessionManager>

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

```

Note that the application folder paths you set for a multi-tenancy configuration must always point to a specific application pbk folder, like `\plusplus` for example: you cannot point it to the application 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 ap-



plications 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.

### **DomainName and Servers are both optional**

If the machine is already joined with an Active Directory domain, then the relevant domain name will be used automatically. However, in scenarios where you wish to use a different domain, you can specify the `<DomainName>` element in this `<SessionManager>` configuration manually.

If the machine is not joined with any domain, you have the option to specify the domain name manually within the `<DomainName>` element. When the machine is not part of a domain, and if no `<DomainName>` element is specified, a list of LDAP servers must be provided manually in the `<Servers>` element. This manual specification becomes necessary when DNS SRV records are not available for automatic server discovery. However, even when a `<DomainName>` is specified manually or automatically, you retain the flexibility to provide server addresses manually if you prefer to use specific servers, rather than relying on automatic server detection.

On windows systems, you can specify the server port individually with a colon behind the server string. This does not work on Linux, there you can instead supply the port number using the optional `<PortNumber>` setting.

If DNS SRV records are available, the port numbers in the service records will be used. Automatic detection for SSL will also be performed. In that case there is no need to specify any port numbers in the configuration at all. If SSL detection is not available then the default port 389 and no SSL will be used. Override the relevant values if you know SSL is available.

If the servers list contains DNS server names, then set `<FullyQualifiedDnsHostName>` to `true` (default). If you specify an IP address, set the `<FullyQualifiedDnsHostName>` element to `false`.

The `<Connectionless>` setting enables UDP instead of TCP.

Aliases can be used to configure alternative domain names (or aliases). This list is automatically populated with NETBIOS names but you can put any alias in here. Example configuration:

```
<LdapConfiguration>
  <DomainName>example.com</DomainName>
  <Aliases>
    <Alias>example</Alias>
  </Aliases>
  <Servers>
    <Server>ldap.example.com:389</Server>
  </Servers>
  <PortNumber>389 or 636</PortNumber>
  <FullyQualifiedDnsHostName>true</FullyQualifiedDnsHostName>
  <Connectionless>false</Connectionless>
  <EnableSSL>false</EnableSSL>
  <TrustServerCertificate>true</TrustServerCertificate>
  <ProtocolVersion>3</ProtocolVersion>
  <CredentialKey>secret_store_key</CredentialKey>
  <ConnectionTimeout>5</ConnectionTimeout>
  <EnableCache>true</EnableCache>
  <CacheTimeout>30000</CacheTimeout>
  <PasswordChangeAllowed>true</PasswordChangeAllowed>
</LdapConfiguration>
```

---

## 5.2 SessionManager settings

### LicenseInfo

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. See chapter 6.2 for more information about the consequences of setting license information.

### Setting Key=Path

The mandatory `Path Value` for the "SQL" session manager should be set to the actual, full path to your model application folder containing the `.pbk` file. You can't use drive letters for shares: use the UNC path to the server in a format like:

```
\\myserver\Application\Primay\xplus.
```

Note that for the current Standard model application, the pbk sub folder is always named `\main`. See appendix A for more information.

In some situations, multiple application configurations are deployed to target specific areas of the system. Then, instead of setting the path to a specific model application pbk folder (like `\xplus`), you can also

set it to the model application root folder (containing multiple application pbk folders, like *\xplus*, *\library*, *\museum*, etc.) if you'd like to offer the user a choice of applications to log on to when opening Collections in the browser. Example:

```
<Setting Key="Path" Value="\\S01\Application\Production52" />
```

Note that this root folder is not allowed to contain any pbk file itself, otherwise you'll get a *Failed to start web server - invalid path* error when starting Collections.

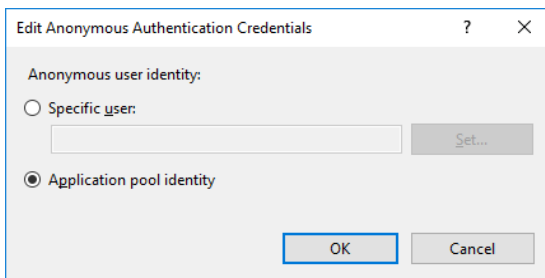
## Setting Key=ImageCacheFolder

Set the *Value* 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.

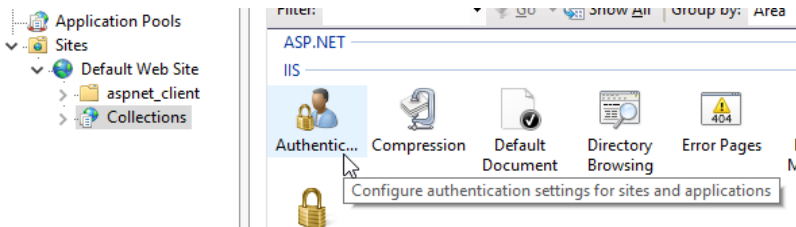
If no image cache folder is set, thumbnail images will be generated on-the-fly when requested.

This setting is only relevant if image fields in the Axiell Collections application are configured to get images from a folder.

\* Note that the application pool user must have sufficient access rights to allow the implicit creation of a new sub folders 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.



## Setting Key=Preload

The `Preload 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)
- `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.

## Setting Key=Domain...

The **DomainController** and **DomainContainer** settings are useful for tightening up security for local demo installations that are not connected to the AD domain. For customers with complex Active Directory configurations – multiple domains and name servers with trusts enabled – Collections may need 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 (DomainContainer) 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.

If no values are specified, the defaults will be used.

**DomainUsersOnly:** with this option set to `false`, if a Collections user can't authenticate against Active Directory (because it fails or isn't available) 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 or development environments 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.

**DomainSSLEnable** enables SSL (Secure Sockets Layer) support when connecting to the Active Directory server.

**DomainSSLPort** is the port number to use (if enabled with `DomainSSLEnable`). The default is 636.

**DomainSSLTrustCertificate** should be set to `true` if you are using SSL and a self-signed certificate: no trust verification will take place. Set it to `false` if the SSL certificate comes from a trusted provider (in other words: if you paid for it) so that trust verification takes place.

**DomainAllGroups** should be set to `true` if AD distribution groups must also be included. Set it to `false` to only include security groups.

**DomainLogUsers** should be set to `true` if log entries are to be created with the user's login ID, groups and application ID.

## Setting Key=Languages

`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 Axiell 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.)

## Setting Key=AlwaysPromptBeforeSave/ EnableResultSetEdit

By default, records can be edited in both the *Record details* and the *Result set* view. The *Result set* context toolbar only offers an *Edit mode* icon and always saves an edited record without asking for confirmation, whichever way you close or leave the edited record, while the *Record details* context toolbar has a *Save record* icon (that never asks for confirmation) and an *Edit mode* icon that does always ask if you'd like to save your changes or not, via a so-called save prompt.

If you're not happy with that and you'd like to limit the number of options to close, save or leave an edited record to just the *Edit mode* icon in the *Record details* context toolbar and to leaving an edited record in the *Record details* view by clicking a different record in the *Result set*, in order to always force a save prompt, then you can simply set the optional `AlwaysPromptBeforeSave` setting to `true`.

```
<Setting Key="AlwaysPromptBeforeSave" Value="true" />
```

This will cause the *Edit mode* icon to disappear from the *Result set* context toolbar while in the *Record details* context toolbar the *Save record* icon will disappear (and you'll have to save records by clicking the *Edit mode* icon again). This will apply to all data sources.

A good reason to set it to `true` is to prevent data issues that may arise from allowing edits in the *Result set* because certain data validations won't be executed there. And for the current Standard model application it is even a requirement for the consistency of your data that you set this option to `true`!

However, a disadvantage of this option set to `true` is that there's no *Save* button in the *Record details* view any more. This can be fixed with a setting introduced with Collections 3.0, to switch off editing in the *Result set* completely:

```
<Setting Key="EnableResultSetEdit" Value="false" />
```

`False` switches editing in the *Result set* off, `true` enables it (but leaving out the this setting altogether also means editing is switched on.)

In the *Record details* view, on the other hand, with this option, by default a *Save record* icon will be present next to the *Edit mode* button, so can choose how to close and save an edited record in there.



Clicking the *Save record* button saves the record immediately, while clicking the *Edit mode* button, still asks for confirmation.

You can still use the `AlwaysPromptBeforeSave` setting to `true` (possibly in combination with the new `EnableResultSetEdit` setting) to hide the *Save record* button, but the `EnableResultSetEdit` setting to `false` combined with the `AlwaysPromptBeforeSave` setting set to `false`, does allow you to disable editing in the *Result set* while still having a *Save record* button in the *Record details* view.

```
<Setting Key="AlwaysPromptBeforeSave" Value="false" />
<Setting Key="EnableResultSetEdit" Value="false" />
```

However, you can also leave out the `AlwaysPromptBeforeSave` setting altogether as this will enable the *Save record* button by default.

## Setting Key=`UseOriginalMediaFileName`

With this option set to `false` (which is also the default when the option hasn't been set), Collections will generate a GUID (Globally Unique Identifier) and use that as the file name for the stored file. In Collections, 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 its own 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 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. This means that names of uploaded files must be unique in this case. 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>>.

## Setting Key=DashboardVisible

This option can be used to enable or disable the dashboard for Axiell Move missions or Workflow. When the `DashboardVisible` key is not there, the dashboard will not become visible (the same goes for setting the key's `Value` to `false`).

## Setting Key=ConnectionString

Use this optional setting to define extra SQL connection string parameters. Use regular connection string syntax:

```
<key>=<value>;<key>=<value>;<key>=<value>;...
```

A list of valid parameters can be found [here](#). Example:

```
<Setting Key="ConnectionString"  
Value="TransparentNetworkIPResolution=true" />
```

## Setting Key=DefaultExpand

Set this option to `false` to disable [the default expand on searches on inherited fields](#) in Collections.

```
<Setting Key="DefaultExpand" Value="false" />
```

## Setting Key=Theme

In principle it is possible to create a custom CSS for Collections to alter certain aspects of its appearance. Such a custom CSS should be placed in a new sub folder in the *Content\themes* sub folder of your Collections installation. The new sub folder and .css file should have the same name, apart from the file extension. Example:

```
<Setting Key="Theme" Value="custom_theme_name_without_extension" />
```



## Setting Key=CacheScripts

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.

## DesignatedJobs

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).

In the *Output formats* dialog, the *Print* button and the following *Select printer* dialog will only be present though if available printers have been set up here in settings.xml file first. This option can only be implemented if you have a local installation of Collections. This setup is discussed in chapter 5.3.

It is also possible to associate one or more of those preset printers with particular output formats, so that users can never select an inappropriate printer for the chosen output format. This way, starting an output format for e.g. printing certain labels can automatically be directed to the proper label printer, while for A4 Word templates maybe only the default office printer can be used. This is especially useful if your organisation has multiple regular and/or label printers.

The implementation is such that after clicking the *Print* button, the user will first have to choose one of the allowed destination printers and click the *OK* button to start printing, even if only one printer is associated with the output format so that it is clear to which printer the output will be sent.

So besides the earlier mentioned settings to specify the available printers to begin with, settings similar to the following should go in the desired <SessionManager> node, so you can specify tenant-specific output format (job)/printer (device) mappings. Example:

```
<DesignatedJobs>
  <DesignatedJob id="Object catalogue/my Excel report">
    <Devices>
      <Device id="DefaultPrinter"/>
      <Device id="PDF"/>
      <Device id="Email"/>
    </Devices>
  </DesignatedJob>
  <DesignatedJob id="Object catalogue/Inventory list">
    <Devices>
```

```

    <Device id="PDF"/>
  </Devices>
</DesignatedJob>
</DesignatedJobs>

```

The above assumes that there are three devices configured already (*DefaultPrinter*, *Email* and *PDF*) in the `<Devices>` section. The `id` attribute value of the `<DesignatedJob>` node should reference the English name of the data source in which the output format appears (Object catalogue in this example) and behind the slash you should specify the English name of the desired output format (in this example, one is called *my Excel report* and the other is called *Inventory list*). So with this setup, the *my Excel report* job can only be printed to the *Defaultprinter* or *PDF* device or sent as an attachment via e-mail to a fixed list of recipients and the *Inventory list* output format can only be printed to the *PDF* device.

This job/device mapping should only list the exceptions; only output formats which target one or more specific printers should have a mapping. All other output formats, for which the user should be allowed to pick any printer, should not be mapped.

---

## 5.3 Setting up direct printing

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) or send it through e-mail to a fixed list of recipients.

In the *Output formats* window, the *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. The direct printing option (`Type="Printer"`) 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>
<Device Id="Email" Type="Email">
    <Name>
        <Text lang="en">E-mail to fixed recipients</Text>
    </Name>
    <Settings>
        <Server>
            <ServerType>SmtP</ServerType>
            <Address>smtp.ourmuseum.com</Address>
            <Port>25</Port>
            <EnableSsl>false</EnableSsl>
        </Server>
        <Format>Text</Format>
        <Sender>info@ourmuseum.com</Sender>
        <Subject>More exciting news</Subject>
        <Recipients>
            <Recipient>john@ourmuseum.com</Recipient>
            <Recipient>lisa@ourmuseum.com</Recipient>
        </Recipients>
        <Content>See attached file</Content>
    </Settings>
</Device>
</Devices>

```

The available settings are the following:

**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.)

**Device Type:** The type of the printer:

- Pdf = Print to a PDF file (no PageLayout settings required).
- Printer = Print to a printer.
- Email = Send resulting document as an attachment to a fixed list of recipients.

**Name:** Localised UI name of the printer for displaying to the user. This is what appears in printer selection drop-downs 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:** For `Printer` and `Pdf`, 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:** Specify page margins in mm.
- **DefaultFont:** Default font to use when the report doesn't specify a font. Used as defaults for text or HTML.

**Server:** for Email, specify you SMTP server details.

**Format:** for Email, specify if the e-mail body text is to be laid out as Text or Html.

**Sender:** for Email, specify the sender e-mail address of your institution.

**Subject:** for Email, specify the subject text (title) of e-mails sent this way.

**Recipients:** for Email, specify each recipient e-mail address in its own <Recipient> node underneath <Recipients>.

**Content:** for Email, specify any fixed body text for the e-mail.

---

## 5.4 AlmSettings (for GIS and image server)

### ImageServerConfiguration

The processing of images retrieved by the Collections Media Viewer can be handled in two ways:

- by the **browser**: this requires no extra configuration in Designer and is expressed in *settings.xml* file by the following default section underneath the <AlmSettings> node:

```
<ImageServerConfiguration>
  <Viewer>Browser</Viewer>
</ImageServerConfiguration>
```

- by **IIIF** (pronounce: triple I F) image processing. Include a section like the following and provide values where appropriate:

```
<ImageServerConfiguration>
  <Viewer>IIIF</Viewer>
  <ImageServerUrl>http://{localhost}/imageserver/
    ipsrv.fcgi</ImageServerUrl> <!-- the URL to the
      IIIF image server -->
  <FolderMappingList> <!-- optional, useful for
    enterprise environments -->
    <FolderMapping> <!-- repeat for each branch -->
      <LowerLimit>1</LowerLimit> <!-- priref of lower
        limit of branch dataset -->
      <UpperLimit>10000000</UpperLimit> <!-- priref
        of upper limit of branch dataset -->
      <Folder>museum-01</Folder> <!-- name of the
        branch sub folder containing the branch
        specific Axiell model application -->
    </FolderMapping>
```

```
<FolderMappingList>
</ImageServerConfiguration>
```

For a third-party IIIF image server, remember to add the following URL to the whitelist:

```
<WhitelistImages>http://localhost:8184</WhitelistImages>
```

For the full story about setting up a IIIF image server, click [here](#).

## GIS

GIS (Geographic Information System) functionality for Axiell Collections allows for an application to be customized to allow users to register locations and areas by means of their location on a geographical map and also to search for records by a variety of map manipulations to select places, precise spots or areas of different shapes. It depends on your model application version whether such GIS fields are present already. In *settings.xml* you'll need to configure a section like the following underneath the `<AlmSettings>` node if GIS fields are indeed present in your application.

When working with the map display in Collections, you'll also see an entry field with a *Search* button next to it. This could allow the user to enter a place name and have the map zoom in on that location automatically. This is so-called geocoding functionality and is usually not included in a map service and must be licensed separately by a third-party like Google or other geocoding web service.

The `DefaultMapProvider` for the current geojson GIS implementation is always `OpenStreetMap`: any other value will be ignored. If you have a geocoding provider then specify it as a `GisProvider`, enter any acquired `ApiKey` and set this provider as the `DefaultGeocodingProvider`; otherwise set it to `None`. For the current GIS implementation you only need to include a `GisProvider` if it indeed provides geocoding. Example:

```
<Gis   DefaultMapProvider="OpenStreetMap"
      DefaultGeocodingProvider="Google">
  <!--optional geocoder caching database: -->
  <DatabaseSettings>
    <ConnectionType>SqlServer</ConnectionType>
    <Server>MySQLSERVER</Server>
    <Database>geolocations</Database>
    <User />
    <Password />
  </DatabaseSettings>
  <GisProvider>
    <Provider>Google</Provider>
```

```

    <Enabled>true</Enabled>
    <!-- If you have obtained an API key for a
         geocoding service, be sure to enter it in
         the <ApiKey> node: -->
    <ApiKey>Aiz9YwB9wSj08096FtB0aEjxZudrHsS4</ApiKey>
    <ApiSignature />
    <CacheType>Memory</CacheType>
  </GisProvider>
</Gis>

```

## The optional DatabaseSettings node

The `<DatabaseSettings>` node, underneath `<Gis>`, is meant to set an optional database to store temporary Collections settings such as the geocoder cache. To improve map display performance and to not overload the map provider with too much requests, you can (optionally, but still recommended) configure a SQL database to contain geographical location names and coordinates that were marked on the map once before for selected fields in displayed records. This database doesn't necessarily need to be on the same server as Axiell Collections and it can be shared by multiple applications run by Collections. Just create a new database in SQL Server like you would normally do. You won't need to create any tables manually: Axiell Collections will automatically do that for you the first time the map functionality is used. In *settings.xml*, then configure the `<DatabaseSettings>` node and replace `MySQLSERVER` by the name of your own SQL Server and `geo-locations` by the actual name of the database you created for storing these place names. `<ConnectionType>` must then be set to `SqlServer`. User name and password only need to be specified if you use SQL Server authentication (instead of Active Directory authentication) to access the database. If you choose not to use a caching database, set `<ConnectionType>` to `None`. Don't leave `<ConnectionType>` empty otherwise you'll get a runtime error when using the GIS functionality. If you are not using a caching database, you can also leave out the entire `<DatabaseSettings>` node and its sub nodes.

## Whitelist openstreetmap

For the GIS implementation, remember to add the following URL to the whitelist in the current *settings.xml*:

```
<Whitelist>https://*.openstreetmap.org</Whitelist>
```

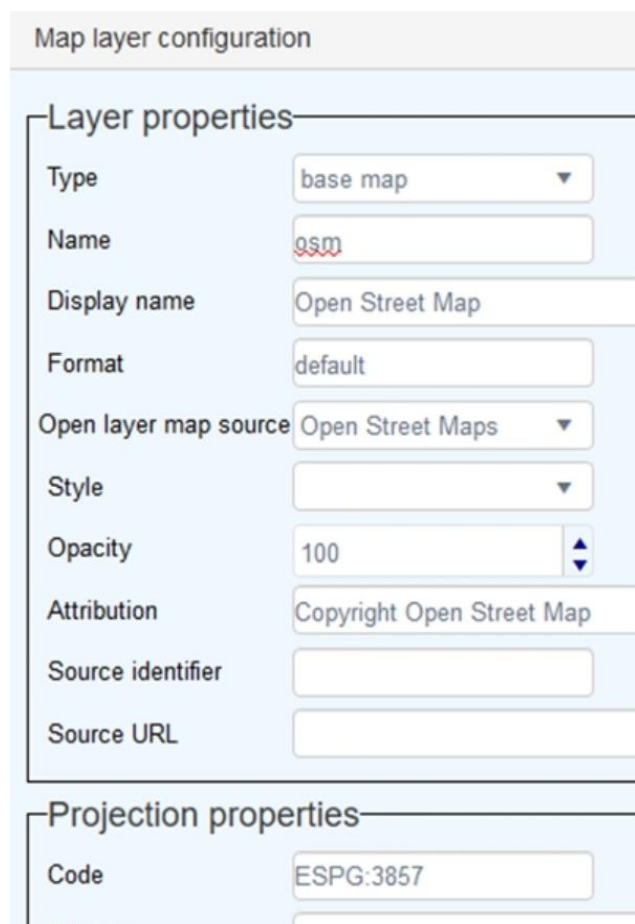
## Creating GIS fields or using different maps

The latest Collections application versions already have appropriate GIS fields, but if you still need to create those or if you need to allow the use of different maps and overlays, please click [here](#) for the full story about GIS.

## Creating Maps records in Collections

Next, you'll need to create a record in the *Maps* data source in Collections for each base map and overlay you would like to make available to users. At least create a record for Open Street Map, with the data as shown in the screenshot below. (Note that for this particular base map a *Source URL* is not required because the OpenLayers library used for the GIS functionality already knows where to find the relevant web service.)

If you need to know more about the different fields in *Maps* records, please see bullet 16 in the [full GIS topic](#).



The screenshot displays a web form titled "Map layer configuration". It is divided into two main sections: "Layer properties" and "Projection properties".

**Layer properties:**

- Type: base map (dropdown menu)
- Name: osm (text input field)
- Display name: Open Street Map (text input field)
- Format: default (text input field)
- Open layer map source: Open Street Maps (dropdown menu)
- Style: (empty dropdown menu)
- Opacity: 100 (text input field with a vertical slider to its right)
- Attribution: Copyright Open Street Map (text input field)
- Source identifier: (empty text input field)
- Source URL: (empty text input field)

**Projection properties:**

- Code: ESPG:3857 (text input field)



---

## 5.5 Default Record details view screens filter

The *Filter mode* option in the *Record details settings* in Collections allow the user to choose between showing *All* screens (panels) and *HasData* to hide empty screens. In *settings.xml* you can set the default as follows and choose from three options in total.

```
<Defaults>
  <!-- Default Record details view screens filter:
        All, HasData, RecordType -->
  <RecordEditorFilter>All</RecordEditorFilter>
</Defaults>
```

---

## 5.6 Logging

From Collections 3.0 the `Logging` setting in *settings.xml* will be ignored. Instead, logging has to be set up in a settings json file. Please see chapter 5.12 for more information about that.

---

## 5.7 Theme Name=dark

Leave the `<Theme Name="dark" />` setting as is. This setting is for internal use.

---

## 5.8 SSO, OpenId and SAML

Configure different ways of signing in to Collections. These topics fall outside the scope of this manual and are described in internal Axiell documents.

---

## 5.9 Configuring the path to uploaded templates

Axiell Collections allows end-users to upload and run their own output format templates. These templates are stored in a designated folder and the location of that folder is set in a text file that sits within the Axiell Collections model application `\texts` sub folder (e.g. *Application\Primary\texts*).

First check if a *Worddoc\uploadedtemplates* sub folder is already present within the application folder structure. If not, create it using Windows Explorer.

Now open *reports.txt* in a text editor like Notepad++ and replace the line with `1no_path_yet` by `1` followed directly (without space) by the UNC path to your uploaded templates sub folder. This path must end with a backslash. (Without this change, uploaded files cannot be deleted by the user from within Collections.) Save the file.

For this to work, the application pool user must also have delete permission in the *uploadedtemplates* sub folder in Windows.

---

## 5.10 Maximum file upload and download settings

Files (images, documents, zip files etc.) larger than 2GB can be uploaded and downloaded 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 en 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 breaks up large files into chunks of about 1 MB only. The current property only re-

ally 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.

## Considerations

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 free 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 succesful 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 *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 *FileCache\chunks* subfolder: you can do

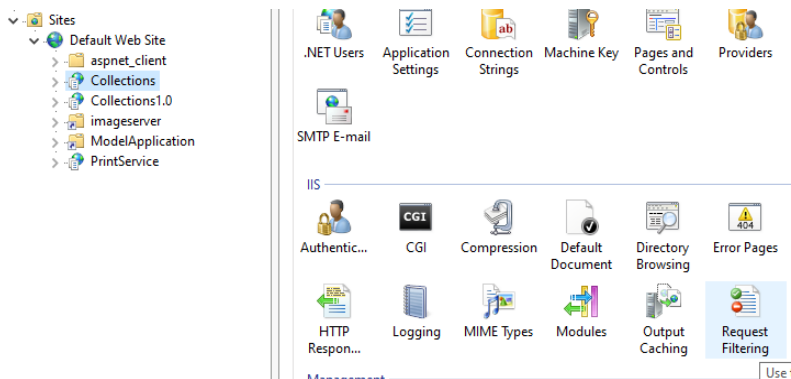
that in Windows Explorer. Make sure no-one else is working in Collections when you do that.

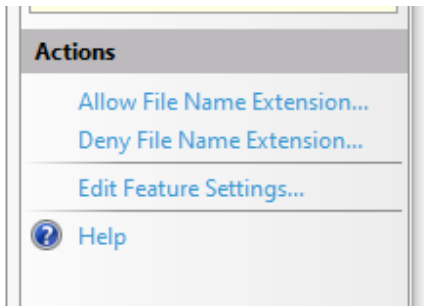
\* The `<add key="GlobalStateFolder" value="" />` property is a Collections-specific setting added by a Collections installation to the *web.config*. For the *value* you could enter a path to an appropriate cache folder (other than the default local *File-Cache\chunks* subfolder of your Collections installation). 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 may be a performance penalty. IMPORTANT: the Collections Application Pool Identity must have *Modify* permissions to that alternative caching location.

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.

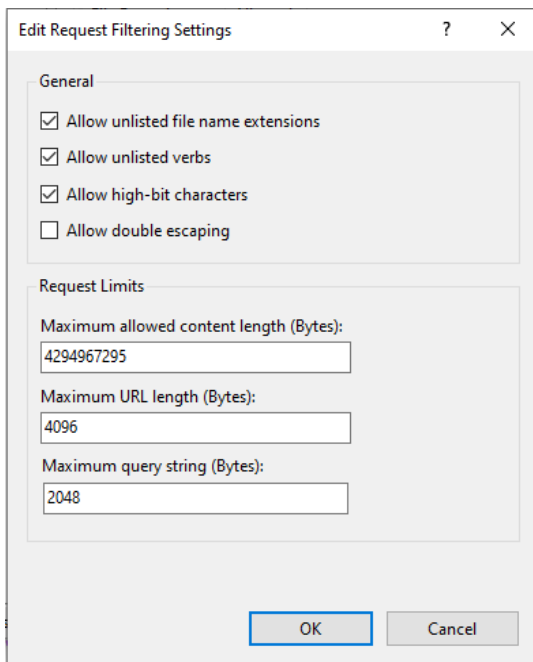
## Using IIS to make the desired settings

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.





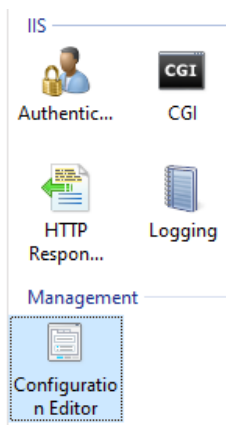
2. After an update of Collections or a new Collections installation, the *Maximum allowed content length* (in bytes), for the Collections site at least, shows the default setting for this version of IIS and cannot be set higher. If it had been set to a lower value than the maximum previously, you can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



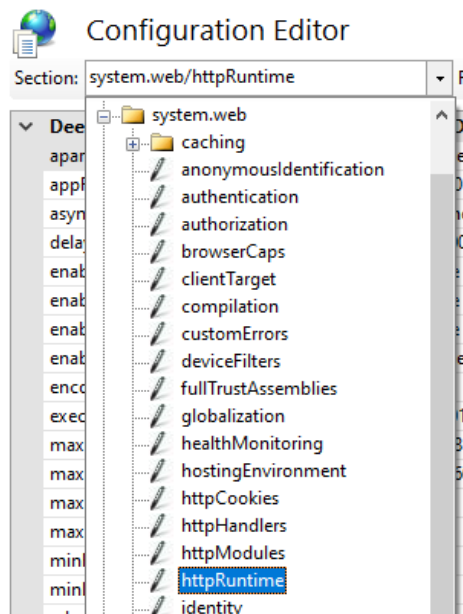
The `maxLength` property can be set via IIS too:

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 behind *maxRequestLength*. Enter it in kilobytes, e.g. 6000000 kilobytes (representing 6 GB). It can be as high as you like (max 2

TB, but files that size should not be uploaded through the Collections interface).

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.

---

## 5.11 Access rights

Axiell model applications must be secured (typically on database and field level) using their access rights mechanism to restrict unauthorized users from access to certain or all data, because with a single Axiell Collections installation per working environment, any domain user can in principle log on and access the application if no 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 table .inf file 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 database tables, and further specify their access rights with the access rights mechanism in the model application.

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 table *Authorisation types* have not been implemented, while *Specified rights* has been. So when preparing a model application for Axiell Collections, you may want to check these settings in case a different access rights solution is required.

For more information, click [here](#).

---

## 5.12 Editing settings json files (optional)

Most common settings reside in the *settings.xml* file in the \Configuration sub folder. Some advanced settings, like logging, e-mail server configuration and multi-factor authentication, need to be done in an applicable settings json file, which can be edited or created in a text editor like Notepad++. (Any logging setting in *settings.xml*

will be ignored.) There are different json files in different locations possible.

Json settings files are part of the configuration system that is build into .NET Core 8.0 and they allow for a cascading configuration. The format of the files is the same, but they are loaded into memory in a certain order: later loaded files can override settings in previously loaded files.

Collections attempts to load the following files (as far as they are present) in the following order when it starts:

1. **appsettings.json** (should never be edited manually)
2. **appsettings.{Environment}.json** (can be used in combination with an environment variable to switch between different settings files, depending on if you are currently using Collections as a production, staging or development environment)\*
3. **settings.xml** (the pre-existing Collections settings file)
4. **settings.json** (an optional file for certain global settings)

1 and 2 are standard .NET Core configuration files and they need to be located in the root folder of the Collections installation, while 4 was added by Axiell.

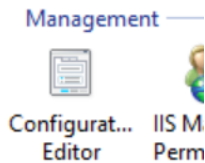
3 and 4 can be placed in different folders. They are searched for in the following locations (in order from top to bottom):

1. `"/Configuration"` (the *\Configuration* sub folder in de Collections installation)
2. `"/Settings"` (sub folder not present by default)
3. `"/App_Data"`
4. `"%APPDATA%/Collections"`
5. `"%APPDATA%/Axiell/Collections"`

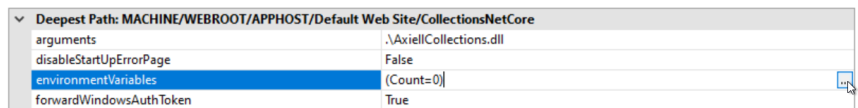
\* In **appsettings.{Environment}.json**, *{Environment}* is a placeholder for *Production*, *Staging* and *Development*, so you can have e.g. an *appsettings.Production.json* and an *appsettings.Development.json* for example. You may want to create different environment settings files if you'd like to be able to switch fairly easily between certain settings for your Collections installation, in particular logging settings, depending on whether you are currently developing or debugging something or not. Which of the *appsettings.{Environment}.json* files will be used (if present) by your Collections installation, is controlled by the `ASPNETCORE_ENVIRONMENT` variable, which when empty defaults to "Production". So if you want the Collections installation to search for the *Staging* or *Development* configuration file instead of the default *Production*, you'll have to set this variable, which can



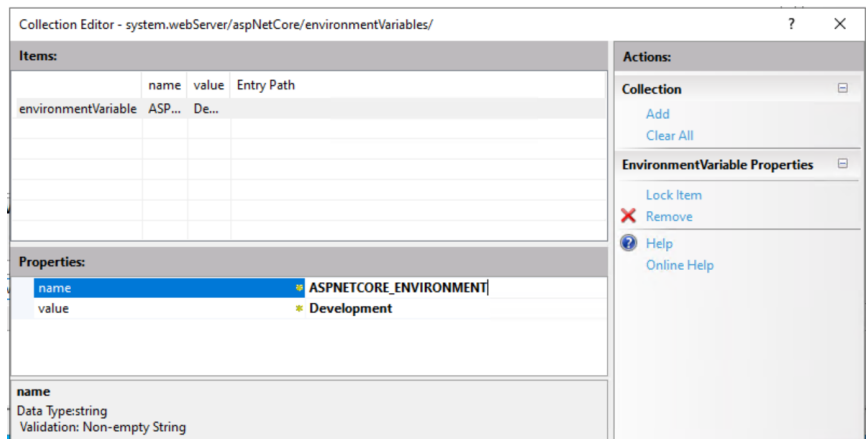
be done via the *Configuration editor* for the Collections application in IIS.



In the editor, select the *Section* "system.webServer/aspNetCore". Then click the ... button behind *environmentVariables*.



In the *Collection editor*, then click *Add* and fill in the following properties behind *name* and *value*:



Then close the window. The environment variable has been set (to *Development* in this example).

## Setting up Collections logging

Log files may help our programmers to find out what the cause of a problem is, so it's good to set up logging. From Collections 3.0, any logging configuration in *settings.xml* will be ignored. Instead, the logging configuration from the active *appsettings.{Environment}.json* will

be used. (Better not put a logging configuration in *settings.json*.) By default, the active file is *appsettings.Production.json*, which comes with a ready-to-use logging configuration (excluding debugging information), but there's also an *example.appsettings.Development.json* file in the root folder, containing a logging configuration for debugging.

The default *appsettings.Production.json* looks something like this:

```
{
  "NLog": {
    "throwConfigExceptions": true,
    "targets": {
      "async": true,
      "logfile": {
        "type": "File",
        "fileName": "logs\\prod-${shortdate}.log",
        "layout": "${long-
date}|${level:uppercase=true}|${logger}|${message}|${onexception:${n
ewline}}${exception:format=ToString,StackTrace}}"
      },
      "console": {
        "type": "Console",
        "layout": "${long-
date}|${level:uppercase=true}|${logger}|${message}|${onexception:${n
ewline}}${exception:format=ToString,StackTrace}}"
      },
      "errorfile": {
        "type": "File",
        "fileName": "logs\\prod-errors-${shortdate}.log",
        "layout": "${long-
date}|${level:uppercase=true}|${logger}|${message}|${exception:form
at=toString}"
      }
    },
    "rules": [
      {
        // Discard Trace and Debug logs from Microsoft libraries
        "logger": "Microsoft.*",
        "maxLevel": "Debug",
        "final": true
      },
      {
        // Log Info and above from Microsoft libraries and all oth-
er libraries to console and logfile
        "logger": "*",
        "minLevel": "Info",
        "writeTo": "console,logfile"
      },
      {
        // Additionally log errors to a separate file
        "logger": "*",
        "minLevel": "Error",
        "writeTo": "errorfile"
      }
    ]
  }
}
```

```

    }
  ]
}

```

This configuration specifies NLog (at the top) as the logging framework to use. This framework makes it easy to capture, filter, and route log messages to different outputs like console and files.

First three “targets” are specified, named *logfile*, *console* and *errorfile*, but they can be named differently and/or you can add or delete targets. Each target specifies whether its output should go to `File` or `Console`. The `fileName` contains the relative path (relative to the root of the Collections installation) to a *logs* folder and a format string for the name of the log file to create. The `layout` specifies a format string for each log line that will be output. In this case, two different log file types can be produced.

Then, a rules section specifies which log events (filtered by level, `minLevel` or `maxLevel`) from which programming libraries must be written to which targets. Levels which can be used (in order of log severity) are: `Trace`, `Debug`, `Info`, `Warn`, `Error` and `Fatal`. Since `Trace` and `Debug` are rather verbose, you usually don’t want these in your log files to prevent them from becoming very large very soon. The rules apply from top to bottom but execution of the rules stops when a “`final`”: `true` is encountered in a matched rule.

The first rule targets any logger whose name starts with `Microsoft.*` (e.g., `Microsoft.Hosting`, `Microsoft.AspNetCore`) for events with levels `Trace` and `Debug` and makes sure that all log messages from Microsoft libraries at those levels are filtered out, so those don’t make it into the log. Higher levels (`Info`, `Warn`, `Error`, `Fatal`) are not affected by this rule. Once this rule matches, no further rules are evaluated for these loggers because of “`final`”: `true`.

The following two rules target all loggers, including Axiell libraries (\* is a wildcard). They use different minimum event levels and write the log events to two different files (for later analysis), where the second rule also writes the events to the console (for real-time monitoring).

## Multi-factor authentication setup

Two-factor aka multi-factor authentication (MFA) can be used as an extra security layer for users to log into Axiell Collections, so that after entering a user name and password on login the user is sent an e-mail containing an extra authentication code to enter before actually getting access to Collections. Collections 3.0 or higher requires the following setup.

In the .pbk, set the (*Two-factor authentication*) *Provider* option to *Email*. If any SMTP server settings are already present on the same properties tab, you can leave them as they were, although these are not required for MFA in the Collections 3.0 setup any more and will even become redundant in future versions of Collections where e-mail servers can only be configured in settings files. The *Authentication method* option, at the top of the *Application authentication* properties tab, can be left as is too, as long as it's either set to *Adlib database*, *Active Directory* or *SingleSignOn*. The authentication method requires a registered e-mail address with the other credentials for each Collections user (and e-mail addresses stored in a *Users* data source for *Adlib database* authentication should not have a "mailto:" prefix).

Then, for authentication types *Active Directory* and *Adlib database*, the easiest way to get MFA up and running is to add a *settings.json* file in the \Configuration sub folder of your Collections installation, with the following content:

```
{
  "EmailClients": {
    "MfaClient": {
      "Host": "myorganisation.mail.outlook.com"
    }
  },
  "Authentication": {
    "Multifactor": {
      "Email": {
        "Client": "MfaClient",
        "From": "noreply@myorganisation.com",
      }
    },
    "UserStore": {
      "Type": "Policy",
      "MfaPolicy": {
        "RequireMfa": true,
        "DefaultMfaMethods": [ "Email" ]
      }
    }
  }
}
```

If a *settings.json* file already exists, you can add the above settings to that file.

The global `EmailClients` section defines a named e-mail client. If you want, you can name "MfaClient" differently (you can have unlimited e-mail clients). The `Multifactor` section, inside of the `Authentication` section, configures which e-mail client and "from" address to use for Email MFA. To force enable Email MFA for all users if not enabled in

the `.pbk`, a MFA policy section is needed adjacent to the `Multifactor` section.

Just change the `Host` to the SMTP server name of your organisation and the `From` e-mail address to an applicable one, to ready your *settings.json* file quickly.

The `\EmailTemplates` sub folder of your Collections installation contains default e-mail templates in four interface languages. You can change their contents as you like or add extra templates in other languages: you cannot use any existing e-mail templates from pre-3.0 versions. Keep backups of any edited templates in a separate location in case they ever get overwritten with the default templates again.

Recycle your IIS application pool and MFA should function.

For more advanced MFA setups, we still recommend putting those settings in a *settings.json* file in the `\Configuration` sub folder. Do see *example.appsettings.json* in the root Collections folder for documentation about all possible MFA settings.

(On a related note: password resetting is not functional in Collections 3.0 yet.)

### **Assorted extra configuration in settings.json**

For examples and documentation for all possible settings, see *example.appsettings.json* in the main Collections installation folder, but do not copy any configuration to *appsettings.json*, but do all your customizations in *settings.json*.

The following might be of extra interest:

- You can have multiple active authentication providers set in *settings.json*. You'll then see them all in the login screen. The `ProviderId` just needs to be a unique string that you can make up yourself.
- To force Collections applications configured with "None" authentication to use LDAP, you can add the `"NoneMeansLdap": true` setting in *settings.json* if you don't want to change the `.pbk` in this respect.
- In *settings.json*, you can add a `SupportLink` with a URL to your own support page shown in the login dialog when the user indicates they need help.

- In the default *appsettings.json*, `"DisableAutomaticSsoRedirect": true` will cause users not to be automatically redirected to the SSO provider when attempting to log out, if only one is configured. The problem with the previous `false` default, was that you couldn't log out when using SSO, because logging out automatically tried to log you in via SSO again. Using `true`, there's a button on the login screen which you must click to login via SSO, so this stops the automatic redirect process. If you want to change this setting to `false` still, then do that in *settings.json*, not in *appsettings.json*.
- In *settings.json*, `"IgnoreLegacyProviders": true` will cause legacy authentication providers (None, ActiveDirectory, AdlibDatabase, AdlibApplication) to be ignored, even if configured with Designer. Use this to disable legacy providers if you only want SSO.
- In *settings.json*, `"AccountStorageEnabled": true` will enable local account storage together with a *Remember me on this computer* checkbox on the password login screen.

## 6 Collections login

The installation is now complete. If you haven't recycled your Collections IIS application pool(s) after changing any settings, then do that now.

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 Collections IIS 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 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: `mymachine\user.`)



From 3.0, normal login is a two or three-step process: first you enter your user name, possibly preceded by a domain. Then click *Next*, en-

ter your password and click *Next* again to log in. If two-factor (aka multi-factor or MFA) authentication has been set up as well, you will receive an e-mail with a verification code to enter in a third step. Note that a password is always required. If in previous versions of Collections you could always log in with a user name only (without having to enter a password) and from Collections 3.0 you are asked for a password, then you have to assign your user name a password. You'll have to use that password to log in from now on.

You can use the language button to switch between interface languages.

The *I can't access my account* option opens either the Axiell support page or a support page of your own institution, in a different browser tab.





In the next screen, your current login name (possibly preceded by an authentication domain name) will be displayed, plus the used authentication method (Active Directory/LDAP in this example).

Normally the password won't be readable but if you click the eye icon, you can see what you're typing. It's safer though to keep it unreadable as much as possible, so that no-one else can see your password accidentally.

By default, your browser may already offer to remember your user credentials, so then you don't need to mark the *Remember me on this computer* checkbox, but if it doesn't you can use the checkbox. (This method of remembering your credentials will also remember the used authentication method.)

Another reason to tick the checkbox is if you sometimes log into this instance of Collections with different user credentials. Having the checkbox ticked will then remember the different user credentials: the *I want to sign in with a different account* in the first and second step of the login process link will become clickable (displayed in blue) as soon as at least one pair of user credentials has been stored via the marked checkbox. Clicking the link will open a list of credentials that you can use to log in. In that list you can also remove individual credentials or remove all of them from this list.

If you don't see the *Remember me on this computer* checkbox, it may be because it has been disabled via *appsettings.json*.

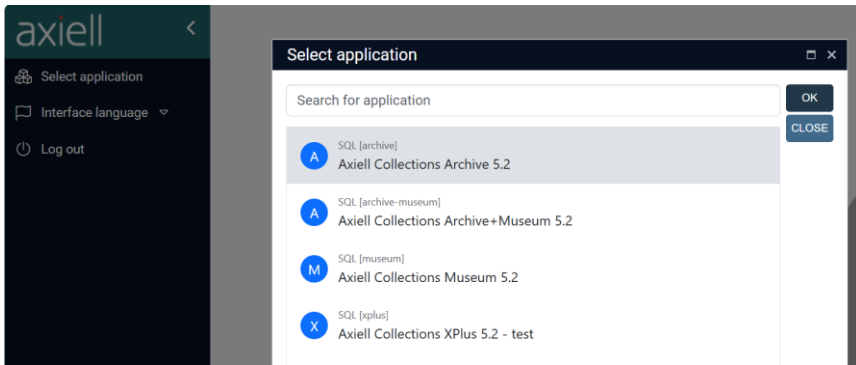
The *I forgot my password* option can only be used when properly set up. Password resetting isn't functional in Collections 3.0 though.

If automatic logging-in has been set up via a third-party authentication service (aka Single Sign-on) you may see a *Sign in with ...* button instead of the user name and password entry fields. Simply click the button to log in.



## 6.1 Application selection

If in Collections, users have access to multiple applications (in multi-tenancy setups or multi application setups), you'll notice that from Collections 3.0 users have to select the application to work in only after they've logged in (making it possible to switch between applications without having to log out and log in again). In that case, the *Select application* window will open with the applications available to the user. Each application will be indicated by its title, the name of folder in the file system which holds the application definition in between square brackets, (for advanced users) the session manager id ("SQL" in this example) and an icon. If you didn't set up any specific icons, users will just see a blue circle with the first letter of the folder name.



Users just have to select the desired application and click *OK* or double-click the desired application to open it. If the list of applications is very long, one can use the *Search for application* box to enter (part of) the name of the application to filter the list and find the application they are looking for more quickly.

If users click the *Close* button in the window when it was opened just after logging in, no selection is made and the user is left with only three menu options. Clicking the *Select application* menu opens the window again.

As a matter of fact, the *Select application* menu is always present if users have access to multiple applications and they can select a different application at any time, even if they are already working in one. And log out/in again isn't necessary.

The user can even open the URL (without `?tabid=<id_string>` in the URL) to Collections on a different browser tab and work in a different application on that tab. Then the user will still be working in a single user login session.

The *Select application* menu and its window won't appear when the user is only allowed access to a single application.

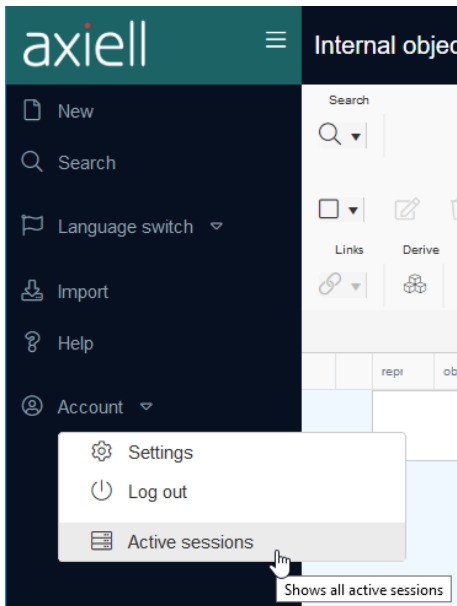
**Technical information:** if you place an *icon.png* file (or other image file type named "icon") next to the .pbk file in the Collections model application sub folder, this icon will be displayed in front of this application in the *Select application* window. (For the icon file, all web formats are supported: .gif, .svg, .jpg, .webp, .png, .ico. The image will automatically be framed and resized for display.)



SQL [xplus]

Axiell Collections XPlus 5.1

## 6.2 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

applications, this means that the admin can see all users logged into any of those applications.

The *Active Sessions* window can show 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.

Active Sessions for

Application Id

User	First Seen At	Session Alive At	User Awake At	Source	Info
Application Id: archive					
martijn	2019-09-23 11:51:48	2019-09-23 12:40:16	2019-09-23 12:19:39		Archives (catalogue)
erik	2019-09-23 11:49:47	2019-09-23 12:39:43	2019-09-23 12:40:57		Internal object catalogue
edwin	2019-09-23 11:56:30	2019-09-23 12:40:32	2019-09-23 11:56:30		

3 ITEMS

OK

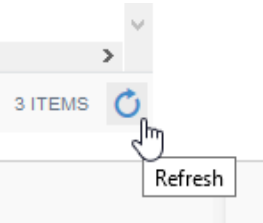
CANCEL

MESSAGE

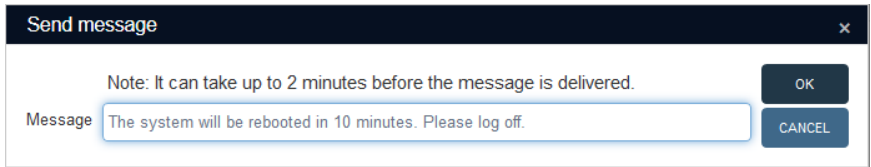
EVICT

ENTITLEMENT

The *Active Sessions* window is not refreshed automatically. Click the *Refresh* icon in the bottom right corner of the grid view to refresh the data shown in the grid view.



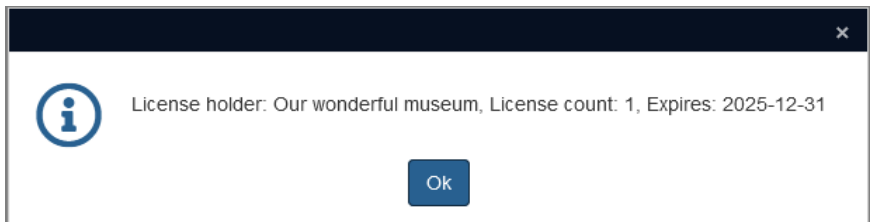
Select a single user and click the *Message* button if you'd like to notify that user of something. (You cannot select multiple users.)



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.



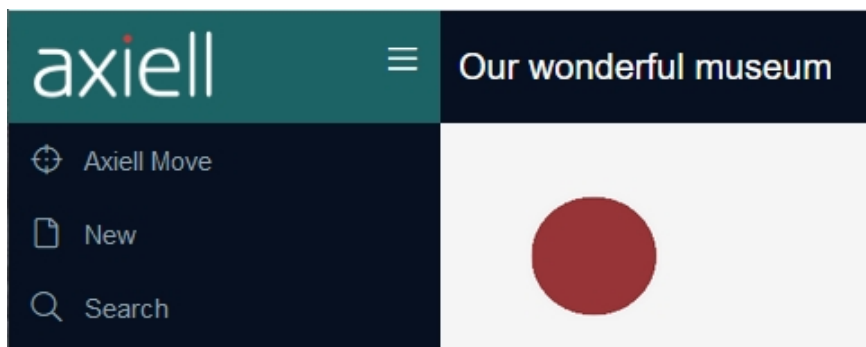
In *settings.xml* you can optionally include license information per `<SessionManager>` node. If you don't, the license will be implicitly unlimited. If you do, note that inclusion is only sensible in hosted environments where the customer cannot edit this information in *settings.xml*. An example of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



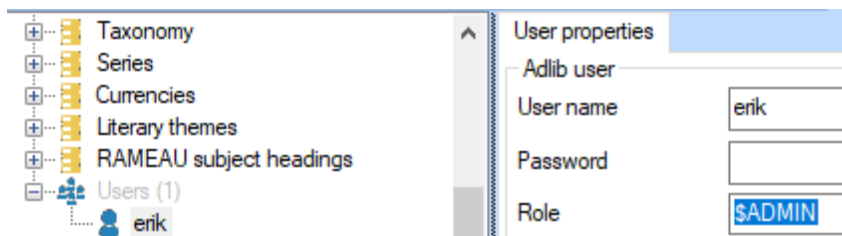
If the license has expired, users get a message when trying to log in (after which login fails):

When a user tries to log in while already as many other users are logged in as there are licenses, then also a message 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 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.

**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).



## Appendix A: preparing the Standard model application

This appendix is only relevant if you do a first-time installation of an Axiell Collections system (core software plus a Standard Model application). You can skip it if you already have a Collections system in place.

Out of the box, the current Standard model application (no version number, but it was released after model application version 5.2), needs some further work to get it ready for use. It's best to do this after you've completed the rest of the Collections installation and configuration.

---

### Populate two tables with default terminology

During the installation and configuration of Collections, probably a SQL .bak file (a backup file) will have been restored to initialize your new SQL database (see chapter 3.2). You will either have a largely empty database because there was no data to import or a database populated with migrated data from your institution.

In either case, if you are installing the current Standard model application as well (instead of model 5.2 or older), two new system database tables need to be populated with default values too: you can't do without this data. These might already have been included the .bak file (in which case no further action is required on this point) or have to be imported separately after finishing the complete Collections installation.

The relevant tables are the *sysoccddef* database table (called *System admin - occurrence defaults* in the Collections interface) and the *systemlists* dataset in the *thesau* database table (called *System lists* in the Collections interface).

If the *AC\_#\_initialisation\_SQL.bak* file was used (possibly included in the *\initialisation files* sub folder of the application root), these two system database tables will already have been populated so then you don't need to take further action.

(Using SQL Server Management Studio you can easily check if the *sysoccddef* table has data in it. If so, the *systemlists* dataset will probably have been populated too.)

If both datasets are still empty, you must use Axiell Designer and the three Axiell-supplied .dat (Adlib Tagged format) import files (sys-

*occdef.dat*, *systemlists-create.dat* and *systemlists-update.dat*) to import this default data in both tables. The files might be located in the *\initialisation files* sub folder of the application root directory. First run *sysoccdef.dat* and *systemlists-create.dat* and then run *systemlists-update.dat* to create the correct links between the earlier imported system lists records.

Note that migrated data might already have filled the *thesau* dataset in the *thesau* table with data, so if *systemlists* data is still missing then the records to import will have to be added in *systemlists* dataset and you **should not** clear the table before import. **Experience with creating and executing import jobs is required!**

---

## Relevant settings in Settings.xml

### AlwaysPromptBeforeSave

As mention earlier in this manual, for the current Standard model application it is a requirement for the consistency of your data that you set the `EnableResultSetEdit` setting in *settings.xml* to false.

```
<Setting Key="EnableResultSetEdit" Value="false" />
```

This will cause the *Edit mode* icon to disappear from the *Result set* context toolbar. This will apply to all data sources.

This setting is necessary in the Standard model application to prevent data issues that may arise from allowing edits in the *Result set* because certain data validations won't be executed there.

### Path

In previous versions of the model application, the root directory contained several .pbk sub folders, one for an *\archive*, an *\archive+museum*, a *\museum*, and an *\xplus* sub folder. In the Standard model application however, there is but a single application .pbk file, located in the *\main* sub folder of the application root directory.

So the path setting will always be:

```
<Setting Key="Path" Value="<my_application_path>\main" />
```

---

## Setting the Application Id

As mentioned above, in the Standard model application there is only a single unified application .pbk in the *\main* sub folder of the application root directory.



However, it is still necessary to set the licensed branch version of the application being installed. You must do this by setting the proper application ID in the .pbk file using Designer:



Valid options are:

- museum
- archive
- archive+museum
- xplus

---

## Uploaded templates setup

In the *Uploaded templates* data source, users can delete records for output templates they have uploaded. However, Axiell Collections cannot delete the actual template files themselves unless you edit the `\texts\reports.txt` file to replace the text “no\_path\_yet” with a full UNC path to `worddoc\uploadedtemplates\`.

See chapter 5.9 for the details.

---

## Default application permission groups

The Standard model application contains several default user permission groups. See chapter 2.2 in the *Release notes Standard model application.pdf* for more information about these groups.

When installing the Standard model application in an environment that uses Active Directory (AD) or Single Sign-On (SSO) user authentication, you will either need to map these default permission groups to existing AD roles, or create new AD roles to enable them.

If mapping to existing AD groups, rename these groups in the application .pbk accordingly.

For clients that do not want any permission controls in their system, rename the ‘App administrator’ role to an appropriate AD group shared by all users that should have access to Collections.

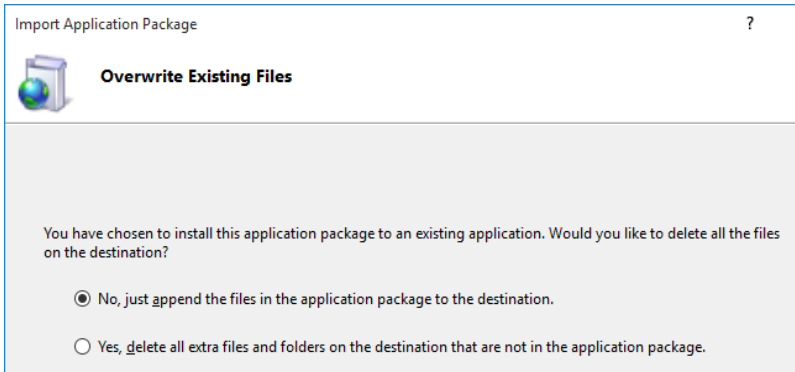


## Appendix B: updating to Collections 3.0

If you are just looking for some concise information about updating your existing Collections system from a core software version older than 3.0 to a 3.0 or higher version and you are already familiar with creating an IIS application for Collections, the following is all you need to know.

1. Make a backup of your existing *settings.xml* file from the Collections *\App\_Data* folder and store it away from the Collections folder. And if you are using custom background images for the Collections login screen, you should back up those as well from the *Content\Backgrounds\Custom* folder.  
Note that each IIS application has a different *settings.xml* file, so make backups of all of them and make it clear in the backup to which IIS application they belong.
2. Install the [.NET Core 8.0 Hosting bundle](#) (consisting of the ASP.NET Core Runtime for IIS support and the .NET Runtime).
3. Since the folder structure of Collections 3.0 is different from before, we recommend creating a new IIS application (with its own application pool preferably) for your existing (test) model application, although that's not a requirement: just updating an existing IIS application is easier, as long as you made a backup of your *settings.xml* file. (For a new IIS application you don't need to make a copy of your existing model application though.) Remember that no spaces or dots are allowed in the IIS application name, as usual.
4. Deploy the 3.# *AxiellCollectionsCoreWebDeploy.zip* file that you got from Axiell. (Note that the name of the zip has changed.) The wizard is slightly different but deployment works the same as before.

In the *Overwrite Existing Files* step, choose the first option if you already have a Axiell Collections 3.# or higher installation in place on that location, which you'd only like to update with a new version of the core software: any custom settings in *\Configuration\settings.xml* remain as they were.



Conversely, select the second option if you're updating a Collections pre-3.0 version to a 3.0 or later version.

5. After finishing deployment, copy the relevant backup *settings.xml* file to the new *\Configuration* sub folder in the new Collections folder structure. Of course, you can also create a new *settings.xml* file instead, but it's easier to use an existing one.  
(Any custom background images for the Collections login screen can be copied back to the relevant new *\wwwroot\Content\Backgrounds\Login\Custom* sub folder.)
6. You may need to tweak your *settings.xml* in the new installation a bit now: for Collections 3.0, the `BaseUrl` option is mandatory and must always contain the public base URL (case-sensitive) of the Collections application. For example:

```
<BaseUrl>https://museum.com/CollectionsProd</BaseUrl>.
```

7. Via Designer, check that the *Authentication method* for the .pbk has not been set to *None*, because this is no longer allowed by default. Instead of *None*, select one of the other methods (this should probably be *Active Directory* then).  
Also check that the *Encoding* property in the .pbk is UTF-8. If it isn't, you'll have to use the Designer Application character set conversion tool to change the encoding still.

For most update installations, this is all you have to do. Recycle the IIS application pool and check out your updated Collections installation in a browser.

Most common settings reside in the *settings.xml* file in the *\Configuration* sub folder. Some advanced settings, like logging, e-mail server configuration and multi-factor authentication, need to be done in an applicable settings json file, which can be edited or created in a text editor like Notepad++. (Any logging setting in *settings.xml* will be ignored.) There are different json files in different locations possible. Please see chapter 5.12 for more information about these files.

Recycle the IIS application pool after any editing of these files.

## Appendix C: possible IIS/Collections software issues

- **Updating Collections through IIS fails because some dll is still in use** – If, when installing a different version of Collections for an existing IIS application, IIS reports that the installation has failed because some dll is still in use, then stop the relevant IIS application pool before trying the update again: right-click the IIS application (underneath *Sites*), choose *Deploy > Recycle...* in the pop-up menu and then select *Stop application pool*. After installing the update successfully, you'll of course have to start the application pool again and also recycle it.
- **Error loading SqlServerSpatial160.dll (Error Code: 126)** - If you are installing Collections 1.19 or later on a Windows 2022 Server, you may receive the following error when attempting to run Collections: *Error loading SqlServerSpatial160.dll (Error Code: 126)*. This indicates that the latest Microsoft Visual C++ Redistributable Version is not yet present on the server. You will find the latest version here: [Latest supported Visual C++ Redistributable downloads | Microsoft Learn](#). Install the appropriate executable for your architecture (e.g. [https://aka.ms/vs/17/release/vc\\_redist.x64.exe](https://aka.ms/vs/17/release/vc_redist.x64.exe) for X64) and Collections should then run without problems. If the error persists after installing the x64 version, try installing the x86 version as well. Having both doesn't cause conflicts.
- **Cannot contact domain controller** - If a server error stating that the system cannot contact a domain controller to service the authentication request is received during the login, it means Axiell Collections has no access to the server which stores all network user names and passwords (aka the domain controller). Therefore, check whether the domain controller is accessible from the server on which Axiell Collections is installed.
- **Runtime error embedded in HTML code when using GIS functionality** – In the <DatabaseSettings> node, which you may encounter underneath the <AlmSettings><Gis> node in *settings.xml*, the <ConnectionType> sub node must be filled with either the value *SqlServer* or *None*, otherwise you'll get a runtime error when using the GIS functionality.
- **Server error in <your> application; runtime error** - This is a vague error on purpose. To get more information, go to the Collections installation folder on your web server, open the

web.config file in Notepad++ and add `<customErrors mode="RemoteOnly"/>` in the `<system.web>` node, save the file, wait for the application pool to recycle, open Collections in a browser on the web server itself and retry the action that resulted in the original error message. You should now get a more detailed report about the error. Check the rest of this possible issues list if your error has a known solution.

- **A potentially dangerous Request.Form value was detected from the client (Password=...)** – check if the password used for login into Collections doesn't contain any of the following characters: `<`, `>`, `*`, `%`, `\`, `?`, `&`, `:`
- **When opening Collections 3.0 or up only a white page is displayed** – In this case, a `BaseUrl` setting in *settings.xml* is missing or is incorrect. Correct the setting and recycle the application pool to solve the issue.
- **Fatal error and log mentions that encoding is not supported** – When on opening of Collections, the browser reports: "A fatal error occurred. The server encountered an internal error and was unable to complete your request.", while the log mentions that an encoding is not supported, then the encoding of application objects might not be the required Utf8. Check the *Encoding* property in the .pbk, using Designer. If necessary, you can change the encoding of database table structures (.inf files), application structures (.pbk files) and screens (.fmt files) simultaneously, with the [Application character set conversion tool](#) in Axiell Designer. (Do make a backup of your application folder before making these changes.) Then recycle the IIS application pool and check the results.
- **"Authentication not configured" message on login page** – If the login page of Collections 3.0 states: "Authentication not configured" and entry fields for credentials are missing, you should check that the *Authentication method* for the .pbk has not been set to *None*, because this is no longer allowed. Select one of the other methods (this should probably be *Active Directory* in your case), save the .pbk, recycle the application pool and try logging in again.
- **Saving a record after uploading a (large) image file causes an "Access to the path <folder\_name> denied error, after installing Collections 2.#"** – This is an access rights issue. Granting the *Modify* privilege to the App Pool user for the Collections *FileCache* folder (where "chunks" of uploaded large image files are temporarily stored) should resolve the issue.

- **When opening Collections, you get a *Failed to start web server – invalid path error*** - In some situations, multiple application configurations are deployed to target specific areas of the system. Then, instead of setting the path to a specific model application pbk folder (like *\xplus*), you can also set it to the model application root folder (containing multiple application pbk folders, like *\xplus*, *\library*, *\museum*, etc.) if you'd like to offer the user a choice of applications to log on to when opening Collections in the browser. Example:

```
<Setting Key="Path" Value="\\S01\Application\Production52" />
```

However, if this root folder contains any pbk file itself, you'll get this error. So make sure the root folder doesn't contain any pbk.